# Spatiotemporal properties of evoked neural response in the primary visual cortex

*Jean-Luc Richard Stevens*

Doctor of Philosophy

Institute for Adaptive and Neural Computation
School of Informatics
University of Edinburgh

2016

# Abstract

How is the visual world represented in the primary visual cortex of the primate visual system? When a novel visual stimulus is projected onto the retina, a complex, dynamic pattern of neural activity is evoked across the cortical surface. Understanding the spatiotemporal properties of this activity is a fundamental problem in neuroscience and it requires a unified framework to bridge the gap from single neuron activity to the response of the population as a whole.

The evoked cortical response of an individual neuron is determined not only by the particular properties of a visual stimulus and the internal response properties of the cell, but by the collective dynamics of an entire network of interconnected neurons. On longer timescales, this network structure is itself shifting as activity-dependent plasticity gradually shapes the connectivity between the cells as they respond to the visual environment.

The response of a single cortical neuron is then the outcome of an extended spatiotemporal history of the activity across an entire population, driven by the interplay between the neural activity itself and the plasticity of the network on short and long timescales. These interlocking processes shape the cortical response and the cortical structure in relation to the short-term and long-term history of visual input. In order to untangle this complexity it is useful to build simplified computational models that incorporate the essential features of these interactions.

The approach taken here is to bring these interlocking processes together in a single, unified spatiotemporal model of cortical activity. The aim is to relate the response of individual units to the response dynamics of the entire, spatially extended population while simultaneously bridging the gap from transient activity responses to the long-term development of the network structure. This attempt to unify broad spatial and temporal scales is novel and requires a synthesis that spans experimental and modeling approaches that are normally considered in isolation.

This thesis starts with background material describing how the early visual system is organized, together with a description of how neurons in the mammalian primary visual cortex develop their basic response properties. This is followed by a review of the relevant experimental data as well and a summary of relevant computational modeling work. This chapter covers the experimental and modeling literature regarding the electrical activity of single cells, the spatiotemporal response patterns observed across the cortical surface, and concludes by describing the emergence of feature selectivity across the cortical surface over development.

Following the background chapter, an important part of the research process itself is addressed, namely the need for an exploratory, yet reproducible workflow. Easy exploration of high-dimensional data is vital when validating large computational models as it is necessary to explore the space of model behavior and check that the necessary experimental constraints are satisfied. A new, general, and reproducible workflow is established in order to ensure that the scientific work presented in this thesis can be understood and used by future researchers in an extensible and maintainable manner.

Using these research tools, the task of bridging spatial and temporal scales is split into three distinct steps that are each assigned to a corresponding results chapter. The first of these chapters is an account of how the evoked response observed at the level of a single neuron relates to the overall population dynamics. The simple model presented aims to relate single-unit responses recorded using electrophysiology to the population response observed with optical imaging. Only data recorded in mature animals on short timescales is considered, allowing the network structure to be treated as static. The resulting model is the first to show how realistic single-unit responses can add up to the very different population-level evoked responses measured using voltage-sensitive-dye imaging over large cortical areas.

Next we turn to modeling how cortical structure self-organizes over slower developmental timescales into smooth orientation maps over the cortical surface. In order to properly evaluate developmental map models, a novel, automated map quality metric is presented that allows simulated maps to be quantitatively compared to experimentally recorded orientation maps across different mammalian species. This metric is used to analyze the components of the GCAL developmental model that forms the basis for the model presented in the final results chapter.

In the last results chapter, these threads are tied together into a developmental model that incorporates a more realistic model of time, including calibrated transient neural response as well as orientation map formation. The resulting unified model demonstrates how neural response in the mammalian primary visual cortex for large cortical populations may be accounted for by a simple model that bridges millisecond timescales to the developmental timescales necessary for the emergence of feature preferences.

# Acknowledgements

First of all, I give my sincere thanks to my first supervisor, James A. Bednar (Jim) for his tireless help and support over the past five years. He has consistently provided me with guidance and encouragement and has shown an immense amount of patience. I would also like to thank my second supervisor, Frédéric Chavane (Fredo) for his crucial scientific insights, and for inviting me to his lab in Marseille.

I would like to acknowledge all the members of the Institute for Adaptive and Neural Computation for providing a supportive academic environment and I would especially like to thank Peggy Seriès for her useful feedback at our yearly review meetings. A special thanks to the Neuroinformatics DTC for offering a flexible and engaging PhD program.

Thanks to Philipp Rüdiger for the many interesting and helpful discussions we have had and particularly, for our close collaboration on our software projects. I would like to thank my colleagues and friends, Nikos Gekas, Duncan Carmichael, and Robert Court for so many enjoyable conversations over our write-up lunches.

Finally, I am most grateful to my Mum and Dad for their support and for allowing me to stay with them over the last stages of my project.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jean-Luc Richard Stevens*)

# Table of Contents

# Chapter 1

# Introduction

The brain is an information-processing organ that integrates sensory information and controls behavior. In higher mammals, a significant portion of this function is directed by the cerebral cortex, the outer few millimeters of brain tissue composed of billions of neurons and trillions of synaptic connections. The cerebral cortex is known to be highly adaptable, and the neural responses within it are associated with a diverse set of faculties, including executive function, motor control, and sensory processing. Of all the cortical areas, the primary visual area (V1) is perhaps the most well studied.

Visually evoked activity in primary visual cortex arises due to interactions between a visual stimulus and the corresponding neural substrate. This neural tissue is defined by an extended network of cells that are richly interconnected, with connectivity patterns that are neither entirely genetically predetermined nor entirely random. Instead, much of the adaptability of the cortex and its connectivity patterns are grounded in a slow, ongoing learning process.

Much of this learning is activity driven, mapping each evoked activity response to tiny synaptic changes in the network which are then projected onto future evoked responses. This process suggests how the cortex is able to functionally adapt and perform useful computations: it engages in a continual learning process that collects information about the structure of the environment while simultaneously responding to it. It is this interaction between environment, neural structure, and evoked response that defines the process of vision as well as many of the other faculties of the neocortex.

This overall theoretical picture is supported by a vast scientific literature, with strong evidence supporting each piece of the puzzle. For instance, recording cortical activity in response to a visual stimulus is one of the core activities in experimental visual neuroscience. The assumption that this activity reflects the integration of in-

coming activity across synapses is one of the central dogmas of the field. Indeed, much of visual neuroscience is concerned with understanding how the activity evoked by a visual stimulus is integrated by neurons across their synapses.

An equally substantial literature is concerned with how neurons and synapses change over time. As outlined in Chapter 2, the idea that the synaptic structure of the cortical network is plastic and shaped through learning processes is supported by decades of experimental work. In primary visual cortex, the functional and structural properties of the neural tissue have been manipulated by modifying the long term visual statistics of the environment using dark rearing, monocular deprivation, and goggle rearing paradigms. These types of experiments are extremely difficult, involving long-running experiments that span the critical periods of development as the animal matures.

The time and expense of performing chronic experiments, especially for primates, has resulted in a clear divide in the scientific literature. On one hand, there is an extensive literature focused on the properties of the evoked activity, typically concerned with recording the response of neurons in the adult animal. On the other, there is a sparser developmental literature, featuring chronic experiments that record how neural responses change as the organism matures. Both these literatures contribute important insights to our understanding of the neural basis of vision.

One unfortunate consequence of this split is that theoretical frameworks are divided along similar lines. It is clearly important for theory to make contact with experiments, which helps explain this symmetry between theory and experiment. Yet given the general lack of overarching theories in neuroscience, and given what we do know about cortex and its functional properties, there is a clear need for suitable theoretical frameworks that can close this gap.

What computational models offer are a way to gain new insights into the process of vision by connecting theory to experiment, allowing different sources of experimental data to be integrated into a cohesive whole. Models can make unstated assumptions explicit and can help illuminate key principles. Ideally, by applying the right simplifications, a model can help cut through overwhelming biological complexity to suggest which of the many observations about a system are crucial for a given result. So far, however, models have not attempted to bridge the gap between the experimental data measuring evoked response over tens or hundreds of milliseconds, and experimental data that probes the developmental process over time scales of days or weeks.

To cover such a broad span of time on practical computing systems, such a model cannot also include a detailed account of each cell's intrinsic properties as well as the

detailed biophysics of their connectivity. What such a unified model *can* offer is the big picture, allowing different types of experiment to be related to each other within a single theoretical framework. If we can establish how the primary visual cortex works at a coarse, general level, it will be possible to add specific details as necessary, by using new experimental data to constrain model parameters and making use of improved computational resources to run bigger, more detailed simulations. Thus the models in this thesis will focus on simulating firing rates rather than individual neural spikes.

Building a framework that account for how neurons respond within a large, extended population timescales covering from milliseconds to weeks is no easy task. The purpose of this thesis is to demonstrate is that this problem can be tackled in a meaningful, extensible way that opens up exciting new lines of scientific enquiry.

## 1.1 Aims and structure

The aim of this thesis is to build a modeling framework that can simulate the evoked response of a spatially extended population of neurons in the primary visual cortex within an appropriate context. The context relevant to a visual neuron spans at least three different levels: (1) the evoked response of the neuron itself, along with the immediate response properties of the surrounding neural population and their connectivity to that particular neuron, (2) the history of spatiotemporal activity across the network that explains how this specific connectivity pattern arose, and (3) the long-term statistics of the visual environment, which shapes this connectivity and gives a neural response the context necessary to convey meaningful information about the visual world.

In addition to these primary aims, a core goal is to make this model simple, understandable, and extensible. Creating a general framework is only useful if it opens up new possibilities for answering research questions posed by other scientific researchers. Simplicity and reproducibility are two aims that do not target specific scientific results but do target something equally, if not more important, namely the scientific process itself.

This thesis has the following chapters, structured in a way designed to bridge two very different forms of computational modeling in a simple, reproducible, and understandable way:

**Chapter 2** The Background chapter starts with a description of different modeling ap-

proaches and their relation to each other, with special attention to developmental modeling approaches. This is followed by the anatomical background and then the background information regarding the key models and experimental data referred to throughout the rest of the thesis. This material is split between material that accounts for the evoked response properties of neurons and material that concerns cortical development; unfortunately there is very little intersection between the two.

**Chapter 3** This chapter discusses the importance of scientific reproducibility and introduces Lancet and HoloViews, two new open-source Python-programming tools developed during this thesis project that greatly assist scientific productivity, reproducibility, and communication within a literate programming environment. The chapter focuses on how these tools were used to enable the scientific work presented in this thesis, but they are also very powerful in general, and actively used by researchers across different disciplines, worldwide.

**Chapter 4** This chapter introduces the SIRD model, which links single unit, firing-rate responses generated by the well-validated IRD model to the corresponding population response as observed with voltage-sensitive–dye imaging (VSDI). This model has a very simple mathematical formulation that incorporates extensive calibration against available experimental data. It is the first model of the spatiotemporal VSDI response that accounts for the mechanisms involved in relating single-unit activity to the bulk population response.

**Chapter 5** This chapter analyzes and validates a self-organizing map model called GCAL, which unlike the SIRD model accounts for developmental processes and includes a mechanistic subcortical pathway. A new map-quality metric is introduced, which was used to complete the analysis needed to validate this model. This led to the publication of the GCAL model, which is simpler, more biologically plausible, and more robust than its predecessors. This analysis is used again in the rest of the thesis and GCAL is the basis of the final TCAL model presented in the following chapter.

**Chapter 6** This chapter further simplifies GCAL while improving the way it processes temporal events to make it more suitable for modeling evoked response properties, resulting in an approximately equivalent model called CGCAL. CGCAL, in turn, is used to build a new, temporally calibrated model called TCAL.

TCAL features plausible firing rate profiles on the timescale of the evoked response, as well as self-organization of connectivity on the timescale of development. TCAL is analyzed using the map metric introduced in the previous chapter and it is demonstrated that this model links to the SIRD model discussed in the fourth chapter, providing a mechanistic implementation of a developmental process that leads to the type of processing supported by SIRD.

**Chapter 7** This chapter discusses the overall results from the thesis, putting them into context, and suggests various possible directions for future work. The intention is to provide suggestions for exciting new research projects that can build on the framework that is developed in this thesis.

**Chapter 8** The final chapter provides a short conclusion intended to summarize the main contributions of this work.

**Appendix** For the convenience of the reader, this section includes three papers relating to this work that were published over the course of this project.

Another way of understanding the structure of the thesis as well as the relationship between the various models provided is using the breakdown in Table 1.1. This table lists the models in the order they are presented, including a breakdown of their various key features. What is meant by each feature listed will be made clear in Chapter 2. What is important to note is that the SIRD model extends the IRD model, and that the final TCAL builds on the CGCAL model which in turn builds on the GCAL model.

The goal of the final TCAL model is to demonstrate how firing-rate models can connect the response profiles of single units in the evoked response to the structural changes that occurs across large populations of neurons in the primary visual cortex over long, developmental time periods. This framework is designed to be simple and extensible, and it is hoped that future researchers will use TCAL as a platform to pose new and groundbreaking research questions that could not be tackled using existing computational modeling approaches.

| | *IRD* | *SIRD* | *GCAL* | *CGCAL* | *TCAL* | |
|---|---|---|---|---|---|---|
| Simulation of several $mm^2$ of cortex | x | ✓ | ✓ | ✓ | ✓ | Development |
| Mechanistic subcortical pathway | x | x | ✓ | ✓ | ✓ | |
| Orientation maps | x | x | ✓ | ✓ | ✓ | |
| Encodes first-order visual statistics | x | x | ✓ | ✓ | ✓ | |
| Specific lateral connectivity | x | x | ✓ | ✓ | ✓ | |
| Diversity of tuning properties | x | x | ✓ | ✓ | ✓ | |
| *Continuous model of time* | ✓ | ✓ | x | ✓ | ✓ | Evoked |
| *Plausible firing rate profiles* | ✓ | ✓ | x | x | ✓ | |
| *Calibrated against VSDI data* | x | ✓ | x | x | x | |

Table 1.1: **Key features of the five rate-based models in the order they are presented in this thesis.** The IRD model is described in Chapter 2 as a way of summarizing the firing rate properties of individual neurons. Next, the SIRD model is introduced in Chapter 4, extending the IRD model to model the response of a spatially extended population of neurons. In Chapter 5, the focus switches to the developmental process using a model called GCAL, simulating a spatially extended population of neurons over long timescales. In Chapter 5, this model is adapted to operate on a continuous timebase to define the CGCAL model. Lastly, a temporally calibrated version of CGCAL called TCAL is constructed which shows how a mechanistic, developmental model can be connected back to the SIRD model. As TCAL inherits many of its properties from the GCAL model, the main focus of this thesis is driven by the features relating to the evoked response, indicated in the italic font.

# Chapter 2

# Background

The goal of the final model presented in this thesis is to integrate a wide range of theoretical and experimental findings into a single mechanistic framework. In order to assess the validity of any computational model it is necessary to understand its structure, behavior, and overall scope. This chapter aims to cover the relevant literature necessary to understand the work presented in later chapters.

The structure of a model may be understood at a mathematical or algorithmic level and for mechanistic models, biological plausibility should also be evaluated. The behavior of a model may be assessed in relation to experimental measurements used to reveal the functional dynamics of the biological system. Which set of experimental constraints are appropriate for assessing a model's plausibility are determined by its intended scope.

This chapter starts with a brief overview of four different types of cortical computational modeling approaches, followed by an overview of the relevant anatomy of the early primate visual system. This data will be used to justify the structural components of the models presented in subsequent chapters. The rest of this chapter outlines the relevant experimental findings and related theoretical concepts that will be referred to later on in the thesis. Data recorded from macaque monkey will be presented where possible, since this is the species that will be targeted for modeling, but data from other species will be considered for the many cases when macaque data is not available.

Note that no single experimental technique has been demonstrated in primate to record a large cortical area over many weeks while simultaneously offering the spatiotemporal resolution to resolve single neurons and their responses within individual fixations. Due to the tradeoffs in spatial and temporal resolution across different experimental techniques, it is necessary to consider several different sources of experimental

data together.

This need to combine results across experimental techniques is driven by the incredible biological complexity that underpins cortical activity. At each moment, the neural response is determined by the collective interaction of a large interconnected population of individual cells that are each shaped by their surrounding environment and long-term history of synaptic inputs.

The experimental data across spatial and temporal scales will be collected together in two main sections. First the evoked response will be considered across spatial scales in adult macaque monkey, offering an insight into how V1 responds at the end of the developmental process. In the second section, cortical development is considered using the scarce data obtained from chronic studies.

The experimental difficulty of recording chronically from single cells is discussed, before the available developmental data is presented in the form of chronic orientation map measurements using optical imaging. Finally, existing self-organizing models of orientation map models are described, including the details of the GCAL model that forms the basis of the model presented in Chapter 6.

## 2.1 Computational modeling approaches

The process of scientific research is based on the twin pillars of theory and experiment. Neuroscience is no exception, and both experimental and theoretical neuroscientists work together towards an improved understanding of the nervous system, driven and validated by experiment.

A computational model represents a concrete implementation of a theory that may be used to integrate information across different experiments, make novel predictions and bridge the gap between theoretical concepts and empirical data. There are many different types of computational model and this section will briefly describe some of them, in order to place the work presented in this thesis in a broader context.

The following categorization of approaches is relevant both for neural map models (Bednar and Williams, 2016) and more widely across computational neuroscience. These definitions constitute a partial and not mutually exclusive list of the different modeling paradigms. In other words, any particular model may have components that partially satisfy any of the following criteria in varying degrees:

**Phenomenological** These models designed to describe or reproduce experimentally

observed behavior without necessarily any reference to the underlying mechanisms in the biological system. These models are also sometimes called "descriptive" models, as they describe a phenomenon without addressing its physical basis. The "invariant response descriptive model" described in section 2.3.1 and used in Chapter 4 is one example of such a model.

**Normative** These models are aimed at explaining some functional criterion that is believed to be essential for the operation of a biological system. Normative models do not need to be derived with reference to the structure of neural elements or circuits and are therefore distinct from mechanistic models.

**Mechanistic** These models explicitly claim an isomorphism between elements of the model and the structure of nervous system. A good mechanistic model must also be a good phenomenological model, i.e., matching behavior as well as just mechanisms.

**Developmental** Developmental models aim to explain how adult-like circuitry or mechanisms emerge from some initial conditions that are simpler or less well ordered. Most developmental models are also mechanistic, claiming an isomorphism between the initial stage and early stages in the organism's life cycle, but some are much more abstract.

The aim of this thesis is to build a mechanistic, developmental model of evoked neural activity in V1. In general, all models have some phenomenological or descriptive component, as every model has to begin with some set of initial assumptions regarding natural behavior. For instance, a detailed compartmental model of a neuron needs to assume the existence of ions with certain behaviors that are not derived from some deeper theory of reality such as quantum mechanics, but are simply assumed. The firing-rate models considered in Chapters 5 and 6 assume that some important aspect of the behavior of neurons can be summarized by its firing rate.

Normative models are often based on abstract criteria regarding optimality, typically involving the minimization or maximization of some objective function. One famous example is the idea that receptive field formation is determined by learning under a sparsity constraint while attempting to minimize the image reconstruction error (Olshausen and Field, 1996). Normative models do not need to be mechanistic and mechanistic models are not necessarily normative, but a complete explanation would ideally include both if the behavior is of value to the organism. For this thesis, we will

focus on developmental, mechanistic models to try to connect behavior across a wide range of time scales, but we will come back to the issue of normative models for these phenomena in the final discussion.

### 2.1.1 Mechanistic modeling

A model may be described as mechanistic if the structure of the model mirrors the relation of physical entities in the biological system. For computational models of the cortex, mechanistic models thus typically focus on neurons and their connectivity in a network. One common approach is to model such a network as a population of interconnected spiking elements, where individual action potentials are simulated.

There are many different spiking model types, ranging from detailed conductance-based approaches such as multi-compartment models, to approaches that greatly simplify the biophysics of action potential generation such as integrate-and-fire networks. For a review of the simulation strategies and algorithms used in spiking network model simulations, see Brette and others (2007).

In general, the greater the detail of a spiking network model, the greater the demand on computational resources and the greater the requirement for experimental data to suitably constrain the biophysics of all the model components. These technical challenges make it impractical to simulate the detailed spiking response of cortical tissue on a spatial scale of millimeters over long periods of simulation time such as hours, days, and weeks. For this reason, spiking models are not a suitable platform for examining the timescales that will be considered later on in this thesis.

Another mechanistic approach is to approximate the behavior of neurons directly at the level of their firing rates, further simplifying the biophysics of the model elements. Network firing-rate models typically use individual units to represent small collections of neurons instead of individual cells. With more spikes to consider per unit time, this approach helps improve the validity of representing the spiking activity as a single floating-point number per model neuron. Moreover, at least on general-purpose computing hardware with floating-point processors, simulating a firing-rate network is much less costly than modeling individual spikes at the level of individual cells. Moreover, many of the experimental analyses in wide use, even those for measuring very precise temporal phenomenal such as PSTHs (peri-stimulus time histograms), effectively use a firing-rate level of description. Together these properties make firing-rate approaches particularly suitable for building mechanistic developmental models,

which we will consider in the next section.

### 2.1.2 Developmental modeling

Unlike many mechanistic models that simulate the neural activity on the scale of milliseconds up to seconds, developmental models typically simulate the structural changes to the nervous system that occur on the timescale of hours, days, and weeks. This is because developmental models aim to explain how adult-like circuitry arises in an organism from an earlier stage of maturation. The initial condition of the developmental models we will be considering is not concerned with the initial emergence of neural cells in the cortex, but will focus on how their connectivity later changes as the organism matures.

In order to make simulations feasible, mechanistic developmental models have simplified cellular biophysics and often simulate neural activities in terms of firing rate. What is lost in biophysical detail is compensated for by the different types of phenomena and new scientific questions that developmental models are able to address.

In cortical modeling, developmental models can be used to investigate the remarkable plasticity and functional flexibility of the neocortex. One striking illustration is provided by a set of experiments that induced visual projections into the auditory cortex of ferret, resulting in the development with visual receptive fields similar to those of complex cells of primary visual cortex in auditory cells (Sur et al., 1988; Roe et al., 1992). In humans, there is similar evidence for cross-modal compensation effects, such as evidence for altered visual processing in the congenitally deaf (Karns et al., 2012).

Similar effects have been demonstrated in developmental models, where the same model results in different learned features, patterns, or behavior according to the training statistics (Bednar, 2012; Miikkulainen et al., 2005). For instance, it has been shown that simulated orientation maps acquire similar biases when trained with skewed orientation statistics, as experimentally observed in goggle-rearing experiments (Stevens et al., 2013b; Tanaka et al., 2009).

These examples are compelling demonstrations that the structural and functional properties of cortex are plastic and are emergent properties of the developmental process. In turn, developmental models are the only way to explore how the structure of the nervous system depends on the statistical structure of the environment, which has important philosophical implications.

Firstly, this dependence suggests a way to link mechanistic and normative models.

A mechanistic, developmental model with plasticity can potentially show how a concrete, biologically plausible mechanism implements a normative criterion, such as the expression of receptive field structure in terms of natural image statistics (Hyvärinen et al., 2009).

Secondly, a developmental model can explain the causal link between neural structure and the sensory input that has been received from the external world via plasticity. This link is what makes a cortical area such as the primary visual cortex be about vision and the receptive fields in ferret auditory cortex also be about vision after suitable experimental manipulation. This issue regarding what neural processing is *about* is closely related to the philosophical question known as "symbol grounding".

The philosophical debate around the symbol grounding problem originated with John Searle's "Chinese Room argument" (Searle, 1980). This thought experiment claims to demonstrate that a traditional computer program cannot generate its own semantics, whereas natural systems, such as the human brain, can. Framed in another way, the issue is to understand how the components of a computation can meaningfully refer to the appropriate entity in the external world.

In a non-developmental, mechanistic model where synaptic structure has not been shaped by visual input, there is a similar problem. That is, how can the computation performed by such a neuron truly correspond to vision if the synaptic structure of the computation has at most an accidental relationship to anything visual, rather than a causal relationship as in developmental models?

The purpose of this thesis is to show how it is possible to construct a single model that captures this relationship between visual input and cortical structure on the timescale of development, and then to relate this network structure to the evoked response addressed by other types of mechanistic model on short timescales. These results will establish a new class of mechanistic model that is able to explore scientific questions that were outside the scope of all previous modeling approaches, helping us get closer to a true understanding of how the brain is constructed to represent its inputs in its evoked responses. To explain how these models relate to the underlying biological systems, the following sections will summarize the relevant biological results.

## 2.2 Anatomical background

The neocortex is the highly convoluted layer of neural tissue covering the surfaces of the cerebral hemispheres. The cortical surface is composed of numerous regions asso-

ciated with different faculties including areas involved in primary sensory processing, cognitive and linguistic performance, and motor output. This remarkable diversity in function is supported by a laminar organization that remains remarkably constant across the entire cortex.

The primary visual cortex (V1) is one of the most widely studied cortical areas due to the relative simplicity of the afferent pathway and the ease with which visual stimuli can be controlled and manipulated. In addition, the surface of V1 is readily accessible once a suitable opening in the skull has been made, enabling a number of different experimental approaches including electrophysiology and optical imaging techniques.

This section outlines the anatomical structure of the mammalian early visual pathway involved in the transmission of information from the visual environment to the cortical neurons of V1. This summary covers the background material necessary to understand the various components of the mechanistic models that will be discussed later on and puts the experimental data used for calibration in context.

Although there is anatomical variation between different mammalian species and the specific goal is to account for the cortical response observed in macaque, this section is general enough to describe the early visual pathway of any mammal that has smooth, well-organized orientation maps. This level of generality is deliberate, as experimental data is not always available for macaque, especially when considering chronic recordings needed to calibrate the developmental process. In particular, the chronic orientation map recordings described in section 2.4.1 are only available for ferret.

Figure 2.1 shows how the photoreceptors of the retina of the eyes connect via the retinal ganglion cells (RGCs) to the lateral geniculate nucleus (LGN) via one-to-one projections which in turn connect to primary visual cortex via the optic radiations. The information from the two eyes splits at the optic chiasm so that the left visual hemifield maps the right hemisphere and vice versa.

The classical receptive field of a visually responsive cell corresponds to the best stimulus pattern found to evoke a response. Figure 2.2A shows a schematic of the center-surround receptive-field structure typical of a mammalian LGN ON cell as well as an example recorded from cat. Part B of the figure shows the corresponding schematic for a V1 neuron composed of an ON and OFF lobe as well as a typical example of an oriented receptive field recorded from simple cell in area 17 (cat). This elongated, Gabor-like receptive field is also observed in macaque, as can be seen in the simple cell spatiotemporal RF in Figure 2.2C.

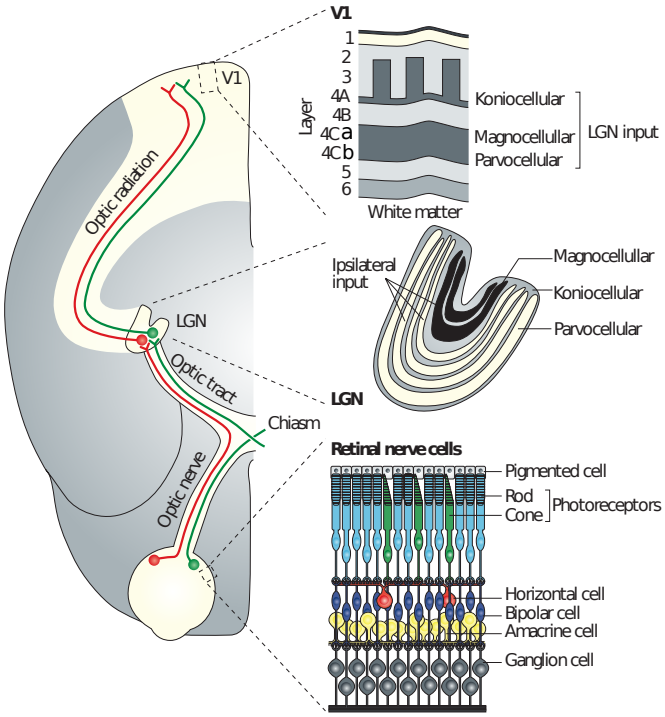


Figure 2.1: **The mammalian early visual pathway.** Anatomy of the visual pathway from the photoreceptors in the retina through the lateral geniculate nucleus (LGN) in the thalamus to the primary visual cortex (V1). On the right, the anatomical structures at each stage of visual processing are shown. The top right schematic shows the photoreceptors at the back of the retina where light is transduced into an electrical signal which passes through the bipolar cells to the ganglion cells. The axons of the ganglion cells form the optic nerve which projects to the LGN shown in the middle schematic on the right. The LGN has a laminar arrangement where each layer aligns input from both eyes to form a retinotopic map of the contralateral portion of the visual field. The majority of the axons from LGN neurons then project via the optic radiations to V1, terminating in layer 4. The organization of the parvocellular, magnocellular, and koniocellular pathways is shown in both the LGN and V1 schematics, where the shading indicates the pattern that emerges when V1 slices are stained for the metabolic marker cytochrome oxidase. Reproduced from Solomon and Lennie (2007).

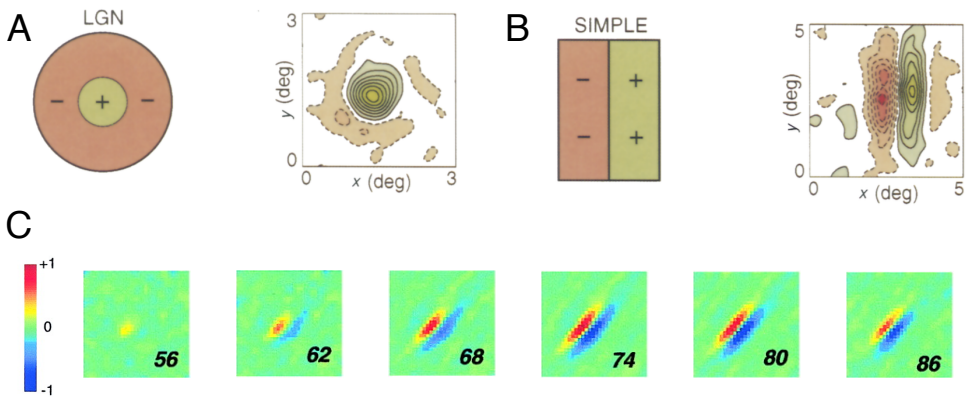


Figure 2.2: **Example receptive fields in the LGN and V1.** (A) Schematic of a typical LGN center-surround receptive field next to an example recorded from an LGN ON cell. (B) Schematic of a typical V1 simple cell receptive field next to an example recorded from a V1 cell. The two examples are shown are recorded in cat and reproduced from DeAngelis et al. (1995). (C) A spatiotemporal receptive field recorded from a macaque simple cell using subspace reverse correlation. The inset number shows the time in milliseconds at which the RF is computed. Reproduced from Ringach (2002).

The orientation-selective cells in V1 have a spatial organization across the cortical surface that falls into two classes when considering mammalian species. Rodents, for instance, have a "salt and pepper" organization where orientations appear randomly distributed down to the cellular level, as shown in Figure 2.3A. In contrast, carnivorans and primates such as macaque have a smooth orientation map organization as seen in Figure 2.3B and C. As the model presented in this thesis aims to model the cortical response in macaque, any orientation maps of the model should have this sort of smooth organization.

Smooth orientation maps have more identifiable structure than salt and pepper arrangements. In particular, as the map varies smoothly, hypercolumns can be identified as a continuous region over which the full set of receptive-field parameters are covered. In the case of hypercolumns, this corresponds to the average distance over which the orientation preference cycles over 180°. The corresponding circular feature where a 180° change in orientation preference is observed around a point is called a pinwheel. Pinwheels have two different polarities depending on whether the orientation preference increases or decreases when circling the pinwheel center clockwise. Examples of both these features are shown in Figure 2.3B. The orientation selectivity map is also smoothly varying as shown in Figure 2.3C.

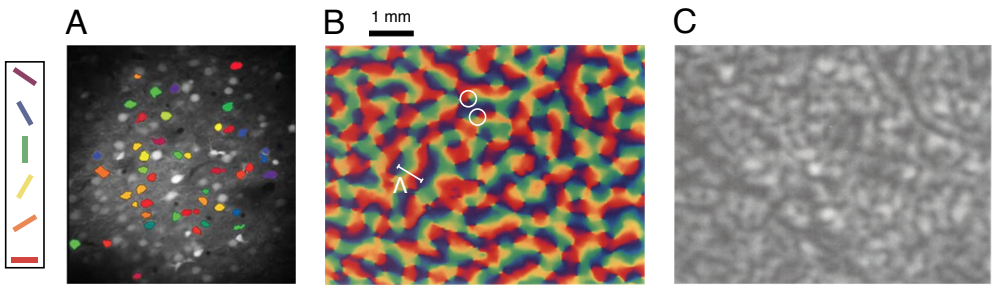


Figure 2.3: **Example orientation maps in rat and macaque V1.** (A) "Salt and pepper" arrangement of orientation preference in rat V1 as recorded with two-photon calcium imaging down to subcellular resolution. Preferences are indicated by the cyclic color key on the right. Reproduced from Ohki et al. (2005). (B) Orientation preference map recorded using optical imaging in anesthetized macaque using the same color key. The distance $\Lambda$ covers a change of $180^\circ$ and corresponds to the hypercolumn distance. The white circles mark two pinwheel locations around which all preferences are represented. These two pinwheels have opposite polarities with a clockwise progression moving up the color key for the left pinwheel and a clockwise progression down the color key for the pinwheel on the right. (B) The corresponding orientation selectivity map. Reproduced from Blasdel (1992b).

In summary, the visual system is composed of a large population of neurons with a diversity of receptive field and tuning properties. There is organization that can be observed at the level of individual neurons such as the receptive fields of a particular cell shown in Figure 2.2 and there is organization that is only apparent across a large population of cells, such as the orientation maps shown in Figure 2.3.

From this evidence, it is clear that there are different spatial scales relevant to the visual response, ranging from single-unit recordings to measurements of entire feature maps. In the next portion of this chapter, the ways activity can be recorded across these different scales will be discussed. This will included a common experimental approach for measuring activity at the level of an individual neurons and then a discussion of optical imaging techniques used to record from a large, spatially extended population of neurons at once.

## 2.3 Dynamics of the evoked response

The first step toward validating a model of neural response in V1 is to consider the available experimental data in the fully developed, adult animal. In this section, the ob-

served experimental responses to artificial stimuli in adult macaque V1 are discussed, both at the level of individual neurons and across a spatially extended population.

First the local spiking response will be considered in terms of the peristimulus time histogram (PSTH) profiles of both simple and complex cells in macaque. The experimentally observed PSTH profiles are summarized by the invariant response descriptive model developed by Albrecht et al. (2002).

Next the evoked dynamics across a large population of neurons is considered using voltage-sensitive–dye imaging, also recorded in macaque. This technique captures the evoked pattern of response over several square millimeters of the cortical surface with a high temporal resolution. The key properties of the observed spatiotemporal dynamics are then summarized as a function of stimulus contrast.

These two experimental techniques yield very different temporal profiles for the evoked response. Understanding how these two sources of data can be consistently accounted for within a single model of the evoked response is the basis of Chapter 4.

### 2.3.1 Local spiking responses

The temporal properties of the spiking response of a neuron can be recorded using an electrode with a high temporal resolution, then expressed as a peristimulus time histogram (PSTH). Such recordings typically have a high enough temporal resolution to resolve individual action potentials in a localized volume of neural tissue. One way to begin quantifying the dynamics of the response of individual neurons in the visual system is to examine the properties of PSTH profiles evoked by an appropriate test stimulus. In this section, the PSTH profiles of both LGN and V1 neurons will be presented.

#### Spiking profiles in the LGN

The mechanistic models we will consider later on simulate the propagation of activity from the photoreceptors in the retina to V1 via the LGN. Therefore, in order to understand what drives the spiking responses in V1 mechanistically, it is first useful to examine the spiking response profiles in the LGN.

Figure 2.4 shows average PSTH profiles for magnocellular and parvocellular neurons in macaque LGN. Both types of cell have a peak in spiking activity although the ratio of the peak to the sustained response is lower in magnocellular neurons. These PSTHS have been plotted on a 150 millisecond axis to allow easy comparison with the

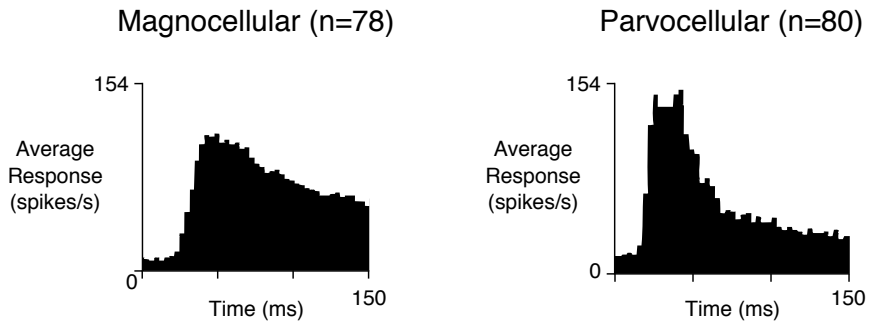


Figure 2.4: **Average PSTH profiles for magnocellular and parvocellular LGN neurons in anesthetized macaque.** Cells were stimulated by $0.25^\circ$ radius spots that were either brighter (ON) or darker (OFF) than the background by $28\ \mathrm{cd}/\mathrm{mm}^2$. On the left, the average PSTH profile for 78 magnocellular neurons is shown. On the right, the average PSTH for 80 parvocellular neurons. Data reproduced from Maunsell et al. (1999).

V1 PSTH profiles described in the next section.

#### The invariant response descriptive model

The invariant response descriptive (IRD) model of Albrecht et al. (2002) offers a mathematical description of the experimentally observed spiking response of V1 cells. This phenomenological model summarizes the observed PSTH profiles as a function of stimulus contrast for simple and complex cells of adult cat and monkey. For a population of 50 cells, this model was found to account for approximately 94% of the variance observed across the microelectrode recordings and will serve as a way to quantify the properties of "typical" PSTH profiles.

The temporal spiking response profiles of cortical neurons vary in both shape and amplitude according to the particular cell that is being recorded, as well as the shape of the driving stimulus. It is nonetheless useful to try to capture the general properties of observed PSTH profiles with a simple descriptive model.

The invariant response descriptive (IRD) model is a phenomenological model that captures the shape of a typical PSTH profiles over 200 milliseconds in response to the presentation of a stationary sinusoidal grating pattern at a fixed contrast. First, the profile shape is approximated in a piecewise manner using a Gaussian function up to the peak value at $\tau_c$ and a "$\frac{1}{2}$ Gaussian" profile to capture the sustained response

thereafter:

$$r_t(t) = \begin{cases} \exp\left(-\ln 2 \frac{(t-\tau_c(c))^2)}{\sigma_a^2}\right) & t < \tau_c \\ \exp\left(-\ln 2 \frac{(t-\tau_c(c))^2)}{\sigma_b^2}\right) + (1-\alpha)\exp\left(-\ln 2 \frac{(t-\tau_b(c))^2)}{\sigma_c^2}\right) & t \geq \tau_c \end{cases} \tag{2.1}$$

The time to the peak response $\tau_c$ is described with a contrast-dependent offset computed according to an inverted Naka-Rushton equation. The relative response amplitude $r_c(c)$ is defined in a similar way:

$$\tau_c(c) = \tau_{max} - \tau_{shift} \frac{c^\varepsilon}{c^\varepsilon + s_{50}^\varepsilon} \tag{2.2}$$

$$r_c(c) = \frac{c^n}{c^n + c_{50}^n} \tag{2.3}$$

The PSTH profiles are then described as the product of these terms offset by a fixed response $r(t,c) = r_{max} r_c(c) r_t(t) + r_0$ where $r_{max}$ is used to scale all the responses by a constant factor.

The mean parameter values, the bounds at one standard deviation and the corresponding median values given in Albrecht et al. (2002) are as follows: $\sigma_a = 19.0 \pm 1.9.1, 13.6; \sigma_b = 761 \pm 76.5, 543; \alpha = 0.27 \pm 0.03, 0.23; n = 2.5 \pm 0.18, 2.2; c_{50} = 38.7 \pm 3.51, 32.3; \tau_{max} = 121 \pm 4.53, 114; \tau_{shift} = 65.3 \pm 3.48, 61.2; \varepsilon = 1.80 \pm 0.28, 1.18; s_{50} = 24.6 \pm 3.27, 23.1; r_{max} = 81.8 \pm 12.2, 50.9$. Note that the median is outside of one standard deviation from the mean for nearly all of these parameters.

Taking the median values for these constants, we can use this model to generate a set of temporal response profiles as a function of contrast as shown in Figure 2.5. As expected, there is an onset peak that occurs before the first hundred milliseconds at high contrast, followed by a sustained response. Note that this model only captures the onset response when the stimulus is turned on, and does not capture how the response decays once the stimulus is turned off.

This concludes the background material regarding single-unit activity recording, as the IRD model is a sufficiently detailed level of description for the firing-rate models presented in this thesis. Next we turn to an experimental technique used to record activity from entire populations at once.

### 2.3.2 Voltage-sensitive–dye imaging

Voltage-sensitive–dye imaging (VSDI) is an optical imaging technique that measures the neural activity over a large cortical area with a high temporal resolution. Unlike

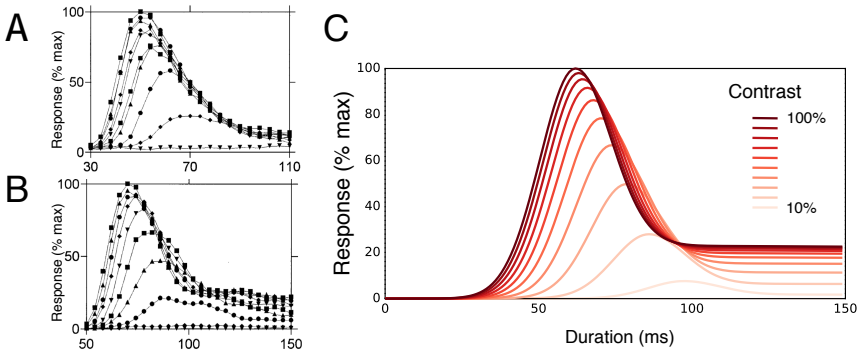


Figure 2.5: **Behavior of the invariant response descriptive model for ten different contrasts**. Each plot starts at 10% contrast and ends at 100% contrast in steps of 10%. (A) Example PSTH record from a macaque simple V1 cell across all ten contrasts where all PSTHs are onset aligned. (B) Similarly aligned example PSTHs recorded from a macaque V1 complex cell. (C) Output of the IRD model, replicated using the published median parameter values. Experimental data reproduced from Albrecht et al. (2002).

the intrinsic optical techniques suitable for chronic studies such as those described in section 2.4.1, voltage-sensitive–dye imaging requires the direct application of a dye to the cortical surface.

The voltage-sensitive dye binds to the external surface of all the cell membranes it comes into contact with (including both neurons and glial cells), acting as a transducer from electrical potential to optical signal. The dye responds rapidly to changes in the local voltage, giving the technique sub-millisecond resolution over several millimeters squared of cortex. A good review of the technique may be found in Grinvald and Hildesheim (2004) and Chemla and Chavane (2010b).

The optical signal indicates the voltage change across the whole tissue, as a mixture of both subthreshold and spiking response. As the dye seeps into the neural tissue from the cortical surface, there is a depth-dependent fluorescence gradient, with little dye binding at depths greater than 800 µm. The charge-coupled device (CCD) records the photonic flux from above the cortical surface, which results in a signal that is biased towards activity in superficial layers.

The spatial resolution of a typical frame output by the CCD is around 20µm – 50µm per pixel, where each pixel is integrating the photonic emission from a small cortical volume. The scale of scattering of photons in the neural tissue is approximately 200 µm (Orbach and Cohen, 1983) which determines the primary limiting factor in the spatial resolution of the technique.

These features of VSDI make it an excellent way to record the evoked activity dynamics of a large cortical population. The indiscriminate binding of the dye to all available cell membrane allows recording of the bulk electrical response, but it also complicates the process of tracing the exact contributions of different neural populations to the overall signal.

### Spatiotemporal dynamics of the VSDI response

A preliminary step towards understanding the spatiotemporal dynamics observed with VSDI is to observe the evoked activity in response to a simple stimulus that is flashed on and off. A simple stimulus protocol helps reveal the basic properties of the VSDI signal in the absence of complicating factors such as varying stimulus motion or shape. Even using the simplest possible flashed stimuli, the observed spatiotemporal dynamics will still depend on factors such as the stimulus size, retinotopic location, the spatial profile of the stimulus, the duration of presentation and the stimulus contrast.

Two different sources of experimental data obtained from two different laboratories will help reveal the general properties of the VSDI response in macaque V1 (Sit et al., 2009; Reynaud et al., 2012). Both of these sources of data will help constrain the spatial model of population response presented in Chapter 4.

**Sit et al. (2009)** The VSDI data published in Sit et al. (2009) is summarized in Figure 2.6, showing the response evoked by a flashed Gabor stimulus in awake, fixating macaque. A $0.167^\circ$ Gabor stimulus at $2.2^\circ$ eccentricity was turned on at time zero and sustained for 200 milliseconds with the VSDI response recorded over a period of 500 milliseconds. The spatial profile of the VSD signal at the time of peak response was found to be approximately Gaussian as shown in Figure 2.6A–C. Six different contrast levels were used with subfigures A and B of Figure 2.6 showing the peak spatial response evoked by 6% and 100% contrast Gabors respectively.

The annular regions of interest (ROIs) used in the spatiotemporal analysis are shown in subfigure C. The central region is circular, with all remaining ROIs constructed by intersecting a millimeter-wide strip with nested concentric rings, each ring incrementing the radius by 0.5 millimeters. In Figure 2.6D, the average responses in the six ROIs are presented for the whole 500 millisecond period across all six different stimulus contrasts. The average response of each ROI is shown as a thin strip, with the mean central circular ROI response shown and the mean response of the most distal annulus shown at the bottom.

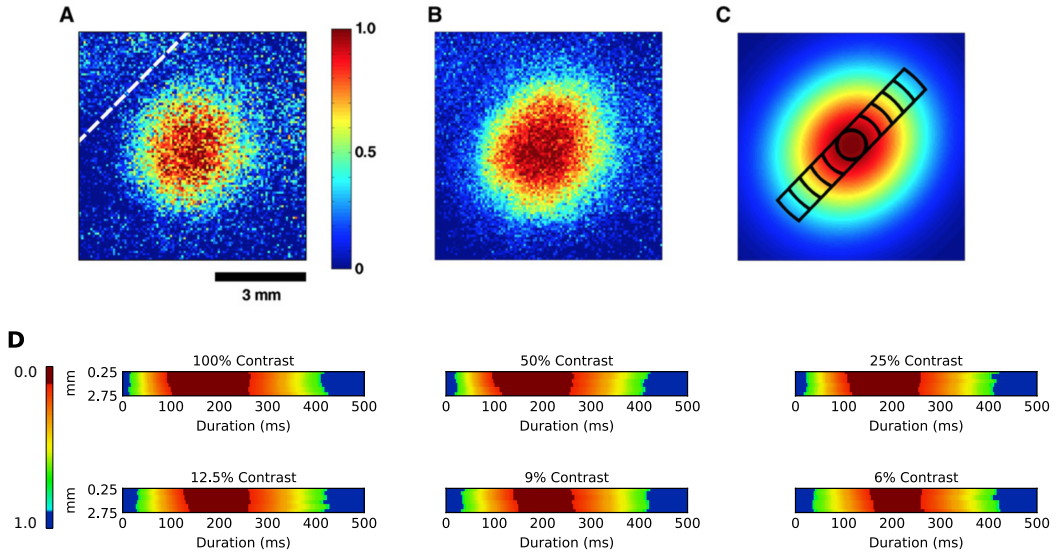


Figure 2.6: **Dynamics of the VSDI response in macaque V1 observed in Sit et al. (2009)**. Response evoked by a $0.167^{\circ}$ Gabor stimulus flashed on from time 0 to 200 ms. (A) Example peak response for stimulus at $6\%$ contrast. Color bar indicates normalized response and scale bar marks a distance of 3mm across the cortex. (B) Example response at $100\%$ contrast. (C) Layout of spatial regions of interest (ROIs) shown on top of a Gaussian fit of the response in (B). The strip is 1mm wide and each circular annulus has a width of 0.5 mm. (D) Plots of the spatiotemporal response averaged in the ROIs shown in (C). Each plot is the spatiotemporal response at the indicated stimulus contrast with each plot composed of a horizontal strip per ROI at the indicated position. Each strip is a 500 millisecond long fit of the mean response in the ROI to a logistic function based on the response at $10\%$, $50\%$, and $90\%$ of the maximum. Each row in such a strip is normalized independently, and then displayed according to the color bar on the left.

The spatiotemporal plots shown in Figure 2.6D are a useful way to summarize the response dynamics over time and space and will be used extensively in Chapter 4. For reasons that will be discussed later, only the onset portion of the responses up to 200 ms will be examined. There are several features of interest in the onset response: (1) the initial onsets are nearly vertical in the spatiotemporal plots, suggesting the initial response appears at roughly the same time across cortical space, (2) towards the peak of the response, a spatiotemporal gradient develops with more distal regions responding more slowly, and (3) as overall stimulus contrast decreases, the overall latency of the responses increases.

All these properties are inferred from the six plots shown in Figure 2.6. It is important to understand that these plots could potentially be misleading as they do *not* show the experimental data directly. Instead, what is actually shown in each row is a piecewise-fit to a rising and falling logistic function. These logistics are constrained using the response at 10%, 50%, and 90% of the maximum amplitude.

This means the responses are always going to be smooth within each ROI as the logistic function that is representing the data is smooth. If the fitting error is high, it is conceivable that the true VSD signal has a very different profile that is obscured by this fitting procedure. For this reason, we need to verify that the raw VSD signal does rise slowly and plateau, in order to ensure the logistic fit is not invalid.

**Reynaud et al. (2012)** It is worth considering a second source of experimental data to ensure that the plots in Figure 2.6 are a suitable summary of the VSD response after the application of the logistic fits. This is achieved by Figure 2.7 which shows the raw $\Delta F/F$ responses from Reynaud et al. (2012) visualized using the same format of spatiotemporal plot.

Each such plot shows the response averaged in the three ROIs shown in Figure 2.7A using the same color map as Figure 2.6D. Although the shape and position of the ROIs are not the same as those used in Sit et al. (2009), the general properties of the raw VSD signal across space and time remain consistent.

Generalizing the experimental results from both studies, it is clear that the dynamics of the VSDI response is qualitatively very different from the PSTHs of the spiking response shown in Figure 2.5. In the 200 millisecond period in which the Gabor is presented, the VSD signal slowly reaches a plateau, whereas the PSTHs expressed by the IRD model are long past their peak. Because of this large discrepancy between PSTH and VSDI profiles, it has been difficult to interpret findings from VSDI, and so it is

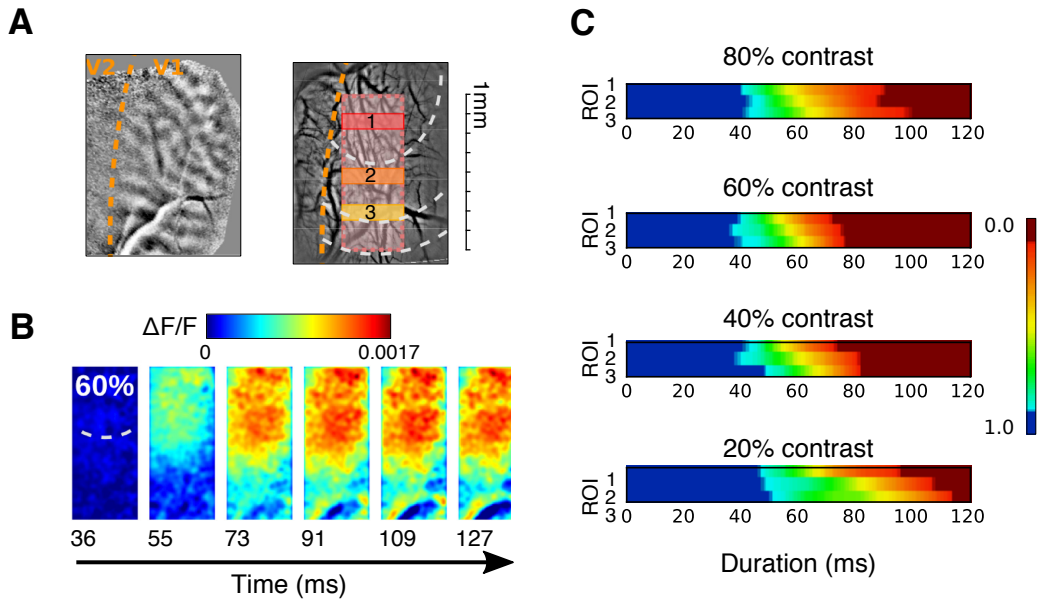


Figure 2.7: **Dynamics of the VSDI response in macaque V1 observed in Reynaud et al. (2012)**. (A) Left, ocular dominance map used to delimit the boundary between V1 and V2. Right, image of the cortical vasculature annotated with the three labeled regions of interest used in analysis. White dotted arcs indicate retinotopic representations of the stimulus for the center (2° diameter) to the periphery (dotted arc-circles at 1° and 1.8° eccentricity to the center outer border). (B) Example of the $\Delta F/F$ VSDI response over time for the $60\%$ contrast stimulus. Stimulus consists of a drifting sinusoidal gratings presented behind a circular window (2° diameter) (C) Spatiotemporal plots of the $\Delta F/F$ response for four different stimulus contrasts plotted in the format used by Sit et al. (2009).

important to resolve these conflicting views of temporal evoked responses. Achieving such a resolution is the topic of Chapter 4.

## 2.4 Development of feature selectivity

In the experiments described in the previous section, oriented stimuli were used to evoke a neural response, because the neurons of the primary visual cortex are driven strongly by oriented visual stimuli. In adult macaque V1, strong orientation selectivity is expected assuming the animal has reached visual maturity. In order to account how neurons become feature selective in the first place, it is necessary to understand how the cortical response changes over the course of visual development.

Observing the neural response over development requires difficult and time-consuming chronic experiments. First, the animals need to undergo experimental preparation involving surgery. After surgery, every precaution must be taken to ensure the animal makes a full recovery in order to avoid later complications. The experimenter has to be constantly available for several weeks, in order to perform regular recordings and to ensure that both the animal and the experimental preparation is kept stable. Single-unit studies are particularly difficult, as keeping track of an individual neuron over many weeks with an electrode preparation is nearly impossible.

These practical difficulties make chronic studies a risky proposition for an experimenter, involving a high investment of time and laboratory resources for uncertain gains. When acute studies can yield useful scientific results far more quickly, it is not surprising to discover that there are far fewer chronic studies published in the literature. Specifically, there is essentially no published data that tracks how individual V1 neurons respond over development, in a mammalian species with smooth maps. Two-photon imaging techniques may soon make this sort of recording possible, but so far none have been published.

### 2.4.1 Robust and stable orientation map development

Luckily, at the population level, chronic optical imaging data of orientation map formation is available, though only for ferret V1 (Chapman et al., 1996). The properties of orientation maps as they emerge in a maturing animal are an important constraint on any developmental model simulating a large region of V1, and so this data is a crucial point of reference despite potential species differences. This study of eight ferret

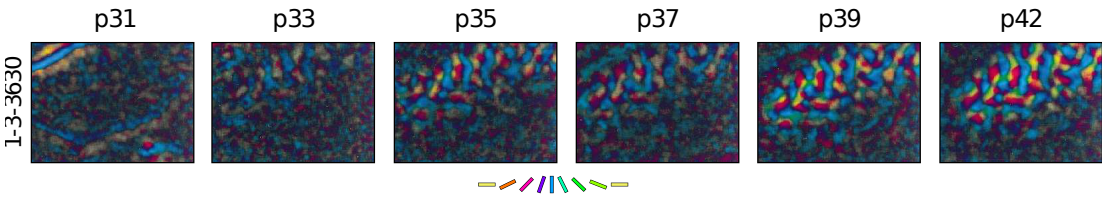


Figure 2.8: **Stable development of orientation maps in ferret striate cortex**. Maps shown were recorded from a single animal, reproduced from Chapman et al. (1996). Each pixel shows the orientation preference according to the key above and pixel value indicates the orientation selectivity, with black indicating relatively unselective regions. Each map has the selectivity scale normalized independently, revealing blood vessels at P31 even though these features have no significant orientation selectivity. As the map matures, the stability of the spatial structure of the map is apparent, as selectivity increases but the overall map pattern stays constant.

pups also gives some insight into how the average orientation selectivity changes over development, as (non-chronic) electrode recordings were carried out over similar time periods.

Figure 2.8 shows the development of the orientation map preference and selectivity in a single animal from postnatal day 31 to postnatal day 42. As orientation selectivity increases, the map structure gradually becomes apparent without a significant rearrangement in the spatial arrangement over time from p33 onwards. In other words, the neurons become more selective over development but the orientation map structure itself is stable.

In Chapman et al. (1996), the stability of orientation maps recorded with intrinsic optical imaging is evaluated over development using a map similarity metric. This similarity is calculated as the maximum mean orientation difference between the map measured on a particular postnatal day with the final map recording. Dividing these mean difference values by 90 and subtracting the result from one yields a normalized index that varies between zero and one. A value of zero indicates anticorrelation, a value of a half indicates no correlation, and a value of unity indicates identical maps.

Using this similarity metric as well as a selectivity measure obtained from the optical signal, it is shown that orientation selectivity continues increasing well after the maps have attained a stable spatial organization early on in development. These results are shown in Figure 2.9.

In order to validate the selectivity measure obtained from the optical signal, additional recordings were obtained over development using single-cell electrophysiology.

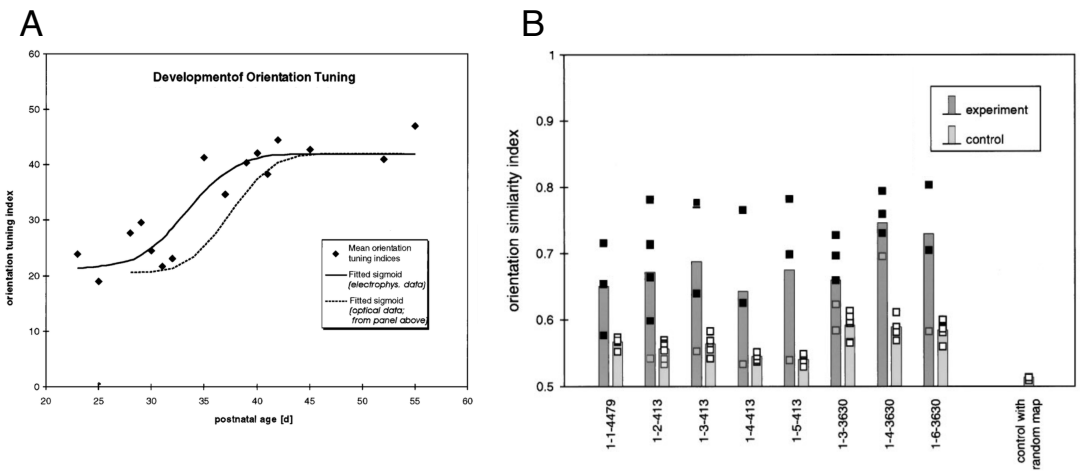


Figure 2.9: **Selectivity and stability of ferret orientation maps over development.** (A) Change in orientation selectivity over development, plotting orientation tuning index against postnatal age in days. Points indicate tuning index values recorded with single-cell electrophysiology and the solid line shows the corresponding sigmoid fit. The dashed lines shows the sigmoid fit based on the optical imaging data. (B) The orientation similarity index of the recorded maps for eight ferrets for experimental (filled squares) and control conditions (open squares). The difference between these conditions is that the controls use between-animal comparisons of similarity whereas the experimental condition shows the similarity of the recorded map with the final map measurement within the same animal. Reproduced from Chapman et al. (1996).

It was found that these more traditional measurements of orientation selectivity correlated well with the orientation tuning measured with optical imaging. As a result, we can be confident that the observed selectivity changes in the orientation maps are also reflected at the level of individual neurons.

Taken together, Figures 2.8 and 2.9 show that map development exhibits *stability* while developing *selectivity*, but there is one other property that will be important in the analysis presented in Chapter 5. Orientation map development is also *robust* against differences in inputs, as similar final map patterns develop until nearly 3 weeks of age in cats, regardless of whether the eyes are open (Crair et al., 1998). Development of selective orientation maps therefore appears to be both robust and stable.

What this section has showed is that there is emergent structure at the population level in terms of how the feature preferences of neurons are organized over cortical space. The smooth organization of orientation maps demonstrates that neurons do not develop their receptive fields independently from one another with local regions of

cortex having neurons with similar preferences. In order to understanding this process, it is necessary to put these experimental results into a theoretical framework.

In the next section, one type of modeling approach that can account for feature map formation over development will be introduced. A particular self-organizing map model will be described that accounts for the stable and robust orientation map development shown in Figure 2.9. This model will be the basis of the work done in Chapters 5 and 6 of this thesis.

## 2.5 Self-organizing network models

Self-organizing map models attempt to explain the emergence of feature selectivity and smooth feature maps through the repeated operation of simple, preferably local learning rules for modifying synaptic strengths and other properties of simulated neurons. The models are driven by afferent input from outside the cortical region, and typically achieve organization through "lateral" interactions between neurons in a region, across the cortical surface.

The fundamental behavior of self-organizing network models, as introduced by von der Malsburg (1973), is to establish a topological mapping from a high dimensional feature space of the visual input onto a two dimensional cortical surface. The ability to preserve continuity of the input space using unsupervised learning is also used as a general data visualization and analysis in the well known Kohonen self-organizing map model (SOM) (Kohonen, 1982).

The core principle used to update the synaptic strengths in these networks is Hebbian learning (Hebb, 1949). In brief, when the pre- and postsynaptic activity is relatively high, the corresponding synaptic weights increase. This most basic rule would result in monotonically increasing synaptic weights, and so there also needs to be a mechanism to decrease synaptic strength. The mathematical formulation of one such rule, divisive postsynaptic weight normalization, will be described shortly for a particular self-organizing map model.

When trained with visual patterns, self-organizing model units can develop orientation selectivity and self-organize spatially in a way that appears very similar to the smooth maps that are experimentally observed. This observation has inspired a number of different developmental models of feature map formation based on the principles of self-organizing maps (Obermayer et al., 1990; Miikkulainen et al., 2005).

All models related to the SOM rely on lateral interactions to preserve the topo-

logical representation of the input feature space when the model weights are updated. Depending on the training stimuli and model architecture, other maps such as direction, disparity, and spatial frequency maps can be simulated and even combined within a single model (Miikkulainen et al., 2005; Bednar, 2012). One fundamental mathematical principle underpinning these models can be expressed as dimensionality reduction, and some forms of the model have been shown to be equivalent to discretized approximation of the principal surfaces of the input distribution (Ritter et al., 1992).

In the next section, a particular example of a self-organizing map model called GCAL will be defined, as this model is the basis of much of the work in this thesis. One the architecture and core equations of GCAL are defined, it will be used to illustrate the fundamental principles of self-organization and development in this class of model.

### 2.5.1 The GCAL model

The GCAL (gain control, adaptation, and lateral) model is the successor to the LISSOM model (Miikkulainen et al., 2005), replacing it for all current purposes. GCAL was developed originally by Judith Law, Jan Antolik, and James A. Bednar, but it was first published in Stevens et al. (2013b) only after I added the systematic analyses outlined in chapter 5 that demonstrate that it robustly and stably achieves high selectivity and realistic maps.

Compared to LISSOM and other related models, GCAL is simpler, more biologically plausible, and yet more robust in its ability to form high-quality orientation maps. The statistics of GCAL orientation maps are analyzed in Chapter 5 to quantify their match to biological results across training conditions, and this model is the basis of the new developmental models presented in Chapter 6.

Like LISSOM, the GCAL model is driven by input patterns that correspond to visual images, patterns of spontaneous activity, or measurement conditions during model training and measurement. Each input pattern is represented as a two dimensional array of activations at the photoreceptor sheet, which then drives the activity of neural sheets later in the visual pathway. Each sheet is a two-dimensional array of computational units where each unit has a scalar activity value. Sheets are connected together by projections where a projection between two sheets specifies the weight contributed by the input sheet, for every unit of the output sheet.

During training, an incremental Hebbian learning rule is applied to all plastic projections once the network dynamics have settled towards a steady state in the cor-

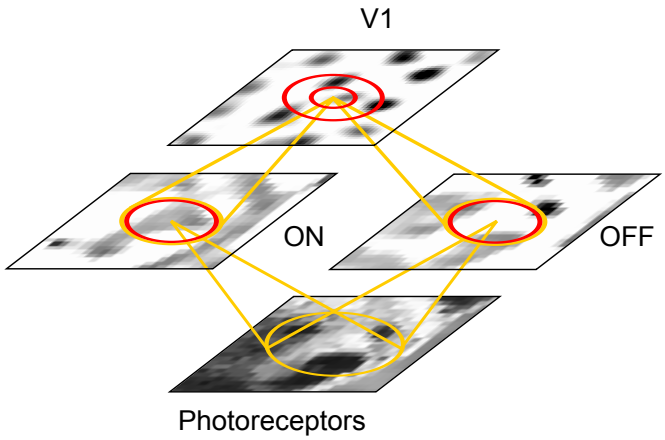


Figure 2.10: **Architecture of the GCAL model.** Connections to the central unit in each sheet are shown with afferent connections in yellow and lateral connections in red. In the cortical sheet, there are two types of lateral connections with different spatial profiles. The short range connections are excitatory and the longer range connections are inhibitory.

tical sheet in response to each individual training pattern presented to the network. The cortical sheet is driven by afferent projections from separate ON/OFF sheets that serve to model the ON and OFF receptive fields of RGC/LGN cells. The architectural schematic of the GCAL model is shown in Figure 2.10.

GCAL is governed by two core equations. The first specifies how the activity is computed at each unit of each sheet. The second specifies the learning step that adjusts the weights of the plastic projections once activity has settled in the network. The rest of the model is specified in terms of the initial weight patterns, the statistics of the training inputs, and the transfer functions that map the input activity per unit into an output activity (which corresponds to the spike rate).

### Activity integration per unit

The activity at each unit at a given simulation timestep is found by summing the activity contribution $C_{j,p}$ of projection $p$ over all the projections feeding into the unit $j$. Each contribution is computed by treating the incoming weight as a vector and taking the dot product of this weight vector $\omega$ with the corresponding activations $\eta_{i,p}$ from the source sheet in the connection field $F_{j,p}$.

$$C_{j,p}(t+\delta t) = \sum_{i \in F_{j,p}} \eta_{i,p}(t)\omega_{ij,p} \tag{2.4}$$

One feature GCAL introduces that is not present in the earlier LISSOM model is a contrast-gain control mechanism in the ON/OFF sheets. This is implemented using lateral divisive inhibitory projections in the ON and OFF sheets. Contrast-gain control is a well established phenomenon (see Carandini and Heeger 2012 for a review) and was found to improve the robustness of map formation in the GCAL model in a way that is rigorously quantified in Chapter 5.

Combining the afferent and contrast-gain control contributions allows the output activity of the ON or OFF sheets $O$ to be described with the following equation:

$$\eta_{j,O}(t+\delta t) = f\left(\frac{\gamma_O \sum_{i\in F_{j,P}} \Psi_i(t)\omega_{ij}}{k+\gamma_S \sum_{i\in F_{j,S}} \eta_{i,O}(t)\omega_{ij,S}}\right) \tag{2.5}$$

The numerator corresponds to the contribution as specified in Equation 2.4 for photoreceptor activities $\Psi_i$ weighed by the overall afferent projection strength $\gamma_O$. The $\gamma_S$ constant is the strength of the inhibitory gain control projection expressed in the denominator. The constant offset $k$ is necessary to ensure activity is defined for weak inputs. $f$ is a half-wave rectifying function.

In the cortical layer, the contributions specified by Equation 2.4 are summed over the ON, OFF, lateral excitatory, and lateral inhibitory projections, where the lateral inhibitory component is negative to model a subtractive effect of relatively long-range inhibition (up to a couple of millimeters of cortical tissue). The total of these contributions is passed through the transfer function specified in Equation 2.9 and the network is allowed to settle for 16 timesteps to allow the network activity to settle, before the Hebbian learning rule given in Equation 2.10 is applied.

#### Weight profiles of the initial condition

The weight profiles for the afferent and lateral projections in the ON/OFF sheets are static. The weight $\omega_{ij}$ from photoreceptor $i$ to the ON or OFF unit $j$ is defined by a standard difference-of- Gaussians profile to model LGN receptive fields:

$$\omega_{ij} = \frac{1}{Z_C}\exp\left(\frac{x^2+y^2}{2\sigma_C^2}\right) - \frac{1}{Z_S}\exp\left(\frac{x^2+y^2}{2\sigma_S^2}\right) \tag{2.6}$$

Here the $x$ and $y$ values are the spatial position of the unit relative to the central location $(0,0)$, the central $C$ and surround $S$ Gaussian sizes are specified by $\sigma_C$ and $\sigma_S$ respectively and $Z_C$ and $Z_S$ are normalization constants that ensure each term sums to 1.0. The OFF weights are simply the negation of the ON weights given in Equation 2.6. The size of the center and surround is given by $\sigma_C = 0.037$ and $\sigma_S = 0.15$ respectively.

The static weights of the lateral inhibitory gain control projection in the ON/OFF sheets are specified using a straightforward Gaussian profile where the equation is generalized for any projection $p$:

$$\omega_{ij,S} = \frac{1}{Z_p} \exp\left(-\frac{x^2+y^2}{2\sigma_p^2}\right) \tag{2.7}$$

The $x$ and $y$ values specify the location of the presynaptic neuron and $Z_P$ is the normalizing factor that ensures that the total lateral inhibitory weight sums to 1.0 for each unit. The size of this spatial normalization pooling $S$ is given by the above equation where $\sigma_S = 0.125$ and $p = S$.

The weight of the lateral excitatory connection $E$ is also described by Equation 2.7 using $\sigma_E = 0.27$ and $p = E$. The weights for the afferent weights of the V1 sheet and the lateral inhibitory projection are not pure Gaussians but consist of samples from a uniform random distribution multiplied by a Gaussian envelope described by Equation 2.7. For the afferent weights (A) the envelope has $\sigma_A = 0.27$ and for the lateral inhibitory projection (I) the weight envelope has $\sigma_I = 0.075$. Note that for the purposes of computing the normalization factor $Z_p$, the weights for each projection are defined within a circle of radii $r_A = 0.27, r_E = 0.1$, and $r_I = 0.23$.

#### Homeostatic threshold adaptation

The transfer function used in the ON/OFF sheets has already been described as simple half-wave rectification ($f$ in Equation 2.5). Computing the final activity of the cortical sheet is more complicated, because an adaptive per-unit threshold is employed in GCAL. This transfer function corresponds to a homeostatic threshold adaptation mechanism that is found to improve the map quality of the GCAL model in a way that is rigorously quantified in Chapter 5.

This adaptive threshold adjusts in response to the smoothed exponential average of the settled activity $\bar{\eta}_j$ with smoothing parameter $\beta$:

$$\bar{\eta}_j(t) = (1-\beta)\eta_j(t) + \beta\bar{\eta}_j(t-1) \tag{2.8}$$

This exponential average is then used to compute the threshold $\theta$ at time $t$ using homeostatic learning rate $\lambda$ and target average $\mu$:

$$\theta(t) = \theta(t-1) + \lambda(\bar{\eta}_j(t) - \mu) \tag{2.9}$$

The value of smoothing parameter in GCAL is given by $\beta = 0.991$, the learning rate is given by $\lambda = 0.01$, and the average target activity per unit is specified by $\mu = 0.024$, which also sets the initial value of $\bar{\eta}_{jA}(0)$.

#### Normalized Hebbian learning

Now that the means to compute all the activities of the units in the network has been defined, the last step is to specify the learning rule used to update the projection weights. On the first iteration of the model, the weights have the initial distributions specified in the equations above. Then with a timestep of $t = 0.05$, activity propagates through the network with connection delays that also have a value of $t = 0.05$. Activity from the ON/OFF sheets are only propagated after contrast-gain control has been applied and then 16 simulation steps are executed to allow the activity to settle in the cortical layer. At this point, normalized Hebbian learning is applied before all activity is cleared and the network is presented with the next training stimulus.

In GCAL, only the afferent weights feeding into the cortical layer and the long-range lateral inhibitory weights are plastic. The weights to the ON/OFF sheets are static, to focus on development at the cortical level. The short-range lateral weights in the cortical sheet are also static, as plasticity in these lateral excitatory connections has been found to have little effect.

Once the activity has settled at the end of a training presentation, all the plastic connections are adjusted using a simple, normalized model of activity-dependent plasticity:

$$\omega_{ij,p}(t) = \frac{\omega_{ij,p}(t-1) + \alpha\eta_j\eta_i}{\sum_k(\omega_{kj,p}(t-1) + \alpha\eta_j\eta_k)} \tag{2.10}$$

This equation expresses Hebbian learning that specifies how a synaptic weight changes for a given learning rate $\alpha$ and pre- and postsynaptic activities of $\eta_i$ and $\eta_j$ respectively. The denominator applies divisive postsynaptic weight normalization to prevent runaway increases in strength, which models another form of homeostatic adaptation.

### 2.5.2 Principles of self-organization

Now that the GCAL model has been formally defined, it can be used to illustrate the principles of developmental self-organization in this class of model. This will make sense of the model structure and help develop an intuition for the equations involved. When it comes to understand self-organization, it is Equation 2.7 that is critical as it defines the shape of the lateral weight profiles in the cortical sheet. Lateral interactions

are what drive the self-organization process, not only in GCAL but in earlier map models, such as the more abstract Kohonen self-organizing map model.

In the Kohonen self-organizing map model, the lateral interactions necessary for smooth map formation are computed by locating the maximally responding unit in the cortical sheet and imposing a spatial Gaussian kernel at that location in the learning step (Kohonen, 1982). Such an operation requires a biologically implausible global supervisor, a requirement that is removed in later models such as the LISSOM (Laterally Interconnected Synergetically Self-Organizing Map) model (Sirosh and Miikkulainen, 1994; Miikkulainen et al., 2005). The spatial profile of Hebbian learning in the cortical sheet is fundamental to the self-organization process, and in LISSOM, the necessary lateral dynamics are modeled with explicit excitatory and inhibitory lateral connectivity.

The GCAL model has the same effective Mexican-hat inhibition profile introduced by LISSOM, shown in Figure 2.11 that drives smooth map formation and is implemented with short-range excitatory connectivity and long-range inhibitory connectivity. This organization specifies the *effective* lateral dynamics as a function of distance across the cortex, even though cortical anatomy suggests the inverse arrangement. What is important about Equation 2.7 is that the spatial extents of the excitatory lateral connections are smaller than those of the inhibitory connections.

Related but more detailed self-organizing map models such as LESI (Law, 2009) and LESPI (Rudiger, 2016) show how it is possible to achieve the necessary effective Mexican-hat profile while retaining anatomically plausible connectivity of relatively short-range inhibition and longer range excitation. Whether such an account is actually necessary to explain the dynamics of self-organization is debatable as large, inhibitory basket cells exist in the cortex that span at least one orientation hypercolumn (Buzás et al., 2001). As a result, the anatomical basis for Mexican-hat interactions are not entirely clear (Kang et al., 2003) and for this thesis we will retain the Mexican-hat organization for simplicity and ease of implementation.

Why is this Mexican-hat profile in the lateral interaction necessary for these self-organizing map models? The answer is related to the settling steps and how the network activity evolves towards a steady state at which point the learning is applied. In particular, the Mexican-hat profile drives the formation of activity "bubbles", smooth localized areas where lateral excitation helps sustain activity. The emergence of activity bubbles in the GCAL model in response to training patterns at the start and end of development is shown in Figure 2.12.

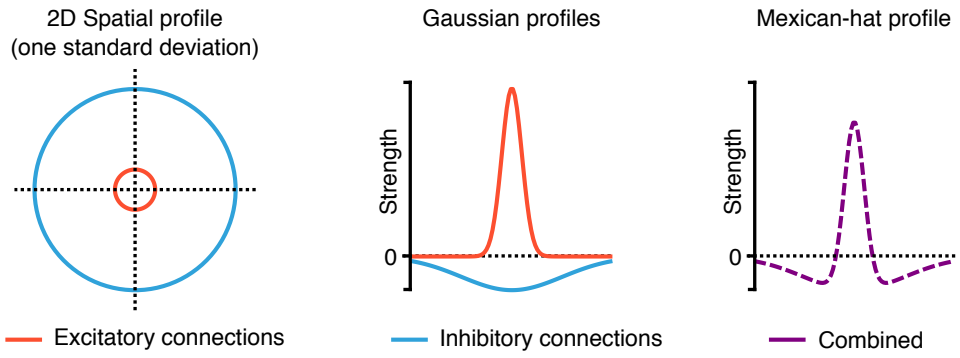


Figure 2.11: **Schematic of Mexican-hat inhibition profile that drives activity bubble formation necessary for smooth map organization, as implemented by two types of connection.** (A) Schematic view of the cortical surface with one unit (center of the cross) making both short range excitatory lateral connections (red) and longer range inhibitory connection (blue). (B) Cross section showing the interaction strengths sliced across one spatial dimension. The excitatory connections have a depolarizing effect (positive) and the inhibitory connections have a polarizing effect (negative) on neighboring units. (C) The combined effect of these two profiles is a "Mexican hat" profile which excites proximate units and inhibits units further away. This is an effective profile that is an abstraction of the neuroanatomy but is a necessary feature of self-organizing map models in order to drive activity bubble formation.

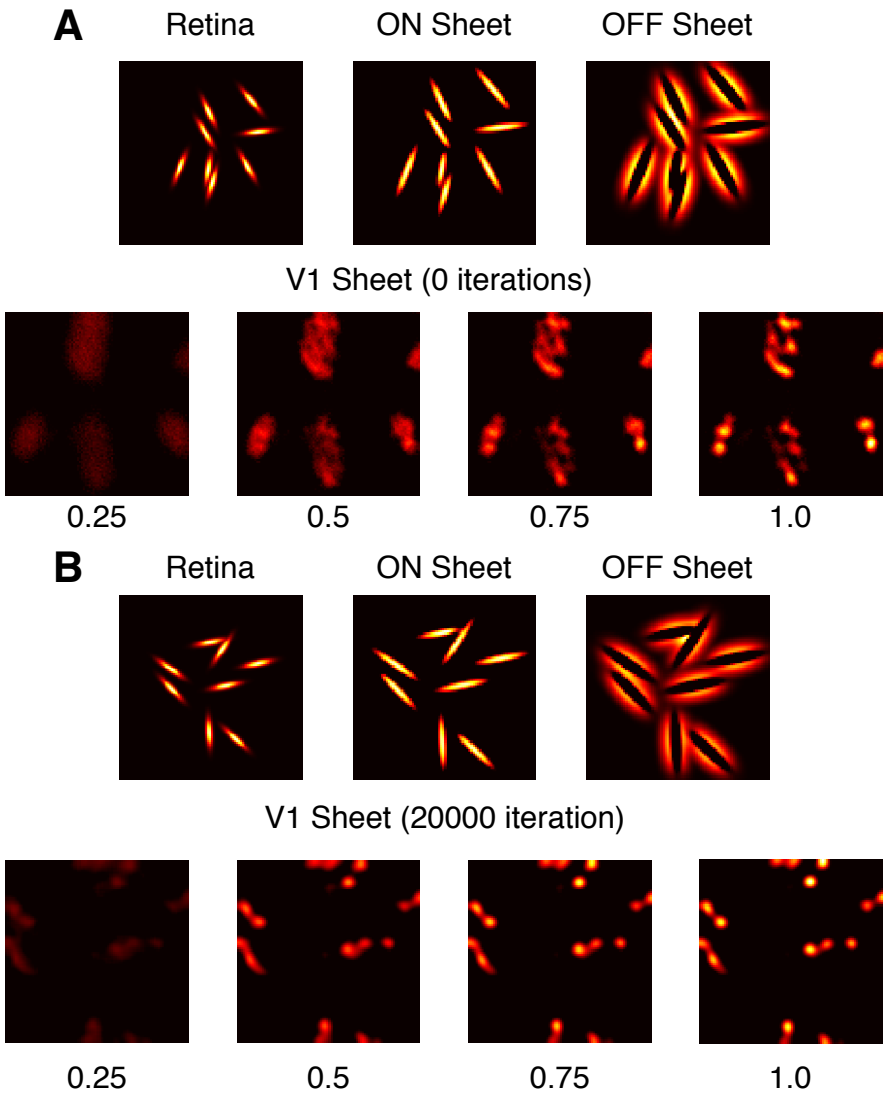


Figure 2.12: **Activity bubbles in response to training patterns in the GCAL model at the start and end of development (20000 iterations)** (A) Activity in the photoreceptor, ON, OFF, and cortical sheets in response to an example training pattern, at the start of development after the model is initialized. The process of settling is shown in the V1 sheet, eventually resulting in the formation of activity bubbles. Activities shown at $0.25$, $0.5$, $0.75$, and $1.0$ units of simulation time after the training pattern appeared on the photoreceptor sheet. Training patterns are updated every simulation time unit. (B) Same plots for the photoreceptor, ON, OFF, and cortical sheets after $20000$ iterations of development. Bubble formation in the network is now much quicker and their positions are determined by the tuning properties of the units.

Activity bubbles tend to center around the units with the strongest afferent drive, as the long-range inhibition of the Mexican-hat profile ensures competitive winner-takes-all interactions between bubbles. Each bubble helps lead to smooth feature maps, in the same way that the artificially imposed Gaussian activity in the Kohonen SOM helps maintain the topological representation of the input. In general, smooth spatial regions of postsynaptic activity make sure that units that are spatially close in the cortical sheet will have similar feature preferences, due to the action of Hebbian learning. Mathematically, this process can be understood as a gradual rotation of the weight vectors (the unit weights as a point in a high dimensional space) of the locally activated units towards a particular direction in feature space driven by each training stimulus.

The process of activity bubble formation varies over development as the weights change. This can be seen by comparing Figures 2.12A and B, showing the process before an after development. Due to the random initial state, also seen at iteration zero of figure 2.13B, bubble are initially diffuse and take longer to form. Once tuning develops, a selective V1 unit is able to quickly boost local activity via the excitatory component of the Mexican-hat profile and suppress units that are further away and competing with it. As a result, bubbles form quickly and reliably in specific locations, as seen in 2.12B.

The effect of activity bubbles and the Mexican-hat profile can be seen in Figure 2.13 which shows development of orientation maps and ON afferent weights in the GCAL model. The ON weights show that the model has self-organized and that units are now selective to oriented stimuli. Note that orientation maps appear early on and are stable over time. This is quantified in the analysis presented in Chapter 5.

## 2.6 Conclusion

The visual system is composed of a large, diverse population of cells with structure on both small and large spatial scales. In addition, there are different types of dynamics on different timescales such as the neural activity evoked by a particular visual stimulus or the gradual developmental processes that shape the functional organization of the cortex over time.

These different processes are recorded with different techniques such as electrode recordings and voltage-sensitive–dye to record evoked single-unit and population responses on short timescales and chronic optical imaging of the intrinsic signal on long timescales. All of these techniques capture different aspects of cortical function and

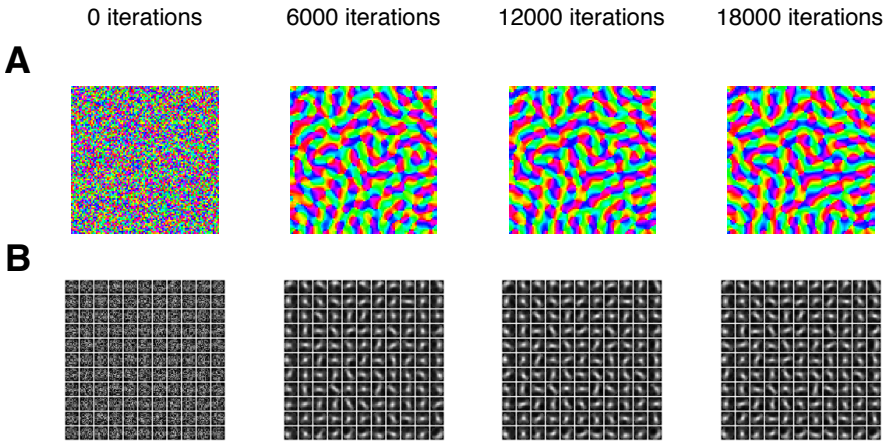


Figure 2.13: **Development of orientation maps and ON afferent weights from model initialization to $18000$ training iterations** (A) Orientation map measurements using the protocol from Stevens et al. (2013b) after $0$, $6000$, $12000$, and $18000$ training iterations. (B) Afferent weights from the ON sheet for $11 \times 11$ regularly sampled units across the cortical sheet. Weights are initialized according to a uniform random distribution multiplied by the Gaussian envelope defined by Equation 2.7. These weights then self-organize during the training process and the elongated shapes of these weight fields after development are indicators of the orientation preference and selectivity of the unit.

ideally all these techniques could be applied at once to record data over the lifetime of a single animal.

It is possible to imagine a hypothetical experimental technique that can chronically record the activity of millions of neurons with high spatial and temporal resolution over the lifetime of a macaque animal subject. Such a technique would be an extremely powerful tool for understanding the dynamics of neural activity. In the absence of such a technique, what is required is a theoretical framework that allows these different sources of data to be integrated together, in order to constrain a computational model of cortical operation.

# Chapter 3

# A reproducible workflow for exploratory research

Understanding visual processing in the primary visual cortex has been the topic of intense research over the past fifty years, resulting in an extensive literature spanning a wide array of different experimental and computational techniques. Many different types of model have been used to investigate the properties of visual processing, ranging from theoretical models that tackle the propagation of visual information in an abstract way, to mechanistic models that attempt to relate visual processing to the neurons themselves.

As mechanistic models expand to account for more of the observed properties of the visual system, they begin to pose some of the same challenges to the computational modeler that the real biological system poses to the experimenter. There is an overwhelming volume of raw data that can be acquired from the subject of study, whether real or simulated, and there is a myriad of ways in which this data can be analyzed and visualized. As the number of model parameters and different analyses grow, it becomes increasingly difficult to communicate results clearly to the scientific community, threatening the integrity of the scientific process.

All researchers need to be able to communicate their findings in order to engage in the peer review process so that future researchers can build on their work with confidence. It is crucial that researchers can disseminate their findings to the rest of the scientific community in way that is clear and understandable so that other researchers can examine the core scientific claims. This is the basis of scientific trust and as the number of steps involved in a research project increase, the issue of scientific communication and reproducibility becomes a primary concern.

Improving reproducibility in modern science is a problem that cuts across disciplines. Many different areas of research now rely heavily on computational resources for collecting, processing, and analyzing large volumes of data. In other words, the challenge of achieving clear, reproducible scientific communication is general and needs to be tackled by the research community as a whole. One approach is to build standardized tools to make the research process more efficient while capturing the steps necessary to achieve reproducibility. The steps that are captured then need to be communicated in a way that is clear and meaningful.

The need for an efficient, exploratory, and reproducible workflow was recognized early on in this project. Two concrete ways of improving research efficiency and reproducibility were identified, resulting in the creation of two new libraries for Python. Both the Lancet and HoloViews projects are general purpose and freely available under an open source license. This chapter describes how these two tools were critical in ensuring that the simulations, calibration, and analysis presented in this thesis were carried out in a maintainable, communicable and reproducible way.

## 3.1 Reproducibility of computational models

Practical laboratory experiments are time-consuming and any procedure involving animal subjects, especially primates, are extremely costly. It is easy to understand why the process of reproducing a practical experiment requires a significant investment of resources. Even when replication studies are attempted, ambiguous results may persist due to the effects of small sample sizes, uncontrolled idiosyncrasies between experimental subjects, or miscommunication in the required steps.

The last point is particularly relevant to computational neuroscience and is often the reason why modeling results are difficult to reproduce, even though researchers working *in silico* face fewer practical issues. In theory, a computational model is specified precisely by code that can execute reliably on any suitable processor.

Occasionally, there are practical concerns to consider. For instance, there may be differences when performing numerical integration on 32 bit or 64 bit hardware or there may be difficulties capturing the necessary software state when trying to generate reproducible, high quality random number streams. When these issues arise, they need to be carefully addressed, but it is typically safe to assume that a working program is an ideal, executable specification of some model.

If a program is an exact specification, why is there any issues of reproducibility?

The key problem is that an executable binary is a specification aimed at the level of computational hardware, and that the program source code only has to satisfy the constraints imposed by the language compiler or interpreter. There are no guarantees that this executable code can also be understood by a human, that the program correctly implements the intended scientific specification, or that the computation performed is scientifically meaningful.

This issue cuts to the heart of the distinction between reproducibility and replicability, that is to say an external researcher's ability to reproduce results exactly, as will be discussed shortly. A program binary may be sufficient to achieve *replicability*, but unless a researcher can understand and manipulate this program at the scientific level, this does not correspond to *reproducibility*. It is when considering the issue of reproducibility that proper scientific communication is essential, as described in the next section.

### 3.1.1 Reproducibility versus replicability

So far the discussion has been framed using the word "reproducibility" but now that the concept known as "replicability" has been introduced, it is necessary to define these two terms. In practice, these concepts correspond to two different points along a continuous spectrum and both properties are essential for the integrity of the scientific process.

Replicability is the notion that the results of an experiment should remain exactly the same if repeated exactly. This means that the hallmark of a replicable experiment is that other researchers are able to repeat the claimed results by following the exact same set of steps. In a computational context this might mean that a simulation can be re-run given a particular binary file or given the appropriate source code in an appropriately defined software environment. A simulation or model may be considered replicable if sufficient information has been made available to allow anyone to regenerate the relevant results.

Reproducibility is a broader concept with a greater overall value to the scientific process. When a model is reproducible, the particular code artifacts, software versions, random seeds, and other details of implementation are not important. Instead, the goal is to establish whether or not the scientific claims associated with the model follow from the minimal definition. The aim is to explore the validity of the scientific concepts underpinning the claims and to test the behavior given different inputs in a way that is

independent of implementation details. As such, reproducibility is a better measure of scientific correctness than replicability.

The merits of achieving replicability are currently under debate. Some argue that replicability is an essential first step towards reproducibility Gent (2013) and others argue that it is "not worth having" due to lack of scientific value (Drummond, 2009). Still, if there were no cost associated with attaining replicability or reproducibility, it is clear that scientific research should ideally satisfy both properties.

In this thesis, replicability is viewed as an important initial step towards the true goal, reproducibility. When one research group attempts to build on work originally carried out elsewhere, achieving an *exact* match with the published values is crucial for establishing confidence. Only by achieving an exact match with published results is there any assurance that the code and software environment has been configured correctly.

Having defined replicability and reproducibility, we can attempt to find a workflow that makes it easier to achieve these two properties while also improving research productivity. In particular, we will now consider how the quality of a scientific output may be improved by exploring how program correctness relates to the scientific tradition of maintaining a research logbook.

### 3.1.2 Logbooks and literate programming

Before the advent of the computer, a scientist would often own a handwritten logbook in order to track the steps taken over the course of a research project. The importance of not leaving technical details to memory has long been recognized; a concrete record is essential if many highly technical steps is to be replicated at a later time.

Nowadays, not only is research recorded and disseminated electronically, significant simulation and analysis code is a core component of many research disciplines. Code itself is composed of a huge number of discrete instructions specifying the computation of interest. Viewed in this way, code is simply another set of research steps that need to be recorded, making them a natural candidate for inclusion in a modern electronic logbook format.

In many domains, code dominates the process of scientific analysis, making it possible to invert this perspective and consider code as the primary artifact. The question becomes one of correctness; does the code satisfy the associated specification? A program that fails to perform its desired function is incorrect as it fails to satisfy the

corresponding scientific intent. In other words, the goal of reproducibility is related to the goal of helping scientists write high quality code to assist their research, as opposed to low quality, incorrect code which if often detrimental. This is why making the code needed to generate a publication freely available for review is important for establishing confidence in the published results.

One approach known as "literate programming" introduced by Knuth (1984) aims to improve program correctness. The idea is to interleave natural language text with the code so that a programmer can explicitly state the thoughts behind the program, making errors easier to identify. The text surrounding sections of code is written in natural language to help the author clarify their thoughts and to provide useful, high-level documentation to other people. With a literate programming approach, it is often easier to maintain the high-level specification in mind, making it easier to establish whether or not any particular block of code fulfills its intended purpose. In scientific research, this makes it easier to verify that the code is correctly implementing its scientific intent.

This highlights the advantages of interleaving code with text, which may include additional figures and mathematical equations, from two different perspectives. From a code perspective, code is likely to be of higher quality and more likely to fulfill its specification if it is kept together with the corresponding high-level context. From the perspective of reproducibility, such a document captures the code necessary to execute the relevant research steps. We now examine the strengths and weaknesses of a particular literate programming format, namely the Jupyter Notebook.

### 3.1.3 The Jupyter Notebook

The ability to interleave code, text, and media content within a notebook format has existed over a decade in the form of various proprietary solutions, such as the Mathematica Notebook (Wolfram, 2003). These attempts have suffered from restricted interoperability and a lack of open standardization, limiting their applicability and adoption. In more recent years, the Jupyter Notebook (jupyter.org) has emerged as a popular, open source notebook format that originated with the IPython project (Pérez and Granger, 2007). It has since diversified in the set of languages it supports and the name "Jupyter" is designed to reflect the three core languages that can be used in the notebook environment, namely, Julia, Python, and R.

The Jupyter Notebook interface is hosted in a web browser, which has recently become feasible as a exploratory environment due to the widespread adoption of the

HTML5 standard. The websockets protocol introduced in HTML5 has made it possible to build rich, interactive browser documents that communicate in a bi-directional manner with a local server. The notebook format itself is based on the standard JSON format and can contain any of the common media formats that can be displayed as part of a HTML web page.

Using well-established standards within a ubiquitous and universal piece of software (the web browser) has been a major factor in the success of the Jupyter notebook. Each notebook can contain multiple sections of code interleaved with formatted text defined using the Markdown markup language, fully rendered mathematical equations, and raster and vector images, as well as client-side scripting using JavaScript. This ability to interleave different standardized formats within a single document is one of the biggest strengths of the notebook format.

A live notebook session is simply a different way of working with an interactive interpreter. The user is free to run any code, load data from a file, and manipulate data structures in memory in an exploratory way. By default, an ad hoc, interpreted session is not a replicable artifact unless the history of commands is also recorded. Yet a complete log of all commands ever executed is usually too verbose for a human researcher to follow and understand. The key difference between a notebook and an automated command logging system is that users can curate and refine the exact portions of code that need to be recorded, while discarding everything else. The history stored in the notebook can be made much clearer and much more meaningful than the full history of every command that was executed.

Notebooks allow this type of free exploration but they also make it easy to write and order blocks of code that can be interleaved with useful explanatory material in order to build up an executable document. The idea is to build a notebook that, even after the state of the memory is reset, is both readable and when executed from top to bottom, regenerates all the desired visualizations, analysis, and data structures.

To achieve replicability, one can then use version control to track the contents of the notebook and its dependencies. In order to achieve reproducibility as well, the notebook also needs to express a clear scientific intent and allow users to modify parameters and explore the corresponding results.

Unlike the proprietary alternatives, the open standards leveraged by Jupyter Notebooks helps ensure that this approach is robust and does not rely on continued support of any particular company. It is important that reproducible research is open, not only so that anyone is free to test the core scientific claims but to allow anyone to maintain

the research artifacts over time without restriction.

#### Weaknesses in the ecosystem

The Jupyter Notebook began as part of the IPython project and the Python programming language remains core to the project, although other languages can now be executed in the environment. Python has a strong ecosystem for scientific computing that can be accessed from the notebook, but the core, most widely used scientific Python libraries were designed before notebooks became popular. This has left some particular weaknesses in the value of Jupyter notebooks as reproducible, literate documents.

One gap in the research process is in the ability to execute a large numbers of independent processes. Such batch jobs are very common in science; they may be used to split a large problem into many independent pieces or to carry out parameter searches for optimization. Although the IPython notebook does offer facilities for running shell commands, process management in this way is difficult in the notebook environment.

As a result, it is common for researchers to switch to an entirely different set of tools, such as shell scripts, when faced with the task of launching and managing multiple program runs. Shell scripts are often verbose and are rarely able to express the scientific intent of the researcher in a clean, readable way. Exiting the notebook environment to run batch jobs in this way leaves crucial research steps untracked, reducing the replicability and reproducibility of the notebook document.

A second major weakness is in how data is presented and visualized within the notebook environment. The matplotlib library (Hunter, 2007) is both mature and popular among Python users and can be used inside notebooks. Unfortunately, this often leads to long stretches of imperative plotting code which can quickly dominate the notebook's content and becomes increasingly fragile as it expands. The verbose and fragile nature of such code has a negative impact on legibility, making it difficult to tell a clear scientific story.

Matplotlib was not designed with the possibilities of the notebook environment in mind. The plotting API is not focused on generating animated output, which was only made possible recently, and this library is not capable of generating the sort of interactive web visualizations that are possible using JavaScript. Without a visualization tool built with the notebook environment in mind, notebooks are less readable and less self-contained as long notebooks often break apart into small pieces. This reduces efficiency, as well as scientific replicability and reproducibility.

These problems are not inherent weaknesses in the notebook format, as these issues can be addressed and improved upon by third party libraries. In particular, the Lancet and HoloViews projects developed over the course of this thesis have largely succeeded in solving these weaknesses.

## 3.2 Lancet and HoloViews

Both Lancet and HoloViews are designed to improve research productivity and quality when working with Jupyter notebooks by minimizing the code necessary to express a research task. This makes it quicker for the author to carry out their work while also making the notebook easier to understand for the reader. By keeping the focus on the needs of the researcher and away from the code itself, it possible to build notebooks that are shorter, more legible, and more reproducible.

Four general principles that have guided the design of both these projects. These are: (1) to maximize research productivity by making research tasks easy to specify, (2) to maximize the generality of each project, making them as cross-domain as possible, and (3) to use compositional primitives to maximize flexibility and expressiveness, and (4) to use a succinct and declarative syntax, discouraging data mutation in order to make reasoning about these data structures easier.

Maximizing research productivity is an essential goal for any tool hoping to improve the reproducibility in a concrete way. Although the majority of researchers do understand the importance of reproducibility, their primary concern is to maximize their daily research productivity. A tool that achieves perfect reproducibility but is onerous to use or otherwise inefficient will not be chosen by researchers and will be unable to improve scientific reproducibility in practice.

Achieving generality is important for several reasons. Firstly, the problems tackled by each library are not domain specific. Managing batches of processes and visualizing data are common tasks faced by many researchers across disciplines on a daily basis. Secondly, aiming at a broad audience improves the chances that the library will be widely adopted. With more users, there are more opportunities for constructive feedback, more bug reports, and increased odds that additional developers can be recruited to the project. Thirdly, aiming at a broad audience ensures that the relevant terminology and documentation is kept simple, understandable, and jargon free.

Keeping the project general also helps identify the points where domain-specific flexibility is required, which helps maintain the focus of the project on the general

structure of the problem. The goal is to always make it easy for users to easily adapt the code to their domain specific needs. The last reason is that a general research tool can serve the user better as there is often no telling which direction research will go in. A general tool will remain relevant as the domain specific research tools change over the course of a project.

Compositionality is the third core design choice. A well designed compositional system can be flexible, expressive, and succinct, as long as the semantics of the compositional operators are clear. Using a compositional model, it is possible to achieve complex results using simple, easily understood primitives. In addition, compositional expressions can be constructed gradually, across multiple stages in a notebook.

This offers more opportunities for interleaving explanatory material between sections of code, further improving readability. Lastly, the code defining the compositional primitives does not need to be modified by the user and can simply be imported into the notebook environment. This reduces the code required in the notebook itself, keeping the bulk of the code separate, in a well-tested, third-party library. Only the scientific *intent* of the researcher should remain in the notebook document, so that the final document only keeps track of the steps specific to that particular research project.

Lastly, a declarative coding style in encouraged together with a succinct syntax. In this context, this means that the internal state of an object is specified when it is constructed and subsequent modifications to this internal state are discouraged. This is closely related to compositionality, as declarative atomic elements are easier to compose together and reason about. A declarative style makes it possible to understand the state of an object when it is created, without having to worry about code that might mutate this state later on.

### 3.2.1 Lancet: Managing sets of independent runs

Lancet is a tool for specifying, launching, and managing large numbers of independent process that is available publicly available (http://ioam.github.io/lancet) under the BSD 3-clause license. A paper describing Lancet was published in the Frontiers in Neuroinformatics "Python in Neuroscience II" special issue (Stevens et al., 2013a). This paper has also been included in the Appendix. Lancet was used to execute all the simulation batch jobs that are analyzed in this thesis.

The main goal of Lancet is to allow batches of processes to be specified and launched from Python in a succinct, declarative way. This makes it possible to de-

fine a collection of simulations, launch them, and then collect the results back into a Jupyter notebook, with the notebook keeping track of all the steps involved. This avoids common, ad hoc approaches that are not reproducible, such as the use of `bash` scripts to execute jobs on a compute cluster.

The components of Lancet are designed to be as declarative as possible. Each Lancet object is fully specified by the parameters of its constructor and these parameters cannot be changed after the object has been created. In addition, all objects have a complete, compact, and readable textual representation. This representation is complete in the sense that it is executable code that can be evaluated by Python in order to recreate the same object in memory.

The core component types in Lancet are derived from three basic classes: **`Arguments`**, **`Commands`**, and **`Launchers`**. Each type nests into the next, with an argument object nesting into a command which finally nests into a launcher. Arguments are used to specify *what* is to be run, such as a parameter space in a *tool-independent* and *platform-independent* way. Once an argument object is specified to define the batch arguments, it is passed to a command object to specify *how* these arguments are to be executed in a *platform-independent* but *tool-dependent* way.

In practice, the definition of a command involves specifying which software to be executed with the supplied arguments and how. This is also the level where Lancet can be customized to support the execution of new software tools. Lastly the launcher object is supplied with a command containing the arguments in order to specify the computational platform used for execution. A launcher is now a *tool-dependent* and *platform-dependent* specification that can be called without any arguments in order to launch a batch of processes.

The role of each of these component types will now be summarized with reference to how they were used to run the simulations presented later in this thesis. The analyses in Chapter 5 were based on the simulations run for Stevens et al. (2013b), which is an example of a fully reproducible publication. These notebooks illustrate how Lancet is used in practice, although the visualization and analysis portions of these notebooks would have been better served by HoloViews, described in the next section, had it been available at the time. The notebooks necessary to reproduce the paper may be found at the following address:
https://github.com/ioam/topographica/tree/master/models/stevens.jn13

**A**

```
model_constants = Args(retina_density=24.0,
                       lgn_density=24.0,
                       area=1.5,
                       cortex_density= 98)
model_constants
```

```
Args(
   area=1.5,
   cortex_density=98,
   lgn_density=24.0,
   retina_density=24.0
)
```

**B**

```
contrasts = Range("contrast", 0, 100,
                  21, fp_precision=2)
contrasts
```

```
Range(
   key='contrast',
   start_value=0,
   end_value=100,
   steps=21,
   fp_precision=2
)
```

**C**

```
contrasts * model_constants
```

```
Range(
   key='contrast',
   start_value=0,
   end_value=100,
   steps=21,
   fp_precision=2
) * Args(
   area=1.5,
   cortex_density=98,
   lgn_density=24.0,
   retina_density=24.0
)
```

Figure 3.1: **Examples of `Arguments` objects that were used to define the parameter space explored in Chapter 5.** The Python code is shown on the top and the printed representation of the object is shown below. (A) This arguments object defines the simulation parameters that were kept constant by defining an **`Args`** object. (B) A range of 21 different contrast values spanning 0 to 100%. (C) The Cartesian product of these two argument objects creates a new arguments object that specifies the 21 contrasts that are to be supplied to the command along with the fixed arguments.

### `Arguments`

The argument objects are the basic compositional components of Lancet. They allow the compact, declarative definition of ordered collections of arguments, where each individual argument set in the collection will be passed to a single process. These objects compose together using two operators: (1) the `*` operator which defines the Cartesian product between two argument sets, and (2) the `+` operator used for concatenation. Arguments offer a convenient way to specify parameter spaces, and they were used to define the contrast robustness analysis presented in Chapter 5.

Two examples of Lancet argument objects and their composition are shown in Figure 3.1, extracted from the notebook used to launch simulations analyzed in Figures 5.4–5.7. Using the **`Args`** class, it is easy to define a constant set of arguments. Similarly, the **`Range`** class is suitable for defining a range of regularly spaced argument values over a numeric interval. Here it is used to define a range of 21 different contrast values.

Applying the Cartesian product operator defines a new collection of 21 arguments where each set contains one of the contrast values but also includes the fixed constants declared in the `Args` object. Assigning handles to Lancet objects before composing them together allows a declarative definition to be built incrementally over the note-

book, giving an opportunity to add text to describe each individual component.

Note that all arguments values are always associated with a label. For the constant `Args`, these labels are specified via keyword arguments whereas the single label "contrast" is used for the contrast range. Taking the Cartesian product of these two arguments defines a new object, which specifies the 21 contrast values together with the constant keywords. The textual representations of these object are clear and readable, and once executed will recreate the object. This property holds for all Lancet components and their compositions.

#### `Commands`

Commands are objects that accept arguments, linking them to the execution to a particular software tool. In other words, a command defines how each set of named arguments will be used to invoke each process in the batch. As it is impossible to explicitly support all possible tools, Lancet makes it easy to define a new command class as needed. This class only needs to return a list of string values to be executed in the format accepted by the Python subprocess module.

By default, Lancet offers a generic and flexible command called the **`ShellCommand`** that specifies how command-line programs are to be executed. In many instances, using a **`ShellCommand`** is sufficient to begin using Lancet. Nonetheless, it is recommended that a tool-specific command class is used as (1) a specific command class can be appropriately documented in a way that makes sense to the users of that particular tool, (2) it encapsulates the invocation in a robust way without relying on the specification of fragile string values, and (3) it allows more sophisticated invocations of a tool, by generating input files, for instance.

The Topographica neural simulator (Bednar, 2008) used to execute the simulations analyzed in Chapter 5 offers one such example of a customized command class called **`RunBatchCommand`**. This component makes it easy for Topographica users to run simulations using Lancet by specifying a model definition file as well as a set of specific analyses and measurements to be executed over the course of each simulation run. All the measurements analyzed in Figures 5.4–5.7 were specified in this way, including orientation preference and selectivity map measurements, pinwheel analysis, and orientation tuning curve analysis.

#### `Launchers`

The last type of component needed to run batch processes with Lancet is the launcher object which encapsulates the computational platform used to execute the command and associated arguments. By default, Lancet offers two launcher classes, **`Launcher`** for running processes on a local machine and **`QLauncher`** for running processes in parallel on a Grid Engine compute cluster.

This separation between arguments, commands, and launchers makes it easier to stay within the Jupyter notebook throughout an investigation. At the start of a research project, it is typical to start with some limited, local exploration that later expands to make use of high performance computing infrastructure before publication. Without Lancet, such a process would requires ad hoc scripts or command-line invocations that change when working locally or on a cluster. With Lancet, it is easy to define a **`Launcher`** and a **`QLauncher`** together within one notebook, allowing the user to quickly switch execution of the jobs from the local machine to Grid Engine.

In simulations analyzed in Chapter 5 were all executed using this approach. First a limited set of small simulations were run locally and later many thousands of larger jobs were launched in parallel on the Eddie cluster provided by the Edinburgh Compute and Data Facility. These simulations generated hundreds of gigabytes of data on disk that had the same organization on the cluster as it was locally. This is the last key feature of Lancet that will now be discussed, namely the way it helps you manage your files as part of a reproducible workflow.

#### Tracking output files and reproducibility

A common reason to running a batch of processes is in order to execute side effects, with disk I/O being the most typical example. Manually managing large collections of data files while tracking their provenance is difficult and poses a barrier to reproducibility. Lancet helps users keep track of any large data files that are generated outside the Jupyter notebook by the launched processes. The details of how processes were launched in the past are automatically recorded, and the way Lancet executes commands consistently helps ensure a uniform organization for all the output data wherever possible. For instance, the orientation map analysis in Chapter 5 was derived from 842 Topographica simulations which generated output data organized into 842 individual subdirectories.

One key concept is the notion of an “output directory”, which holds a collection of

subdirectories containing the output of each run in the batch. Given an output directory path, Lancet will ensure each process is executed within a suitable named subdirectory and will also capture all standard output to the streams directory. The naming scheme applied to these directories may be customized, with the default scheme combining a mandatory batch name with a unique per-job integer identifier as well as the timestamp of when the jobs were launched. As a single command object specifies how all the jobs of a batch are to be invoked, each processes running in each subdirectory will have been called in a consistent way. This in turn helps ensure that output data within each subdirectory is generated consistently.

As Lancet manages the execution of all processes, starting from the definition of the argument onwards, it can keep track of all the arguments associated with each run and with the corresponding collection of output files. A .log file is generated for each batch to record the association from the unique per-job identifier appearing in the subdirectory names to the explicit set of corresponding arguments. This .log file holds information in a simple, human readable format that can then be read back into the notebook using a Lancet **`Log`** object in order to recreate the arguments used in the original execution of that batch.

Next to the .log file in the output directory, Lancet stores an .info file to store as much metadata as is available. Some of this metadata may be specified by the user although Lancet also offers helper utilities to make this process of generating metadata easy. For instance, Lancet offers a **vcs_metadata** utility that makes it easy for a user to capture version control information in the .info file. Other metadata is always stored automatically, such as various timestamps and the textual representation of the launcher object. As Lancet's objects are declarative, this representation is executable and can be evaluated in Python to rebuild a launcher object that is identical to one used to generate the original batch. This launcher can then be called to execute the jobs once more, using the same arguments, command, and launcher.

This concludes the description of Lancet and demonstrates how it helps improve reproducibility by better capturing the researchers intent without leaving Python. The key ways Lancet achieves this is by: (1) improving the ease with which researchers can specify and launch batches of jobs in Python, (2) recording the steps necessary to launch these processes within the notebook environment, (3) encouraging a clear organization of data files output by each process, and (4) automatically storing metadata to trace the output of each process back to the original arguments used to spawn it.

In order to be part of a fully reproducible workflow, Lancet must also be used in

conjunction with other essential research tools, and in particular, version control. It is essential to use version control to keep track of the notebooks themselves as they evolve, along with all the associated support code and the versions of the tool that is being launched. Large data files can be difficult to track properly but there are dedicated version control solutions that are suitable for handling large files, such as git-annex.

With Lancet, it is easy to build a productive and reproducible workflow that lasts for the duration of a research project. It is designed to work well from the Jupyter Notebook, encouraging a literate programming style for reproducible research. Lancet makes it easier to launch and manage jobs from Python, but if you also require greatly improved visualization and analysis capabilities, you will want to use HoloViews, described in the next section. In fact the `RunBatchCommand` supplied by Topographica for use with Lancet outputs simulation data in the form of HoloViews objects serialized to disk.

### 3.2.2 HoloViews: Succinct visualization and analysis

The success of a Jupyter notebook as a reproducible, literate document is proportional to how much code can be *eliminated* from the notebook while keeping a complete record of the scientific intent and the decisions relevant to the research task at hand. General-purpose code should be moved out of notebooks whenever possible into properly tracked files or into the broader Python ecosystem where it can be tested and improved by large numbers of users.

In other words, the goal of a reproducible research notebook is to tell a clear story that is focused on the task at hand, with as little distracting code as necessary. Unnecessary code in the notebook not only distracts the notebook author but it can derail later readers who wish to understand the purpose of the document. Given that code in notebooks tends to be less rigorously tested and more fragile, one way to improve reproducibility is to eliminate all unnecessary code.

HoloViews is a new library (holoviews.org) that I developed together with Philipp Rudiger that greatly reduces the custom visualization and analysis code required in a research notebook. Like Lancet, HoloViews is compositional and designed to encourage a declarative style, helping to compress large blocks of plotting code into succinct expressions. It is also designed with the Jupyter notebook in mind and aims to make interactive data exploration quick and easy. The design of HoloViews is outlined in the

Stevens et al. (2015) paper, included in the Appendix.

When working with HoloViews, you do not need to write plotting code in order to visualize and analyze your data. Instead, you declare data structures that lightly bind your data together with the associated metadata. These objects can very flexibly be nested and composed in different ways. Then whenever the notebook environment requests the appropriate representation of the data, a rich visualization is returned. In other words, HoloViews helps establish a clear correspondence between how the way your data is structured and how it is visualized.

#### HoloViews design principles

HoloViews is not a plotting library, even though it does make building complex, interactive visualizations easy. Instead, it is a library of compositional data structures that lightly wrap the supplied data in order to offer analysis methods as well as a rich visual representation. Using the object representation as its visualization, instead of relying on a separate plotting step, helps ensure immediate visual feedback at all times.

This idea of immediately returning a human understandable representation that represent an object in memory is not new. Interactive programming sessions have existed for over half a century, although this type of interaction has traditionally been purely text based. When working in an interpreter, various data structures can be directly manipulated and the changes can be immediately understood in terms of its textual representation. For instance, the string representation of a float seen by the user is not the same thing as the corresponding bytes in memory, even though this distinction is not immediately obvious as the translation is so transparent. HoloViews extends this idea of transparent translation from data to representation to make data visualization easy.

As Jupyter notebooks are hosted in the web browser, there are a host of new formats that could be used to represent an object. HoloViews makes use of the display hook system supplied by Jupyter Notebook to represent objects in many different formats, including raster formats such as JPEG and PNG, vector graphic formats (SVG), animated formats such as GIF and MP4, and finally, interactive visualizations using JavaScript that are only possible in the web browser.

As there are many semantic and esthetic choices that must be made in order to generate a rich, high quality visualization, HoloViews makes it easy to customize the properties of the visual representation associated with each object. The way this is flexibility is achieved is to enforce a separation between semantic information, the

information *about* the data, and the plotting choices which is information about how the data is to be rendered visually.

An example of how HoloViews is used in practice is shown in Figure 3.2. In a single line of code, a richly structured, composite HoloViews object is created that holds a complete set of map measurements that were recorded over the course of a simulation. This object appears in the notebook as an interactive visualization: by moving the slider, the orientation selectivity and preference measurement can be viewed at any simulation time for which this measurement data is available. The line at the top of the cell `%%opts Points (marker='s')` declares that this object is to be visualized with pinwheel locations rendered with square markers. Although this is a rich visualization, the object itself *is* the output data of the orientation preference, selectivity, and pinwheel analysis. All data is available as full-precision binary arrays, wrapped in a data structure with the textual representation shown in Figure 3.2.

The line starting with `%%opts` is not valid standard Python syntax but a form of IPython-specific syntax called a "magic". These magics are very convenient in the notebook environment, because they support additional features such as tab-completion. Although HoloViews supplies these magics, including tab-completion support, it is worth noting that *all* the features in HoloViews can be accessed in pure Python, outside the notebook environment. This allows HoloViews to be run in "headless" mode to process data and generate visualizations without user intervention which can be useful when running batch processes, e.g. when running Topographica simulations via Lancet.

Figure 3.2 serves to illustrate how the various components of HoloViews interact in a real example that will be dissected throughout the rest of this section. The way HoloViews can build this type of complex visualization with minimal code will be described in several stages, explaining how the visualization in Figure 3.2 is composed in several steps. First we turn to how HoloViews defines semantic content.

#### Elements and Composition

HoloViews is a library of atomic element classes that compose together to form collections of elements called containers. Elements are named according to particular visual representations, such as **`Curve`**, **`Histogram`**, and **`Image`**. These names serve to inform the plotting system how data is to be rendered but also helps the user construct a useful mental model regarding the dimensionality and structure of the data. HoloViews makes it trivial to cast between compatible elements types, keeping the

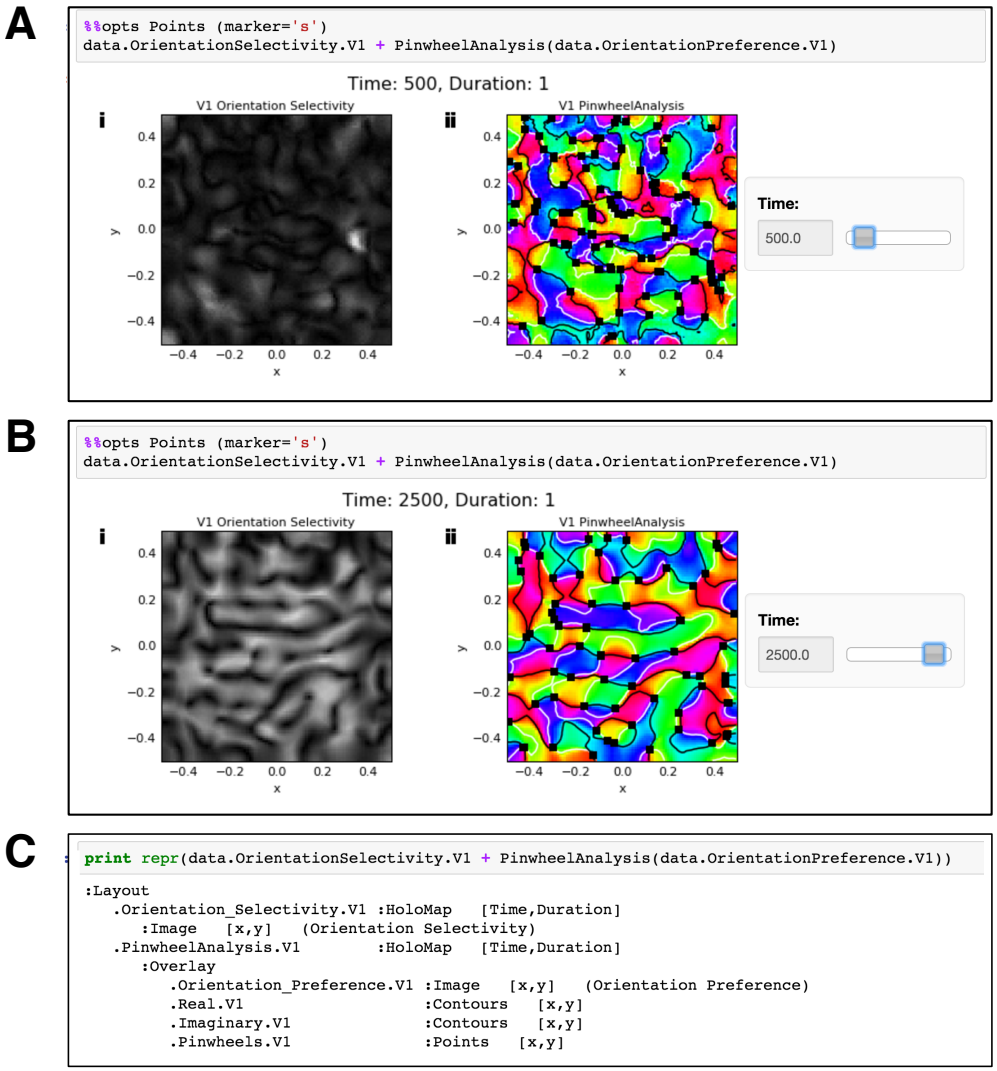


Figure 3.2: **Example of how HoloViews can be used to explore a simulation of the model analyzed in Chapter 5.** (A) Example of the simulation output at time 500, run using the Topographica simulator. Both `data.OrientationSelectivity.V1` and `data.OrientationPreference.V1` are container objects that hold the output of orientation selectivity and preference measurements. `PinwheelAnalysis` is an operation that can processes these types of container, annotating the orientation preferences with pinwheel locations using the method described in Chapter 5. The top line `%%opts Points (marker='s')` illustrates the HoloViews style system, in order to instruct the plotting code to render the pinwheels with square markers. (B) The same notebook cell with the slider moved to display the results at a later simulation time (C) This shows the textual representation of the composite object that is displayed above, built up from the components described in this section. This object contains all the raw data output by the analysis and is composed of two `HoloMap` objects, one of which is composed of `Image` objects only, combined in a `Layout` with another `HoloMap` composed of `Overlays` which in turn are composed of `Image`, `Contour`, and `Points` objects

binding between data and its assigned element class as fluid as possible.

Elements accept raw data, accessible via the objects `.data` attribute but also encourage the user to associated meaningful metadata with it. In particular, it is useful to supply semantic information regarding with the dimensionality of the data, specifying a dimension name and optionally units and allowable ranges. For instance, a **`Curve`** may be supplied a list of tuple pairs and this data will be visualization but it is useful to specify that the dependent dimension is "weight" and the independent dimension is "height" in order to give this data meaning. Once this *semantic* information has been supplied, the visual representation will be updated accordingly, for instance, by labeling the x- and y-axes with "weight" and "height".

Given a number of distinct HoloViews objects, it is possible to compose them together using the binary operators `+` and `*`. These operators are used to organize data by building composite, indexable objects. When considering the composition of elements, the crucial distinction between `+` and `*` is that `+` can always be applied whereas `*` demands the two components have matching dimensionality. In the visual representation, the `+` operator acts to position two elements side-by-side whereas the `*` operator acts to overlays the first element with the second. When the `+` operator is used, the result is called an `Layout` and when `*` is used, the result is called an `Overlay`.

These features are illustrated in Figure 3.2. First, the `+` operator is used to associate the orientation selectivity data with the orientation preference data which is then shown side-by-side. The metadata specifying the dimensionality of the data also appears in the visualization. For instance, the orientation selectivity data is labeled as `Orientation Selectivity` which appear in the corresponding title.

The leaf nodes of the data structure shown in Figure 3.2 C are the element objects, in this case the set of elements types are `Image`, `Contours`, and `Points`. These objects contain the raw analyzed data, so for instance, the `.data` attribute of the `Points` object contains the exact pinwheel locations as computed by the analysis. Alternatively, you could compute the mean orientation selectivity without any loss of precision given the corresponding `Image` objects.

Now we have seen how compositional operators and the specification of dimension metadata can be used to specify semantic content in a way that is reflected in the corresponding visual representation. In the next section, it will be shown how HoloViews also allows the customization of all the visualization options that are purely esthetic and independently of the semantic content.

### Content and presentation

HoloViews allows you to generate visualizations by constructing semantically meaningful data structures around your data, avoiding the need to write plotting code entirely. For this to work, HoloViews has a style system that allows an object's visualization to be customized without having to store this style information on the object itself.

This design is similar to the recommended separation between content and presentation in HTML and CSS, allowing visualization options to be easily changed without affecting the content. This approach makes it easy to immediately visualize data using the default visualization settings and then customized its appearance as a separate step later on in the notebook.

When the notebook environment requests the appropriate representation of an object, the HoloViews object is automatically passed to the appropriate plotting classes in the background. There, the style system makes use of a unique integer ID on the object to determine the appropriate style options. These styles are supplied by the user as keyword arguments to be passed directly to the appropriate plotting call.

To illustrate, when using the matplotlib plotting library (Hunter, 2007) as a plotting backend, a particular `Points` object may have a style associated with it, specifying the marker with the keyword pair `marker='s'` as shown in Figure 3.2. The integer id on the object is the only link necessary between the semantic content of the object and the style system and both the key (`marker`), and the value (`'s'`), are defined by the matplotlib plotting API, and not by HoloViews.

The way HoloViews passes style information to the plotting classes is entirely general and entirely independent of any particular plotting library. For this reason, HoloViews is able to support multiple plotting backends, where matplotlib is only the default option. The Bokeh library (Bokeh Development Team, 2014) is also supported by HoloViews as a separate backend, allowing for even more interactive visualizations in the browser.

The separation between presentation and content and the separation between HoloViews and the various backends further illustrates why it is incorrect to think of HoloViews as a plotting library. In terms of visualization, it acts more like a common interface for rapidly viewing data in notebooks that makes it easy switching between different underlying plotting libraries.

#### Immediate, interactive, and exploratory visualizations

Using the style system together with elements, layouts, and overlays allows HoloViews to quickly generate static plots that could also be defined with blocks of imperative code using a plotting library. Perhaps the most compelling reason to use HoloViews in the Jupyter notebook is for its flexible and general system for interactive data exploration.

In addition to the composition operators introduced in the previous section, elements can be grouped into high dimensional containers for instant interactivity. The core container type, called the `HoloMap` may be thought of as high dimensional dictionary that allows the user to scrub through the set of keys with a JavaScript slider in order to browse the elements it contains.

As with all HoloViews data structures, `HoloMaps` can be composed using the `+` and `*` operators and accept additional metadata to specify the dimensionality of the space in which the elements reside. Keys are specified as n-tuples, where each position in the tuple corresponds to the value of a particular dimension of the container associated with the value. `HoloMaps` can be visualized as animated GIFs, as video using MP4 but most commonly, they are rendered as interactive visualizations where they key dimensions define corresponding JavaScript sliders as seen in Figure 3.2A and B.

`HoloMaps` offer an extremely convenient and flexible way to explore data. Any software can integrate with HoloViews by returning `HoloMap` objects making it easy to explore results in the notebook. As all HoloViews objects are thin wrappers around the data, this is a powerful way of simultaneously processing data and generating the corresponding visualization at the same time. In addition, it is easy to build operations that accept `HoloMap` objects as input in order to generate a new, derived `HoloMap`. This makes it possible to build data processing pipelines using HoloViews operations and this is the approach used by the Topographica neural simulator (Bednar, 2008) for the visualization and analysis code.

Figure 3.2 shows an example of this approach. The `data` variable is output by the measurement code in Topographica that integrates with HoloViews and `PinwheelAnalysis` is an operation that locates the pinwheel singularities from orientation preference maps, generating a new `HoloMap` of the analyzed data. This `HoloMap` is composed of `Overlays` objects which are same type of object as created by the `*` operator. These overlays have the original orientation map image on the bottom layer, overlaid with the real and imaginary contours of the polar representation (shown in white and black respectively) and

then finally, the points marking the identified pinwheel locations. The exact analysis data is held by these objects, computed using the methods of Löwel et al. (1998) and described in more detail in Section 5.3.3.

The overall data structure shown in Figure 3.2A is an example of a `Layout` of `HoloMaps`, which allows data to be interactively explored with widgets across multiple `HoloMaps` in the same way as they can be explored individually. In this figure, the widget traverses the dimension associated with the `HoloMaps` keys, namely simulation time. Both subfigure A and B display the same notebook cell but with the slider moved from a simulation time of 500 to a simulation time of 2500. Using `HoloMaps` make it easy to interactively explore entire data sets in this way.

The style system directly applies to elements within HoloMaps, also illustrated in this figure. The default marker style for pinwheels is circles, as seen later on in Figures 5.4-5.7. In this example, the keyword `marker='s'` has been passed to the matplotlib backend for rendering the `Points` objects in the `HoloMap`, switching the points to use square markers across the `HoloMap`. Note that the style information regarding this choice of marker is stored separately from the object itself, which is why this information will not appear in the textual representation of the object of the sort shown in 3.2C.

## 3.3 Discussion

In this chapter, two new libraries have been introduced that have greatly improved research efficiency and productivity in the Jupyter notebook. Framed this way, these projects are not primarily about achieving scientific reproducibility. Paradoxically, not focusing on reproducibility is a crucial feature for any reproducible approach to gain traction, as without a clear, pragmatic incentive, users have no reason to switch to a more reproducible workflow. There need to be clear productivity advantages before researchers engaged in less reproducible practices will be willing to switch to better work patterns.

One of the issues with reproducibility is that it is often perceived to be in opposition to research efficiency. For instance, researchers are known to invest time and effort in order to make their work available and reproducible after publication. Reproducibility is generally understood to be important but it is also seen as an additional burden to an already challenging research process.

The core philosophy underpinning the projects presented in this chapter is that,

in the right context, research efficiency can serve to *improve* reproducibility. A lot of the difficulties regarding reproducibility in computational neuroscience stem from workflows that are ad hoc, unmaintainable, and inefficient. A solid workflow is one that with practical benefits by making the research process quicker, easier and more maintainable. Approaches that aim to improve reproducibility but that prove inefficient or onerous to use will fail to gain traction.

Reproducibility is recognized as important across the research community, but many do not see the practical benefits of reproducibility as part of the daily research process. An experienced programmer will comment code, not just to assist other people but to aid their own understanding at a later date. Similarly, a researcher who uses version control to track concise, well-structured notebooks will find that, not only is their research easier to communicate with others but that their regular workflow has also become easier to manage.

Both Lancet and HoloViews are designed to allow a researcher to express their intent as succinctly and efficiently as possible. As a result, not only can you do more with less code, you more functionality can be included in a single, self-contained notebook while presenting a clear scientific story. Together with version control, the adoption of clear, well-written notebooks as readable, literate documents will go a long way towards improving scientific reproducibility.

To date, HoloViews has been a very successful open source project with a more general scope than Lancet and a correspondingly greater adoption. It won in its category in the UK Open Source Awards 2015 and is now in use by researchers worldwide, both in computational neuroscience and in other fields. For instance, it has been used as part of an EdX online course in condensed matter physics (*Topology in Condensed Matter: Tying Quantum Knots*). This made use of an extension to HoloViews to support plots generated by `qutip`, the "Quantum Toolbox in Python". It has also generated plots used in several physics publications (Nijholt and Akhmerov, 2015; Tenner et al., 2016). In addition, HoloViews has been extended to create a new project called GeoViews with support from the U.K. Meteorological Office (Met Office). GeoViews adds support cartographic projections for exploring geographical and meteorological datasets.

These examples serve to show that HoloViews is an entirely general tool, growing in popularity, which can support researchers from the initial exploratory research stage to final publication. The adoption of Lancet has been more limited, but it has been used by other researchers to launch microprocessor simulations, again helping to

demonstrate the generality of the approach (Elver and Nagarajan, 2014).

All the code used in this thesis is publicly available and under an open source license (BSD 3-clause), including the Topographica simulator and all dependencies. The work presented in Chapter 5 is based on a fully reproducible publication using Jupyter Notebooks (Stevens et al., 2013b). These notebooks demonstrate the use of Lancet to manage and execute simulations from Python, but predated HoloViews. Had HoloViews been available at the time, the work necessary to generate reproducible figures would have been dramatically reduced. The notebooks associated with this paper are available from
https://github.com/ioam/topographica/tree/master/models/stevens.jn13.

## 3.4 Conclusion

Lancet and HoloViews are general, open source research tools that enhance both scientific productivity and reproducibility. By allowing researchers to declare their intent with less code within a literate programming environment, specifically the Jupyter Notebook, data can be rapidly generated, visualized, and explored in a more reproducible way. This results in a more powerful, more efficient, and more enjoyable scientific workflow.

The general design of these tools has led to them being adopted by scientific researchers across the world and across disciplines. Although HoloViews was essential for enabling the work presented in this thesis, the generality and flexibility of the design means that the same tool that enables rapid, interactive data exploration has also been used to generate published visualization in an entirely different field, namely quantum physics.

# Chapter 4

# Dynamics of the evoked response across spatial scales

Understanding how neurons respond to a sequence of visual images is crucial for establishing the function of the primary visual cortex (V1). In this thesis, the aim is to build a mechanistic model that can account for the spatiotemporal responses of neurons in macaque monkey V1 across a wide range of both spatial and temporal scales. To make initial progress towards this goal, in this chapter we will focus on understanding the time course of neural activity in response to a brief stimulus (a peri-stimulus time histogram, or PSTH). Later chapters will then consider how these responses vary in the long term, over the course of development. Thus for the purposes of this chapter, we will consider response properties to be a fixed function of a neuron's inputs, not simulating any plasticity or development.

Despite the many years of study of V1 using different experimental techniques, characterizing how a population of neurons responds even to a simple visual stimulus turns out to be surprisingly difficult. The main reason is that there is no single technique that can accurately measure the firing rates of the many individual cells that make up a large population. Single-unit electrophysiology can very accurately record the spiking activity of individual neurons responding to a stimulus, but even large electrode arrays can only return information from a very sparse sampling of the cells present. Voltage-sensitive-dye optical imaging (VSDI) provides complementary data, reporting temporally precise information about membrane voltage from very large numbers of cells, but lacks the spatial accuracy to resolve individual neurons, instead integrating indiscriminately across neural tissue that includes many different cell types, *en passage* fibers, etc.

Because these two techniques are measuring quite different quantities, it is not surprising that the results from such studies differ dramatically. As shown below, single-unit studies indicate that individual neurons have sharp, transient responses, yet population measurements indicate slow, gradual, sustained responses for the population as a whole. Resolving this discrepancy is the focus of this chapter, because it will allow us to relate single-unit and population data to characterize how a densely connected network of neurons responds to a stimulus over time. In a series of steps, I will show how to extend the well-validated IRD model (Albrecht et al., 2002) of single-unit responses into a model for the population as a whole that is compatible with both the single-unit and VSDI data.

Specifically, three new mechanisms needed to be added to the single-unit model, each accounting for a particular feature of the VSDI signal. Each mechanism is introduced only when necessary, using the simplest mathematical formulation consistent with the experimental data. The result is the first spatially extended, mechanistic account of the voltage-sensitive–dye signal.

This model is distinct from previous work on the VSD signal that has focused either on modeling the detailed spiking biophysics within the extent of a single VSDI pixel (Chemla and Chavane, 2010a), or on modeling the spatiotemporal dynamics in terms of the bulk electrical properties of the tissue, without reference to single-unit responses (Sit et al., 2009). In contrast, the model presented in this chapter has a simple mathematical formulation that shows how the firing rates of individual neurons are transformed by the structure of the network in a mechanistic way. This implementation will serve as a reference for the spatiotemporal developmental model to be presented in Chapter 6.

The sections below analyze the differences between single-unit responses characterized by the IRD model, population responses from a diverse collection of neurons each characterized by the IRD model, and a spatially organized population with spatially dependent responses. Each section builds incrementally on the IRD model, introducing one new mechanism in turn, along with the experimental data that motivates it: (1) latency scatter, which effectively stretches out the time constant of the VSDI signal compared to individual units, (2) spatial variation in latency with respect to the stimulus center, and (3) diversity of tuning dependent latency across units. Together these mechanisms characterize important types of diversity of the units in the real neural population, and the resulting model is able to predict accurate VSDI responses from the single-unit activities, allowing us for the first time to illustrate how each of the

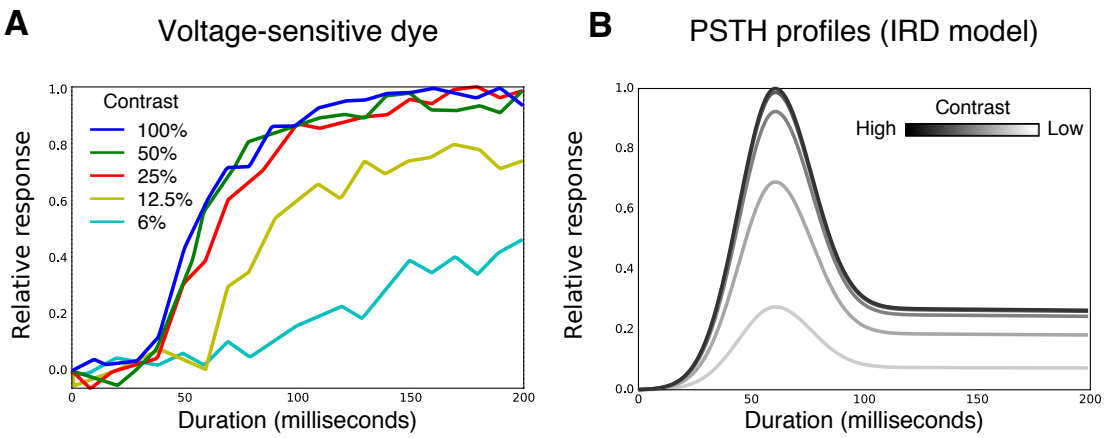


Figure 4.1: **Temporal onset profiles recorded using voltage-sensitive imaging are qualitatively different from temporal profiles of single-unit spiking data in macaque V1.** (A) Normalized voltage-sensitive–dye response for the five contrasts indicated by the color key for the central region of interest in Sit et al. (2009) that was marked in Figure 2.6C. (B) Normalized PSTH profiles generated by the IRD model of Albrecht et al. (2002) for five contrast values chosen to match the corresponding voltage-sensitive–dye amplitudes for easier comparison. The resulting onset times for both types of data are similar, but the PSTHs quickly reach a peak and then decay, while for this constant stimulus the VSDI signal slowly reaches a plateau.

experimental techniques relates to the underlying neural activities.

## 4.1 Dynamics of electrical and optical responses

Single-unit electrophysiology allows precise measurement of the activity in a small neural population with a high temporal resolution, whereas optical imaging measures the overall activity across an extended volume of neural tissue (see section 2.3 for detailed background). Data obtained using electrophysiology is often easier to interpret, since it is a direct measurement of the cell's electrical activity, whereas voltage-sensitive imaging more indirectly conveys the bulk response of the tissue. However, it is very difficult to infer properties of the whole population from the sparse and biased sample of single-unit data available, and so both types of evidence are invaluable.

Figure 4.1 shows results from both methods, including PSTH profiles fit to single units by the IRD model, and VSD responses for similar stimuli. Both plots show neural responses evoked by similar stimuli of various contrasts presented for 200 milliseconds, one measured electrophysiologically, and the other with VSDI. The VSDI

results are measured from the retinotopic location corresponding to the stimulus, so that they will be comparable to the single-unit PSTH data.

There are some very broad similarities between the two experimental data sources shown in Figure 4.1. Both signals first rise at a roughly similar time (though slightly later for the VSDI response). In both, low-contrast stimuli lead to slower responses and a lower overall response amplitude. Responses for both types of signal saturate with contrast, as neurons approach their maximal response rate for higher contrast levels. This response saturation is well approximated by the Naka-Rushton equation given in Equation 2.3 as the contrasts used in B only differ from the corresponding contrasts in A by a constant multiplicative factor.

However, in nearly every other respect, the two types of profile have very distinct temporal properties. The most obvious discrepancy is that the VSD response gradually approaches a plateau reached only by 150 or 200 milliseconds, while the PSTH profiles each peak within the first 75 milliseconds. I.e., the optical signal monotonically rises throughout the response, while the firing rate rises and then falls dramatically. There is also a small mismatch in the onset time, with these PSTH profiles rising slightly earlier than the voltage-sensitive–dye responses.

One way to express the difference between results from each method is in terms of varying time constants, by analogy to various physical systems exhibiting hysteresis or damping. The single-unit PSTH profiles rise and fall rapidly around the peak, indicating that the firing-rate response has a low time constant. Conversely, the changes in the voltage-sensitive–dye response occur more gradually, suggesting that a large time constant is associated with the population-level interactions reflected in the VSDI signal.

This larger time constant is not a property of the voltage-sensitive–dye itself, which is able to transduce fluctuations in electrical potential within a few milliseconds (Chemla and Chavane, 2010b). The non-linear relationship between membrane potential and spiking activity also cannot account for this discrepancy, as the incoming afferent activity from the LGN already has a strongly peaked PSTH profile. Because changes in the membrane voltage are driven by spiking input received synaptically, the fact that the afferent input from the LGN cells also has a clearly peaked PSTH profile suggests that the membrane potential for individual cells will also have a similarly peaked shape. Figure 2.4 showed the firing-rate responses for magnocellular and parvocellular LGN neurons recorded from macaque (Maunsell et al., 1999), indicating that a peaked response applies to both the spiking input and spiking output of cortical neurons.

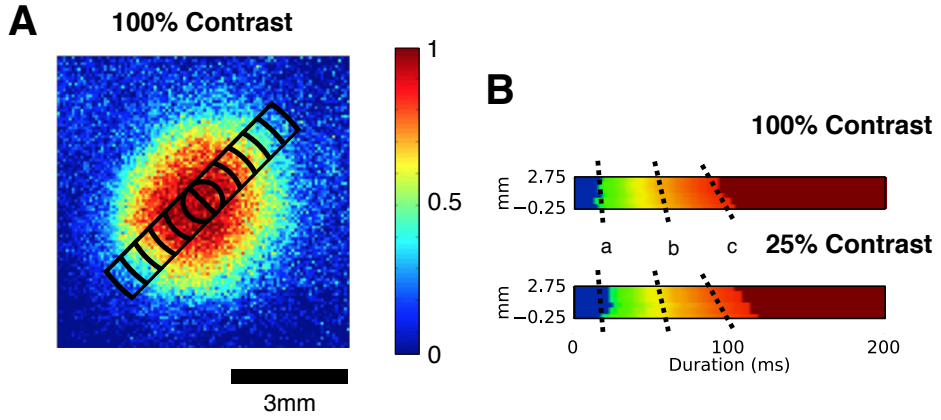


Figure 4.2: **Increasing spatiotemporal gradient over the course of the VSDI response in macaque V1.** (A) Analysis region boundaries overlaid on top of the peak response to the stimulus at 100% contrast. The optical response is averaged for each pair of successive annuli at each distance from the center. Each of these mean responses over time gives the data for one row of the spatiotemporal plots on the right. (B) Spatiotemporal plots for the 100% and 25% contrast conditions. At the point where evoked activity is first detected (a), the response appears almost simultaneously across cortical space. Part way through the response (b), a spatiotemporal gradient emerges. Lastly as the response starts to plateau (c), there is an increased spatiotemporal gradient. In other words, the response in the more distal regions takes longer to plateau than the response in regions near the center. The three dashed lines are shown at the same position in the two plots, making it clear that the overall response is also delayed in the lower contrast condition. Adapted from Sit et al. (2009).

In addition to the differences in the temporal profiles for neurons at the center of the response, there are spatially dependent effects at the population level that also require explanation. Figure 4.2B shows the spatiotemporal analysis of the VSD response for two different contrast conditions, reproduced from Sit et al. (2009). There is a variation in latency across space that is time dependent, with a nearly simultaneous onset across cortical space but a diverging latency to the peak response. In Figure 4.2 this phenomenon is indicated by the increasing spatiotemporal gradient, marked by the dotted black lines at different points in the response. These effects also vary as contrast is reduced, which will be discussed near the end of this chapter.

So far we have been focusing on the stimulus onset, even though the offset response data is also available in Sit et al. (2009). For two main reasons, only the duration of

fixation after the stimulus onset will be considered in this chapter. First, the IRD model is only defined for stimulus onsets. Second, there is much more experimental data regarding onsets than offsets, and in particular all the data presented in this chapter were recorded with respect to stimulus onset. In any case, there are clearly dramatic differences in the onset responses between the two experimental techniques that require explanation.

### 4.1.1 Relating the response signal across scales

The VSDI response and the firing response profiles described by the IRD model are a reflection of neural response at two very different scales. The optical response reflects the activity of a large, diverse neural population whereas the IRD model captures the typical firing rate responses of individual cells. The differences in the two response types shown in Figure 4.1 are therefore related to understanding the properties of the response for different neural population sizes and across different spatial scales.

The challenge of bridging across these different scales can be tackled either by a top-down experimental approach or a bottom-up modeling approach. The experimental approach would be to observe the activity at the population level and use a combination of experimental techniques to identify and quantify the different contributions from the elements of the population, such as different cell types, synaptic inputs, connectivity profiles, etc. Such an approach is becoming more feasible due to new genetic and imaging techniques, but is still a long way from being truly practical, particularly in macaque.

The bottom-up modeling approach is to start with a description of the single-unit responses and to consider a large population of such neurons, incrementally adding diversity or new mechanisms to this population to account for additional types of cells or interactions, until the resulting model approximates the VSD signal. This chapter takes the latter approach, starting with a population of IRD model units to capture the observed responses of single neurons. At first this population will be entirely homogeneous, with every unit responding in exactly the same way. This will result in an aggregate, population response with exactly the same temporal response properties as individual neurons, which is clearly not the case in animals.

Of course, real neurons in a population are not homogeneous in their responses, due to differences in their intrinsic properties, structural variations in their connectivity, and their different tuning properties. It is also important to recognize that the IRD

model captures the single-unit responses of a subpopulation of neurons that respond to a specific stimulus protocol. No experimental procedure can capture the full diversity of all possible responses across a neural population, and it is necessary to study the methods used to understand the explicit and implicit biases that result. In other words, the particular details of the stimulus protocol used to calibrate the IRD model have crucial implications regarding the types of responses that are adequately captured by this model.

Note that the IRD model is a high-level, descriptive model that maps from an input of a specified contrast to a cortical firing-rate response profile comparable to experimentally measured PSTH curves. It does not model specific input patterns or processing of those patterns by circuitry in the eye and LGN, and so it cannot directly relate the activity across cortical space in terms of the retinal image. Instead, it is designed to fit only the response for a stimulus well matched to this particular neuron's preferred retinotopic location and pattern shape, including orientation preference.

The task of this chapter is to consider the IRD model as a summary of single-unit responses and to ask how these responses may diverge across a population. By considering different types of such diversity, it will be shown how to generate a population of neurons whose temporal responses are governed by the IRD model but which together can account for and predict the observed VSDI signal when the activities are pooled together. Each new addition to this population yields a qualitative change in the simulated spatiotemporal response, eventually bringing it in line with the observed VSD dynamics.

The following mechanisms will be considered in turn: (1) latency scatter to explain the increased time constant of the VSDI signal relative to the single-unit PSTHs, (2) spatial variation in latency with respect to the stimulus center, which is then combined with (3), diversity in latency due to the variable tuning properties of the different cells in the population. The resulting model then explains both the spatiotemporal gradient shift as well as the extended plateau in the VSD signal response, showing how individual cells can have responses well-characterized by the IRD model yet the overall population responds as in the VSD imaging.

## 4.2 The time constant of the population response

In order to understand the time constant of the voltage-sensitive–dye imaging signal, it is worth considering the simplest form of temporal diversity missing from the IRD

model. Given a particular contrast and set of model parameters, the IRD model always outputs a fixed description of the corresponding firing response profile.

A collection of PSTH recordings from multiple cells are not all the same as the profile used to summarize them, varying in response amplitude, shape, and latency. In order to explain the temporal properties of the population response, it makes sense to consider the response latency spread across the different neurons. As a basic intuition, it would make sense for the population response after averaging across a temporal spread to act with an increased time constant. This hypothesis is examined next.

### 4.2.1 Latency scatter across the cortical population

In order to decide on an appropriate distribution of latency scatter across a population of IRD units, the first step is to considering whether a diversity of responses could be generated by the IRD model itself. The curves shown in Figure 4.1B used the mean parameter values of the IRD model defined in section 2.3.1, but the standard deviations of these values were also published. This suggests a way to generate a diversity of curves using the IRD model itself.

I.e., you can model model the variation of a parameter given its mean and standard deviation by simply using a Gaussian distribution. By sampling for each parameter of the IRD model from a Gaussian probability distribution with the appropriate mean and standard deviation, it is possible to obtain a diverse population of IRD PSTH curves from which a latency scatter histogram can be derived.

Figure 4.3A shows the relative onset distribution of the IRD model as inferred from 1000 profile samples assuming normally distributed parameters. The means and standard deviations used are as stated in Albrecht et al. (2002) and are available in section 2.3.1. For the purposes of validation, this distribution is shown next to an experimental onset distribution recorded directly from macaque V1 in Figure 4.3B (Nowak et al., 1995).

There is a marked difference in the inferred IRD latency distribution and directly recorded distributions, demonstrating that a diversity of profiles cannot be generated from the IRD model parameters in this way. The reason for the mismatch is that a core assumption in the inference process is invalid. The parameters of the IRD model are *not* normally distributed, which is quickly established by comparing the mean parameter values and their standard deviations with their corresponding median values, also given in Albrecht et al. (2002) and section 2.3.1. Of the ten parameter values of the

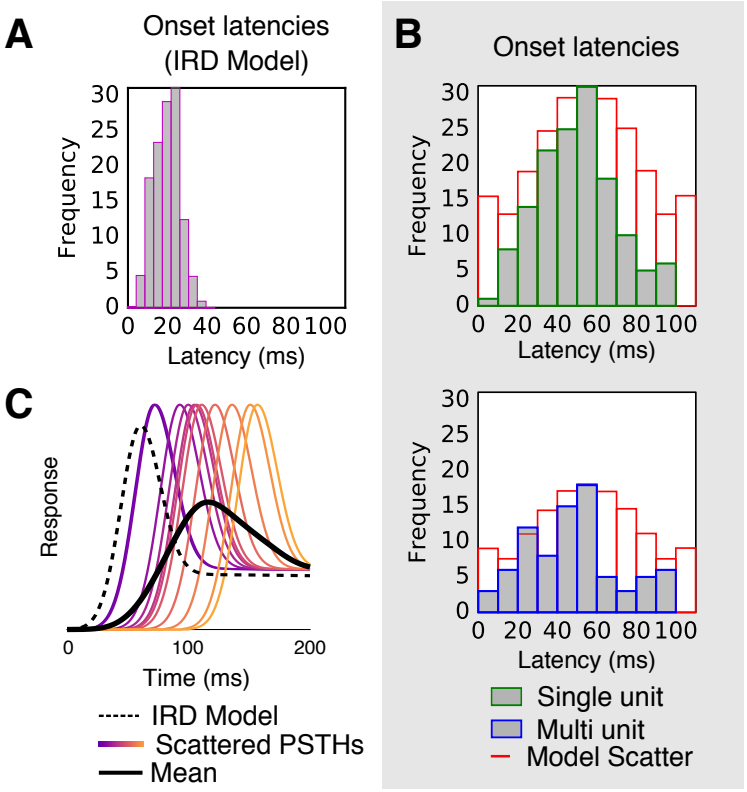


Figure 4.3: **Calibrated latency scatter across a population of IRD PSTH profiles.** (A) Onset latency scatter sampled from $1000$ PSTH profiles generated with the IRD model, where onset is defined as the time to $10\%$ peak height. Profiles were generated by assuming all parameters of the IRD model are normally distributed with the means and standard deviations given in section 2.3.1. (B) Experimentally recorded onset latency scatter distributions in macaque V1 (Nowak et al., 1995) are much broader. This analysis shows that it is not possible to generate realistic diversity from the IRD model using normally distributed parameters, presumably in part because the actual distributions are highly non-normal. Comparison shows zero-aligned single-unit (green) and multi-unit (blue) zero-aligned distributions. Red histogram outline shows the Gaussian latency distribution profile used later in the model. Heights of model latency histograms normalized to the maximum frequency of the corresponding experimental plot. (C) Mean response (solid black line) of a population of $100$ IRD PSTH profiles after the application of the calibrated scatter. Ten randomly sampled profiles from this population are shown, indicated by the color gradient. The template IRD curve ($100\%$ contrast) without latency scatter is shown by the dashed black line. Grey background indicates experimental data, a convention used throughout this chapter whenever model and experimental data are shown together.

IRD model, only one of these, $s_{50}$, has a median that falls within a standard deviation of the mean, whereas normally distributed data will have medians close in value to the means.

As the parameters of the IRD distribution cannot be assumed to be Gaussian, their distribution profiles are only weakly constrained by the values supplied with the IRD model. As a result, the diversity of IRD responses will need to calibrated using data from a different experimental source. In our initial work, we will use the experimental data that has already been shown in Figure 4.3B, i.e. the observed distribution of onset latencies.

The two latency distributions shown in B are obtained by directly recording latency onset distributions, using the same microelectrode to compute single- and multi-unit distributions. This data from Nowak et al. (1995) is a suitable match for calibrating the latency scatter of IRD model units as (1) it also is recorded in macaque V1 (2) it also represents the responses of neurons responding to an oriented stimulus centered on the cell's receptive field. Specifically, both studies varied spatial frequency, orientation, and retinal position of the stimulus until the response of the neuron was maximized, and then measured the onset latency.

Both the directly recorded latency distributions shown in Figure 4.3B *are* approximately Gaussian, but onset latency is not directly parameterized in the IRD model, and the actual parameters are non-normal. To model the latency scatter explicitly, samples are drawn from a Gaussian distribution, resulting in the distribution profile indicated by the red histograms. This distribution is made to be symmetric with some additional weight in the first and last bins. This is because latencies must be positive, requiring some redistribution of probability mass. In addition, the chosen distribution is slightly wider than suggested by the data in order to (1) acknowledge that any finite experimental sampling is likely to under-represent the true width of the latency distribution in V1 (2) as we shall see shortly, latency scatter is *not* sufficient to explain the VSDI signal and this is most clearly demonstrated by using a plausible level of scatter that is slightly highly than what is experimentally observed (to guard against issues like (1) obscuring the results).

To do this, we will apply the chosen scatter distribution to a population of IRD profiles generated at 100% contrast, shown in Figure 4.3C. All the profiles have the same shape and are simply shifted relative to each other by the latency scatter, representing a localized group of neurons that are responding in a similar way but have different absolute onset times. When averaged, these responses do have a higher time constant,

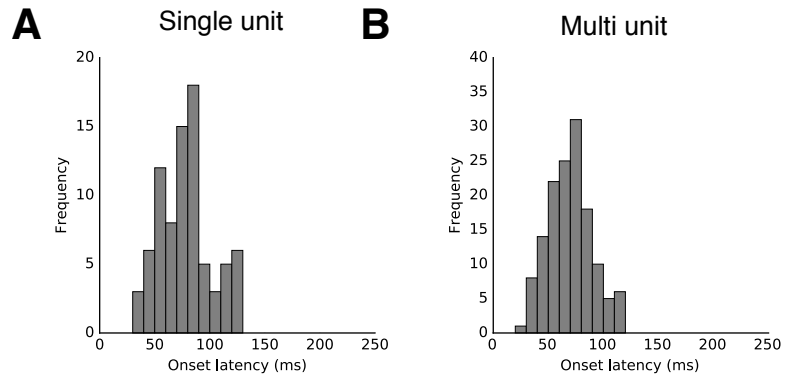


Figure 4.4: **Distribution in onset latency evoked by the onset of a flashed rectangular spot of light in macaque V1.** (A) Distribution of onset latencies in spiking activity recorded using single-unit recordings (B) Distribution of onset latencies in spiking activity recorded as multi-unit activity. Onset was computed from the PSTH histogram by finding the bin at which the Poisson $P = 0.01$ level was crossed, provided the next bin did not fall below this level and the following bin did not fall below the $P = 0.05$ level (Maunsell and Gibson, 1992). Reproduced from Nowak et al. (1995).

as indicated by the solid black line. The template IRD profile without any application of scatter is shown by the dashed black line.

The relative onset distributions shown so far begin at zero, to indicate that causal processes require positive latencies. The issue with this formulation, after scatter is applied, a typical PSTH shown in Figure 4.3C has an additional shift relative to what is predicted by the IRD model, shown by the dashed black line. To correct for this effect, we need to look at the *absolute* latency distribution which is also given in Nowak et al. (1995) and shown in Figure 4.4.

Figure 4.4 shows how the systematic latency increase due to the application of scatter after the IRD model profiles are generated can be correctly compensated for. What is necessary is to align the absolute onset latency distribution of the IRD model units after scatter so that the distribution begins at the same absolute latency as shown in this figure. For the single-unit data, the first bin appears at approximately 40 milliseconds and for the multi-unit data, the first bin appears at around 30 milliseconds. All the simulations in this chapter involving latency scatter apply the necessary shift to align with these absolute latency distributions.

What Figure 4.3C shows is how the time constant of a population of IRD units is increased when averaged after the application of calibrated latency scatter. The population response is now a closer match to the VSDI response profiles shown in

Figure 4.1 than any individual single-unit response curve. Latency scatter is therefore a promising mechanism for increasing the time constant across a spatially extended region *without* modifying the time constant of single-unit profiles.

Note that the width of the plateau region remains narrower than in the experimental data, which will be addressed in the final section of this chapter. In addition, latency scatter will not be able to explain the spatiotemporal gradient effect shown in Figure 4.2D. The collection of IRD units we have been considering have no spatial position and will therefore be unable to account for these kinds of spatial effects.

As discussed in section 4.1.1, the goal is to identify the various forms of response diversity across a population that are *not* captured by the IRD model. Now that latency scatter has been accounted for, we will focus on responses not captured in the onset latencies distributions recorded by Nowak et al. (1995). Having just noted a spatial effect at the population level not captured by latency scatter, we will now examine how neural responses across space. In particular, we will examine the neural responses when the stimulus is not spatially centered on an individual cell's receptive field, as it was in both the IRD model and in the onset latency recordings.

## 4.3 Dependence between space and time

As formulated so far, the model is a collection of IRD response profiles, each with its own variable latency. As noted in the previous section, not only is this insufficient to fully account for the VSDI signal at the center of the response, but it will not be able to account for the spatial effects described in Figure 4.2 as there is no description of how the population of neurons are extended spatially.

In this section, the implicit assumptions of the IRD model are examined, focusing on what particular types of response this descriptive model may not have captured with respect to the distance across cortical space.

### 4.3.1 The implicit assumptions of the IRD model

The IRD model is a descriptive model that summarizes the firing-rate response of simple and complex cells in cat and monkey to a particular stimulus. This stimulus is an stationary grating pattern, chosen to be at the optimum spatial frequency, orientation, and phase for each recorded cell, centered on the cell's receptive field.

This is to say that the measurements used to build the IRD model capture a par-

ticular relationship that is enforced between the recorded cell and the stimulus that drives it. This constraint reflects the decisions made by every experimenter to study a subset of all the possible neural mechanisms. It is simply impossible to capture all the possible responses to all possible visual stimuli for all the recorded cells.

When considering a population response in order to understand the VSDI signal, it is necessary to consider how these constraints should be relaxed in order to think about other responses in the population that have not been adequately captured. In particular, the spatiotemporal phenomenon observed at the population level, shown in Figure 4.2 indicates there must be some spatially dependent response change that has not yet been considered.

In the population model as formulated so far, all the simulated IRD profiles are independent and are not assigned any particular position in the cortex. That is to say, if each IRD unit in the population were stimulated and measured in turn, with the appropriate changes made to the stimulus parameters between successive trials, then the current model would be valid. In reality, all cells respond at once, such that only a minority of cells will be optimally driven by the particular visual stimulus presented at any given time. Given what we know about the spatiotemporal properties of the VSD signal, we are looking for diversity in responses across the population that varies with space.

### 4.3.2 Observed latency variation across space

In the same way that the latency distribution expressed in Nowak et al. (1995) connects to the IRD model due to similarities in stimulus protocol, it is now necessary to find a new source of data that makes contact with these two protocols that both optimize the stimulus for a particular recorded cell. This contact needs to be made only at one point, as we are now interested in what happens when the constraints imposed by this stimulus protocol are relaxed. In particular, we want to know what happens when the stimulus is *not* optimized spatially and begins to fall outside the cell's receptive field.

Suitable data can be found in Bringuier et al. (1999) showing how the onset latency of the subthreshold depolarizing response changes as a function of the eccentricity of the stimulus from the receptive-field center, for two different stimulus protocols. One type of stimulus used was a square patch whose location with respect to the receptive-field center is varied over two dimensions. The other stimulus was a long bar that is varied over the direction orthogonal to its orientation. The change in latency of

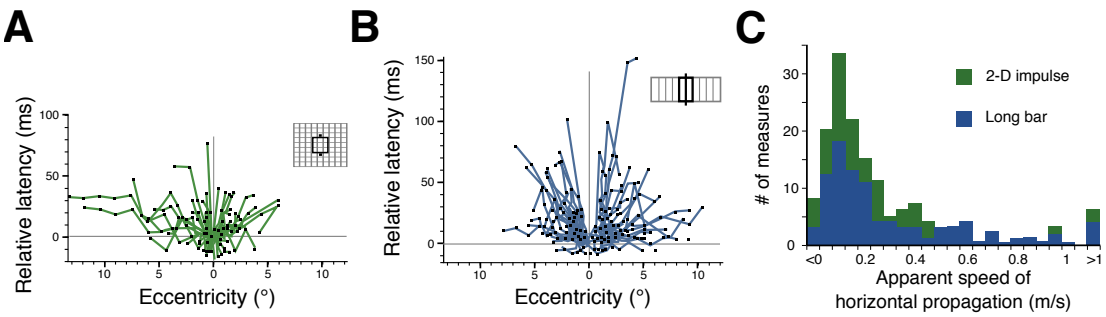


Figure 4.5: **Latency change in subthreshold depolarizing response as the stimulus moves away from the receptive-field center in cat V1.** (A) Relative latency as a function of eccentricity for a square, impulse-like input ($0.4^\circ \times 0.4^\circ$) that varied in two dimensions relative to the center of its receptive field (n=37) (B) Relative latency as a function of eccentricity for a long flashing bar ($0.9^\circ \times 3.9^\circ$) whose position is varied over the direction orthogonal to its orientation. (C) Estimate of the apparent horizontal velocity based on the data shown in A and B. The mode appears around $0.1\ \mathrm{mm\,ms^{-1}}$. The latency clearly varies depending on the stimulus location relative to a cell's receptive field, with widely varying levels of spatial dependence. Reproduced from Bringuier et al. (1999).

the response as a function eccentricity for both these protocols is shown in Figure 4.5 where the response onsets were determined using intracellular recordings in cat primary visual cortex.

There is a clear connection between this stimulus protocol using the rectangular bar and the stimulus protocols we have been considering at eccentricity zero. Only at this eccentricity is an oriented bar presented at the center of the recorded cell's receptive field, in the same way it was for both the IRD model and for the scattered recorded by Nowak et al. (1995). Using this data, we can now observe how the latency changes when the eccentricity varies away from zero.

What can be seen in Figure 4.5B is that there is a spatially dependent latency shift as the eccentricity of the stimulus changes with respect to the cell's receptive field center. This spatiotemporal phenomenon is a clear candidate mechanism for explaining the spatiotemporal gradients in Figure 4.2B. It is important to note that as only relative latencies are shown, these spatially dependent latency changes are independent of the latency scatter that has already been introduced, i.e., additive on top of the zero-eccentricity scatter characterized by the IRD model.

The goal is now to incorporate this spatially dependent latency shift to our popu-

lation of scattered IRD units. As the current population of IRD units do not have an assigned spatial position, it will be necessary to build a spatial extension of the IRD model. This introduces the spatially extended IRD (SIRD) model that will be used throughout the rest of this chapter.

Before switching to modeling considerations, it is worth noting that the exact nature of the spatial latency dependence is difficult to determine exactly due to the large spread between the cells recorded. This wide spread is present in 4.5B but it is even more pronounced in Figure 4.5A where a square stimulus was used. This is a clear example of how a different feature, namely a different stimulus shape can impact the spread of onset latencies.

Lastly, it is worth mentioning the hypothesis put forward by Bringuier et al. (1999) to explain this effect. It is proposed that the spatial latency shift occurs as the dominant contribution to the response shifts from an afferent drive when the stimulus is centered on the receptive field to slower lateral intracortical interactions when the stimulus moves out of the classical receptive field. This suggestion is discussed later in section 4.7.3 when we try to interpret the SIRD model but for now, we will continue to calibrate the model in a way that is agnostic to the underlying mechanism.

### 4.3.3 A spatial extension of the IRD model

The model now needs to be updated to incorporate this second form of response diversity. This will be achieved in two stages. First, one equation of the IRD model will be modified to accommodate a distance-dependent term. Secondly, the IRD profiles will be assigned a cortical position in order to enable a spatiotemporal analysis to allow comparison with the experimental data. These two changes, in addition to the latency scatter constitute the core of the spatially extended IRD (SIRD) model.

#### A distance-dependent latency equation

The first step is to identify the most appropriate place in the IRD model that would allow a dependence between spatial position and latency to be modeled, in order to account for the phenomenon shown in Figure 4.5. The most suitable equation to extend is exactly where the PSTH latency is computed. In the IRD model, a latency shift is computed as a function of contrast, $\tau_c(c)$ as shown in Equation 2.2.

The issue of contrast-dependent latency shifts has been ignored until now, even though characterizing contrast responses was central to the IRD model itself. This

omission is deliberate, allowing us to simplify the analysis of the core mechanisms of the SIRD model. Coupling between response onset latency and contrast will be the last mechanism to be (re)introduced to the SIRD model.

For this reason, the $\tau_c(c)$ term of the IRD model may be replaced by a fixed value of 60 milliseconds for the time being, corresponding to Equation 2.2 evaluated at 100% contrast. What is relevant to this section, is that $\tau_c(c)$ can be generalized to form a function of both contrast and spatial distance, namely $\tau_{cd}(c,d)$ as the sum of two independent components:

$$\tau_{cd}(c,d) = \tau_c(c) + \tau_d(d) \tag{4.1}$$

Here $\tau_c(c)$ is unmodified from Equation 2.2 although we are considering a fixed value of $c$, disabling the contrast dependence. The fixed value of $c$ of 100% contrast may be thought of a meta parameter that has no bearing on any of the results discussed in this chapter, until variation in $c$ is re-enabled. The key point is that a distance-dependent latency term, $\tau_d(d)$ has been introduced that is independent of contrast.

We will consider several possible versions of $\tau_d(d)$, starting with the absolute simplest, linear formulation, $\tau_{d1}(d)$:

$$\tau_{d1}(d) = d\sigma_L + \tau_L \tag{4.2}$$

To make this equation even simpler, we will fix $\tau_L$ to zero for now, reducing the new term to $\tau_{d1}(d) = d\sigma_L$. The variable $d$ is the radial distance from the particular SIRD model unit (once the units are assigned a spatial position) to the unit at the center of the response. The $\sigma_L$ value then expresses the strength of the spatial latency dependence.

As already noted, the exact form of the distance-latency dependence shown in Figure 4.5 is difficult to establish from the data. A linear relationship is the simplest first approximation although a gradient still required to set the value for $\sigma_L$.

Even if there were no variation in the response of all 21 cells recorded in Figure 4.5B, finding a suitable value for $\sigma_L$ would still be difficult for two reasons: (1) this experimental data is expressed in terms of eccentricity, and the spatial scale we have for the VSDI response shown in Figure 4.2A is expressed in millimeters, and (2) this data is recorded in cat and not macaque, making the appropriate relationship between spatial scales even more complicated.

This data is therefore useful to show the spatial-latency effect and to give some idea of the appropriate magnitude of the phenomenon. The values of $\sigma_L$ we will be considering are 0, 35 and 80 $\mathrm{ms\,mm^{-1}}$. These odd units (reciprocal speed) are convenient

with reference to Figure 4.5, helps ensure the expression in Equation 4.2 is in a particularly simple form, and allows this constant to be expressed in terms of millisecond integers instead of with awkward floating point values.

Looking at Figure 4.5 the value of 80 $\mathrm{ms\,mm^{-1}}$ may still appear rather high (i.e., corresponding to a very slow spatial propagation) even with the difficulties in interpreting this value in mind. Let us assume that apparent lateral propagation speed in macaque match those of cat. Figure 4.5C shows that the mode value of the speed based on the data shown in Figure 4.5B is around 0.1 $\mathrm{mm\,ms^{-1}}$. Then $\sigma_L = 80\mathrm{ms\,mm^{-1}}$ corresponds to $\frac{1}{80}$ $\mathrm{mm\,ms^{-1}}$ which is 8 times too slow. An explanation for why this value of $\sigma_L$ is still not unreasonable, is given in section 4.7.3.

Note that the additive constant $\tau_L$ may be used to express an additional delay in milliseconds that is currently set to zero. When $\tau_L$ is zero in this way, the central unit is a pure IRD unit that only has an offset due to latency scatter as $d = 0$ at the response center. This unit experiences no distance-dependent latency change.

In order to examine the effect of introducing $\tau_d(d)$ to the IRD model in the creation of the SIRD model, it is necessary to assign the IRD units a spatial position in order to enable the appropriate spatiotemporal analysis with respect to the VSD signal. This is the topic of the next section, after which it shown that this $\tau_{d1}(d)$ term, together with latency scatter, can begin to explain the spatiotemporal properties of the VSD signal.

#### A spatially sampling distribution of IRD units

In order to simulate the effect of $\tau_d(d)$, it is necessary to move from an unordered collection of IRD units to a model that explicitly incorporates the spatial extent of the neural population. The goal is to introduce a spatial component to the population of PSTH profiles in the simplest way possible that allows the current latency scatter mechanism to be combined with calibrated spatially dependent latency shift. In this section, both $\sigma_L$ and $\tau_L$ of Equation 4.2 are zero, ignoring the new spatially dependent term in the IRD model.

This can be achieved by arranging the unordered collection of IRD units we have considered into a two dimensional array and scaling the response amplitudes to match the approximately Gaussian profile observed in V1 as shown in 4.2A. All the simulations in the rest of this chapter make use of a regular spatial sampling of IRD units designed to approximate the cortical response over the same spatial scale as shown in 4.2A, with the response maximum located at the center of this array. The Gaussian profile used is shown in Figure 4.6A and its impact on the rescaled PSTHs is shown

for a regularly space subset of units shown in Figure 4.6B.

As already discussed, the IRD responses are generated using a fixed 100% contrast input before they are rescaled by the spatial Gaussian pattern shown in Figure 4.6A. The goal is to enable the same type of spatiotemporal analysis as shown in Figure 4.2A by roughly matching the spatial Gaussian profile to the position and size of the regions of interest used by Sit et al. (2009). The scale bar in Figure 4.2A will be used to determine the size of the simulated cortical area in millimeters which will impact the value of the spatial-latency constant $\sigma_L$.

The distance $d$ per IRD unit that is supplied to Equation 4.1 is computed as the Euclidean distance of a unit from the center of sheet. The origin is defined at the center and the sheet of units extends 5 millimeters in both directions, defining a simulated cortical area of $10 \times 10$ millimeters in which 500 regularly spaced model IRD units will be simulated.

This value was primarily chosen for convenience but it is also a good enough match to the area shown in Figure 4.2A which is $8 \times 8$ millimeters in size. This small discrepancy is not a major concern as there are more substantial difficulties involved in calibrating a suitable value of the spatial latency constant as discussed in the previous section.

With this spatial definition, the SIRD model can be analyzed using the exact same form of spatiotemporal analysis as used in the VSD signal. This is shown in Figure 4.6D where the applicable regions of interest are shown in Figure 4.6C, overlaid on top of the peak response after scatter. This latency scatter is applied to the IRD units in the same way as before as shown in Figure 4.3.

These plots in part C show the spatiotemporal analysis for the unscattered, scattered and experimental conditions. The difference between the top plot of Figure 4.6D and the middle plot is exactly the difference between an individual IRD unit and the population average.

As the effect of latency scatter is exactly the same as before, Figure 4.6 introduced no new mechanisms. What has changed is that a spatiotemporal analysis can now be applied and individual units have a distance $d$ defined to the center of the response, allowing the effect of Equation 4.2 to be investigated.

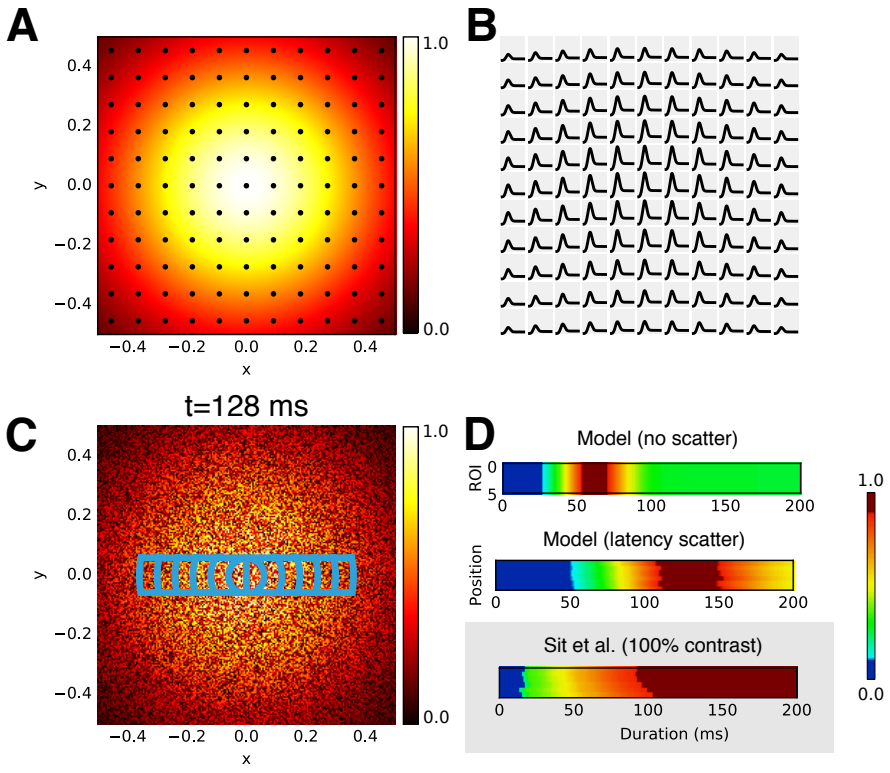


Figure 4.6: **Simple sampling grid of IRD model across cortical space.** (A) Gaussian profile used to scale the IRD unit responses across space, shown at the point of maximum response. Black points indicate the sample positions that scale the responses in B, drawn from the underlying $500 \times 500$ sampling grid used throughout this chapter. (B) PSTH profiles generated by the IRD model that are now given an explicit spatial positions in the cortex. Note that all these units have identical latencies as the response is scaled after the IRD model has generated a $100\%$ contrast profile, a restriction that is relaxed at the end of the chapter. (C) Regions of interest used to analyze the spatiotemporal dynamics of the response following the protocol used in Sit et al. (2009). The ROIs are overlaid on top of the peak mean model response after the application of latency scatter. (D) Spatiotemporal plots showing the model response (top) shown in (A), the scattered model response shown in (B) and the experimental data reproduced from Sit et al. (2009) (bottom), showing the spatiotemporal logistic fits for the 100% contrast condition. These modeling results are a straightforward spatial extension of those already presented in Figure 4.3 as there is no systematic latency variation across space.

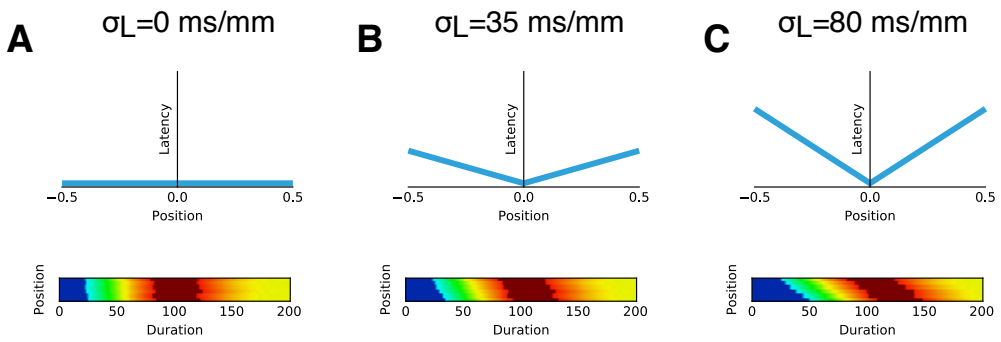


Figure 4.7: **Effect of changing the spatial constant $\sigma_L$ of the spatial latency dependence in the SIRD model defined using $\tau_{d1}(d)$** (A) When $\sigma_L = 0$, there is no change in latency as a function of position as shown in the schematic (top). This is exactly the same result as shown in the middle plot of Figure 4.6D shown zero spatiotemporal gradient (B) Example of a non-zero spatially dependent latency using $\sigma_L = 35$ $\mathrm{ms\,mm^{-1}}$. The plot on the bottom shows that the spatiotemporal gradient is now non-zero. (C) Increasing $\sigma_L$ to $80$ $\mathrm{ms\,mm^{-1}}$ further increases the spatiotemporal gradient. The qualitative effect of a non-zero spatial latency constant is clear, matching the non-zero spatiotemporal gradient in towards the end of the VSDI response seen in Figure 4.2B.

### 4.3.4 Simulating linear spatial latency dependence

Now the SIRD model has a notion of distance, it is time to investigate what happens when $\sigma_L$ is increase from zero, keeping $\tau_L = 0$. As was noted previously, when $\tau_L = 0$, Equation 4.2 ensures the central unit is always a pure IRD unit (with latency scatter) as $d = 0$ at the center. In addition, when $\sigma_L = 0$, the distance-dependent latency term is disabled which is why this constant will now be increased, starting at zero.

Figure 4.7 shows the results using $\sigma_L$ values of 0, 35, and 80 $\mathrm{ms\,mm^{-1}}$. As expected, when $\sigma_L$ is zero, the response properties are unchanged. As $\sigma_L$ increases, a new feature emerges as revealed by the spatiotemporal analysis as there is now a spatiotemporal gradient, reflecting the spatial dependency introduced by this term.

Although Figure 4.7 demonstrates a spatiotemporal dependence in the SIRD model for the first time, in some ways we are no closer to matching the VSDI response having gained an extra model parameter, $\sigma_L$. The reason is that we can still only match the spatiotemporal gradient at one point in the response shown in Figure 4.2B. Previously, the gradient matched at the onset where the spatiotemporal gradient is nearly vertical and now, with $\sigma_L$, we can still only match the gradient at only one specific point, anywhere along the response.

This is because the VSDI signal as shown in Figure 4.2B has a varying spatiotemporal gradient, starting with a low value at point (a) and increasing towards the peak marked at point (c). In other words, any constant spatiotemporal gradient will not be a good match to the VSDI response. This means that the next step is to understand why the spatiotemporal gradient shown in Figure 4.2B changes over the course of the response.

## 4.4 Diversity in latency response properties

The first step towards matching the VSD signal was to introduce diversity between the IRD units in the form of latency scatter. Equation 4.2 introduced a spatially dependent term that reflects how latency changes as the stimulus moves out of a cell's receptive field. There is still only one type of diversity in the model as this spatial term has been applied to all the units in exactly the same way.

This reflects another type of homogeneity across all the units in the model: they all vary in latency as a function of space in exactly the same way. As noted at the start of this chapter, the key to making progress when trying to understand the population response is to consider the different ways that neural responses may vary that are not accounted for when starting with a single-unit model.

Having all the latency of all units depend on spatial distance from the center of the response in exactly the same way is both unrealistic and explains why the spatiotemporal gradients in Figure 4.7 are constant instead of varying. This points to a new form of diversity, whereby all the units of the SIRD model do not all follow the same dependence between spatial position and latency. Note that a non-linear version of Equation 4.2 will not help, as this term only lets you define the shape of the spatiotemporal gradient across space and will not allow it to vary over time.

### 4.4.1 A simple, two population model

Having made these observations, let us try to introduce a second form of diversity in as simple a way as possible, by introducing $\tau_{d2}(d)$:

$$\tau_{d2}(d) = \boldsymbol{w}_i(d\sigma_L + \tau_L) \tag{4.3}$$

Here the inner term is exactly the same as before and the semantics of all existing symbols remains unchanged. What is new is the per unit, random variable $\boldsymbol{w}_i$ which

takes on the values $w_i = \{0,1\}$ with some probability weighting $p$. In other words, each unit has a probability $p$ of $w_i = 0$, in which case it behaves as a pure IRD unit and probability $1-p$ of being assigned $w_i = 1$, in which case it behaves as an IRD unit that is affected by the additional spatially dependent latency shift. We will denote the two populations as populations 0 and 1, according to the corresponding value of $w_i$.

This diversity means that some units of the population have a spatial-latency dependence and others do not. We will be considering $p = 0$, $p = 0.5$, and $p = 1$, such that the entire population consists of either unmodified IRD units, IRD units with a spatial latency dependence or an even split between these two types. With no way to weight the two populations, a 50% split using $p = 0.5$ is the simplest conceptual model to consider.

Although the probabilistic component is as simple as possible, Equation 4.3 could be made simpler by keeping $\tau_L = 0$. A small leap will be taken that will be shortly justified by setting $\tau_L$ to a non-zero value, specifically of 30 milliseconds. This means that the spatially dependent units all experience an additional, fixed delay.

Note that this possibility is *not* excluded by the data in Figure 4.5B as this data shows relative latency only. For instance, it is plausible that all the cells that do not vary in latency as a function of eccentricity have an earlier overall latency.

One possible objection is that a non-zero $\tau_L$ effectively adds an additional spread to the latency scatter mechanism that has already been calibrated. This would be an issue for large $\tau_L$ values but at 30 milliseconds, $\tau_L$ is relatively small compared to the latency spread calibrated against Nowak et al. (1995). In addition, a 30 milliseconds is plausible according to the anatomical interpretation presented in section 4.7.3.

Figure 4.8 shows the results of these three different probability values, using a $\sigma_L$ of 80 $\mathrm{ms\,mm^{-1}}$ and $\tau_L$ of 30 milliseconds. When $p = 0$, the results are the same as Figure 4.7A and the middle plot of Figure 4.6D. When $p = 1$, the results are the same as Figure 4.7C with an additional 30 millisecond delay. When $p = 0.5$, a new qualitative property emerges, matching the changing spatiotemporal gradient effect shown in Figure 4.2B.

Why does this happen? It is important to understand the causal ordering of the two signals and how they are reflected in the VSDI response. By definition, the fastest cells respond earliest, and they belong to the population that has spatially independent latencies. They will therefore be represented most at the onset of the signal, explaining the nearly vertical onset gradient. As time progresses, the slower cells belonging to the second, slower population start contributing to the response, with the cells with the

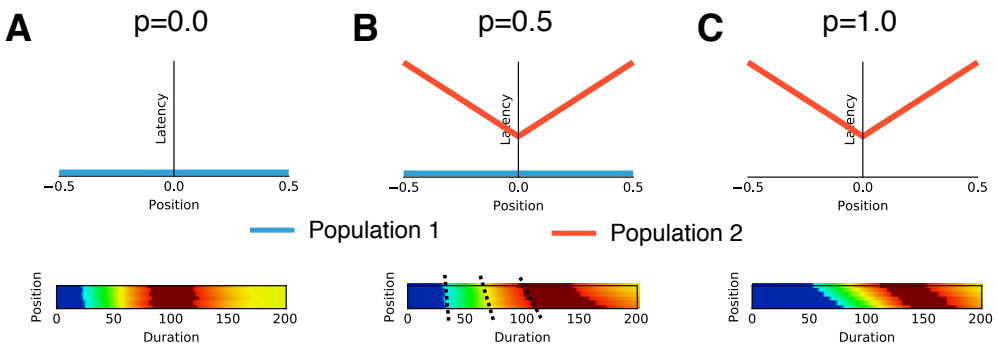


Figure 4.8: **Effect of probabilistic assigning units to one of two populations with different temporal response properties defined using $\tau_{d2}(d)$** (A) When $p = 0$, all units act as standard IRD units, as indicated by the lack of spatial dependence in the top schematic. There is zero spatiotemporal gradient and this plot matches Figure 4.7A and the middle plot of Figure 4.6D (B). When $p = 0.5$, mixing the two types of population responses, there is a qualitative match with the VSDI signal as the spatiotemporal gradient increases over the course of the response. (C) When $p = 1$ the entire population is delayed by $\tau_L = 30$ milliseconds and all units have spatially dependent latencies. There is now a constant, non-zero spatiotemporal gradient throughout the response.

lowest latency scatter appearing first. Eventually, the response is a reflection of the *averaged* response across *both* populations whereas the earliest onset over-represented the fastest cells.

It is important to note that when describing populations 0 and 1, we are referring to the population of IRD units in the SIRD model which have different response properties, *not* neural populations. The SIRD model continues to make no commitments regarding the origin of these different latency effects. All that is required is that there are two types of response component, one which is fast and spatially independent in latency and slower response that has a spatially dependent latency.

For instance, there could be one single population of neurons which has the spatially dependent response component mediated by lateral interactions that occur later, perhaps due to an additional synaptic delays (e.g. single-synaptic versus multi-synaptic responses). I.e., afferent input would drive the onset of the response, but the peak would not be reached until lateral input arrives later. Alternatively, these two populations in the SIRD model *might* correspond to anatomically distinct populations, such as cells in layer 4 vs. layer 2/3. This possibility is discussed in more detail in section 4.7.3. What matters is simply that the spatially dependent population is delayed, as will now be shown.

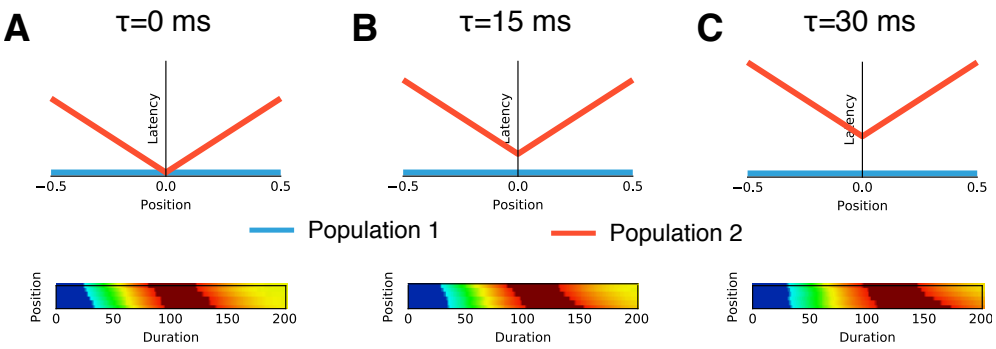


Figure 4.9: **Effect of increasing the $\tau_L$ delay between the two randomly selected populations with equal probability defined with $\tau_{d2}(d)$ and a $\sigma_L$ value of** 80 (A) When $\tau_L = 0$, there is a constant spatiotemporal gradient that is smaller than that when the entire population is affected by $\sigma_L = 80\ \mathrm{ms\,mm^{-1}}$ as shown in Figure 4.7 C. This is due to the averaging with the population that is not spatially dependent. (B) When $\tau_L = 15$ milliseconds there is evidence of a gradual spatiotemporal increase over the course of the response (C) When $\tau_L = 30$ milliseconds, there is a clear gradient shift over the course of the response as also shown in Figure 4.8B.

### 4.4.2 The effect of the inter-population delay $\tau_L$

Was it necessary to make $\tau_L$ non-zero? Figure 4.9 shows the effect of varying $\tau_L$ in order to show that the introduction of the 30 millisecond delay was a necessary step. Using the same $\sigma_L$ value of 80 $\mathrm{ms\,mm^{-1}}$ and a 50% split between the two response types, the spatiotemporal analysis is shown for $\tau_L$ values of 0, 15, and 30 milliseconds. When $\tau_L$ is zero, there is a fixed spatiotemporal gradient that is roughly half of the defined $\sigma_L$, due to the averaging process over a split population. As $\tau_L$ increases, the change in the gradient over the course of the response becomes more apparent.

At this point, the following aspects of the VSD response have been accounted for (1) the increase time constant of the population response (2) the coupling between spatial position in the cortex and latency (3) the way this dependence between spatial position and latency changes over the course of the response. There are still at least two things left to explain, starting with the long plateaus in the VSD signal, followed by the contrast-latency dependence.

## 4.5 Diversity in tuning properties

The SIRD model has still not accounted for all the properties of the VSD signal, namely the way the peak response plateaus and the latency shift with respect to different stimulus contrasts shown in Figure 4.2B. As this point, two types of diversity across the population of responses has resulted in two different, orthogonal ways to match the VSDI response. This suggests that some other important type of diversity has not yet been accounted for.

When the two stochastically defined populations of SIRD units was defined, it was done with reference to the spatial-latency dependence as captured by Equation 4.3. The spatial position of a stimulus with respect to a cell's receptive field is only one factor that might affect the response latency of a cell. In this section, we will consider the diversity of tuning in general to capture the relevant diversity of (nearly) everything else.

There are infinite possible features of a visual stimulus that could affect a neuron in the primary visual cortex, including such things as orientation, phase, spatial frequency, motion direction, motion speed, motion energy, color, disparity, net luminance, temporal frequency, and then the second- and higher-order statistics of all these properties such as orientation co-occurrence and so on. In any visual experiment, a small fraction of these features will be controlled for, and an even smaller number will be systematically varied.

The assumption is that the uncontrolled visual features will most likely wash out between any two experiments. Uncontrolled visual features may be the result of much unexplained variance in neural responses, but because these variables are uncontrolled, there is no reason to expect systematic biases although there are no guarantees that such biases may not accidentally occur.

This type of tuning-dependent latency variation for uncontrolled visual features lives in an undefined, abstract and high dimensional space. It is also already incorporated into the SIRD model via latency scatter, as the mechanistic origin of this scatter is uncontrolled. If there was some uncontrolled feature that explains some of this scatter, it is reasonable to assume it was just as similar in the laboratory of Nowak et al. as the laboratory of Albrecht et al. Therefore, what needs to be accounted for specifically are the stimulus features that were not just controlled, but explicitly optimized for each neuron recorded.

The spatial position of the stimulus with respect to the receptive field was one such

optimized parameter. The key remaining features are orientation, spatial frequency and phase which were all optimized by Albrecht et al. (2002) in order to evoke a clear response from the recorded cells. To make this discussion more concrete, let us briefly discuss orientation tuning.

Perhaps the most well-known feature about the neurons of the primary visual cortex is that they are orientation selective. Every single experimental protocol in all the data we have discussed so far has worked explicitly to find the optimal orientation for a particular cell in order to evoke a response. This is the correct approach when trying to quantify the properties of individual cells. When considering a population of cells, it is now necessary to account for all the cells for which the stimulus does *not* have the optimal orientation.

In macaque V1, as soon as a response spans more than one orientation hypercolumn in cortical space, there will be a cell with every possible orientation preference. This means that as soon as the VSD spatial response is over a millimeter, there will be plenty of cells that are not being driven by the orientation signal of the stimulus at all, for *any* stimulus. The key point is that it is likely that a poorly driven cell with an orthogonal orientation preference to the stimulus is likely to have a higher overall temporal latency.

At this point we could try to calibrate SIRD for orientation-tuning dependent latency shifts, then spatial-frequency tuning dependent latency shifts and so on. This is likely to be difficult, not only due to the lack of data but because these effects are likely to be orthogonal. The true distribution that is needed is the high dimensional orientation–phase–spatial-preference latency distribution that is far too high dimensional to sample effectively. For this reason, SIRD tries to use the simplest possible distribution; the real distribution is unknown.

### 4.5.1 Incorporating the tuning dependent latency shift

Two things need to be done to introduce tuning-latency diversity to the SIRD model. First, the tuning latency shift has to be hooked up to the IRD model and secondly, a distribution has to be chosen in the absence of experimental data.

Now that investigation of $\tau_d(d)$ term is complete, it will be deprecated. This is the form chosen to include the tuning dependent latency shift $\tau_T$:

$$\tau_{cd}(c,d) = \tau_c(c) + \boldsymbol{w}_i(\tau_L + \sigma_L d + \boldsymbol{\tau}_T) \tag{4.4}$$

It will be assumed that whatever the $\tau_L$ delay corresponds to, it is not specific to the

spatial latency effect, now modeled by $\sigma_L d$. It will also be assumed that the population that responds with spatially dependent latency shifts is also the population that is selective to the other tuning features captured by $\tau_T$. In other words, the population with spatially dependent latency shifts are also responsive to a diverse set of visual features.

This can be motivated by looking at the VSDI signal: whatever way we wish to explain how peaks become plateaus, it is clear this effect is important later in the response and is therefore a natural fit for population 1 (the slower population that has spatially dependent latency).

An interpretation of why $\tau_T$ only applies to this second population is given in section 4.7.3, and the SIRD model itself continues to be agnostic towards the detailed causal reasons for these two types of response. The two things that now need to be justified are (1) why is $\boldsymbol{\tau}_T$ introduced in an additive fashion and (2) why is it not parameterized by $c$ and $d$.

Both of these points can be justified by considering the orientation preference component of this distribution. The spread of orientation preferences across space in macaque V1 is roughly uniform across an area a few hypercolumns wide. The latency variation due to orientation tuning is therefore independent of $c$ and $d$ is and should therefore be introduced additively. This reasoning applies to the other orthogonal features such as spatial frequency preference and phase preference.

This is why $\boldsymbol{\tau}_T$ is unparameterized although conceptually, it could be parameterized by $T$, representing the some abstract distance in a high dimensional feature space. In practice this parameter is so abstract and impossible to define concretely that it is not useful to make this parameterization explicit.

### 4.5.2 Modeling an unknown tuning-latency distribution

The form of the $\boldsymbol{\tau}_T$ distribution is entirely unconstrained by experiment. As a result we have only two criteria (1) to use the simplest distribution to show the effect of this term in a very general way (2) the distribution must be positive as latencies are always positive in a casual system.

This leads naturally to the exponential distribution, which is both extremely simple and only defined for positive numbers:

$$\mathrm{Exp}(x;\frac{1}{\beta}) = \begin{cases} 0 & x < 0 \\ \frac{1}{\beta}\exp^{-\frac{1}{\beta}x} & x \geq 0 \end{cases} \tag{4.5}$$

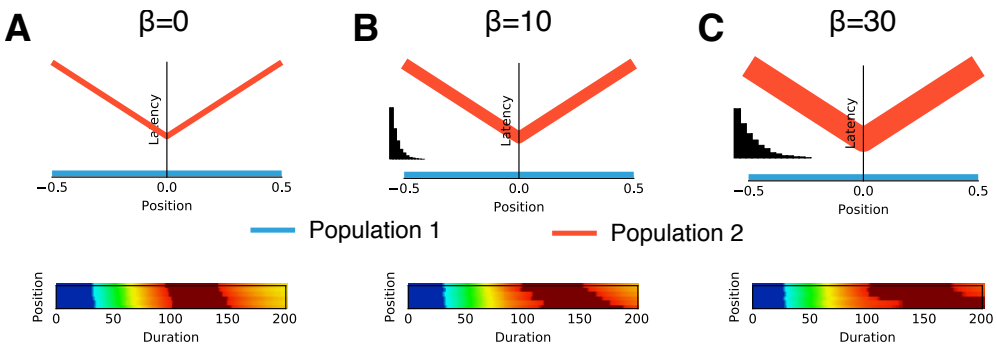


Figure 4.10: **Effect of increasing the β parameter of the exponential distribution used to introduce latency variation via $\tau_T$** (A) When β is asymptotically close to zero, $\tau_T$ introduces no variation and the results are unchanged ($\sigma_L = 80$ ms mm$^{-1}$, $\tau_L = 30$ ms) (B) The rust-colored area is extended when $\beta = 10$ milliseconds (C) When $\beta = 30$, the rust-colored area is greatly extended, showing that what was a clear peak is becoming a plateau, matching the VSD signal.

This expression can be further simplified by substituting $\lambda = \frac{1}{\beta}$, but the form shown above is more intuitive as the expectation (mean) of the exponential distribution is simply given by β. Larger values of β then correlate with a greater variation as opposed to the λ formulation which follows the inverse relation.

Figure 4.10 shows the effect of different values of β used in the exponential distribution used to approximate latency variation introduced by $\tau_T$. The values shown are β when asymptotically close to zero to introduce no variation (β is undefined at zero), $\beta = 10$ milliseconds, and $\beta = 30$ milliseconds.

The effect of increasing β is to increase the size of the rust-colored region, showing the peak gradually becoming a plateau. This is completes the demonstration that the effect of latency variation due to $\tau_T$, is going to be to extend peaks into plateaus, regardless of the details of the distribution.

The final unexplained property of the VSDI signal is already in the SIRD model; it has simply been disabled. This is the contrast-dependent latency shift native to the IRD model.

## 4.6 The contrast-dependent latency shift

All the way through this chapter, the $\tau_C$ term native to the original IRD model was disabled as all profiles were generated as 100% contrast profiles in the IRD model which were them rescaled in amplitude. This $\tau_C$ term expresses a contrast-dependent

latency shift which would have confounded the spatiotemporal analysis of the other mechanisms in the SIRD model.

The contrast variation of a stimulus across space coupled with $\tau_C$, the inverted Naka Rushton equation defined in Equation 2.2, suggests a different type of spatial latency shift that is contrast driven. Suppressing this effect by scaling the output of the IRD model given a 100% contrast input was necessary to avoid conflation the two spatial latency effects.

There is another reason contrast-dependent latency shifts are problematic. The SIRD model only models interactions within V1 and lacks a subcortical pathway, including a retina. As contrast is only defined at the level of the photoreceptors, the semantics of contrast become unclear in the SIRD model. Roughly, contrast will now refer to the net afferent drive to a cell as a result of a contrast signal. The second problem is that it is very difficult to relate contrast scales between experiments using different stimuli. The IRD model was calibrated using stationary sinusoidal gratings but the VSDI data was evoked using a Gabor stimulus.

Another problem is that the reason this effective contrast depends on space is not clear. It could be due to actual contrast changes on the retina, which are true for a Gabor stimulus, but it could also be related to reduced afferent drive as the stimulus moves out the cell's receptive field, conflating the issue with the existing spatially dependent latency term.

With this in mind, it is time to re-enable a term that was already in the IRD model to account for the last qualitative feature of the VSDI onsets that requires explanation. In Figure 4.2B, it can be seen that between the 100% and 25% conditions there is an overall shift in response latency, a feature that cannot be explained with the current version of the SIRD model.

Re-enabling the $\tau_C$ term offers a potential solution, as the lower the contrast value supplied to $\tau_C$, the slower the response. To do this, it is necessary to create a spatial pattern to act as *input* to the IRD units instead of simply scaling the response amplitudes in the output. This spatial pattern acts like varying contrast for the IRD model, but it can be considered the net afferent input to each cell, rather than as the contrast of an external stimulus.

Figure 4.11 shows the effect of re-enabling the $\tau_c(c)$ term of the IRD model, by applying the Gaussian spatial profile to the *input* of the IRD units instead of simply rescaling their output for the $c = 100$ (percent contrast). As the definition of contrast in this context is so nebulous, the details of this "effective contrast" spatial profile

are unknown. In Figure 4.11A, the same Gaussian profile as shown in Figure 4.6A is used and in B, this spatial profile is offset by 20% reducing the contrast change across space. What is clear is that as the *overall* stimulus contrast changes, the inverted Naka-Rushton equation expressed by $\tau_c(c)$ in Equation 2.2 predicts that lower contrast stimuli will have a delayed response relative to high contrast stimuli, matching what is observed in the VSD data as shown in Figure 4.2B.

It is also important to note that the response amplitudes will also have changed as they are now determined by the IRD model, which saturates at high contrast due to Equation 2.3. These differences are not easily visible in the spatiotemporal plots shown in Figure 4.11 as the trace for each ROI has been individually normalized. At this point, the SIRD model would need a more detailed subcortical pathway that would require the development of a more complete mechanistic model of the visual system, starting at the photoreceptors of the retina. As the rest of this thesis focuses on building such a model in general, we will not further attempt to make the SIRD model fully mechanistic.

In any case, this final mechanism completes the account of spatiotemporal VSDI onset response properties by demonstrating latency shifts as a function of overall stimulus contrast. This mechanism also predicts that the VSDI onset gradient is not vertical, in as much the effective afferent contrast input varies across space. This spatial profile is not known but a perfectly vertical onset gradient indicates no such spatial-contrast-latency variation.

## 4.7 Discussion

The SIRD model has been formulated is a very specific way, connecting up different forms of experimental calibration within a very simple mathematical framework. The goal till now has been to explain the VSD signal in terms of what is observed in a way that is as agnostic to the detailed anatomy and mechanistic explanation.

In this section, the whole model is summarized and the interpretation of the model is discussed. One minor, optional extension to the model is described followed by a discussion of the limitations of the model.

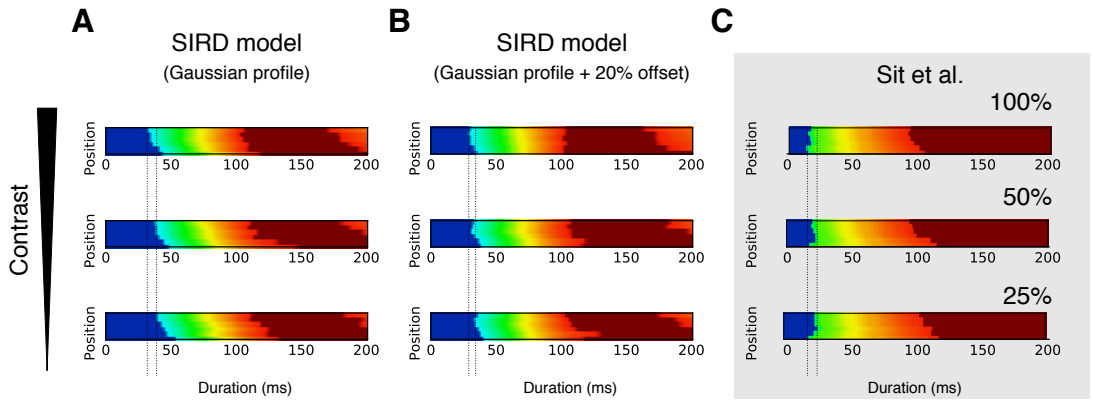


Figure 4.11: **Behavior of the SIRD model as a function of stimulus contrast when $\tau_C$ is not locked to a $100\%$ contrast latency**. Response of the SIRD model as a function of contrast for two different spatial input profiles in relation to the experimentally recorded VSDI signal in macaque V1 Sit et al. (2009) (A) Behavior of the SIRD model when the Gaussian spatial profile shown in Figure 4.6A is used as input to drive the IRD units instead of using it to rescaling the $100\%$ contrast IRD profiles (B) Behavior of the SIRD model as a function of three different contrast levels when this profile is given as a fixed $20\%$ offset. As the relative variation of the afferent drive across space has been reduced, the spatiotemporal gradient due to the contrast-dependent latency shift is decreased. No specific contrast values have been assigned due to the difficulties of relating contrasts between different experiments using different stimuli (C) Experimental results when using $100\%$, $50\%$, and $25\%$ contrast conditions in the Sit et al. (2009) stimulus protocol. Note that the model now reproduces the main features of the VSDI imaging results, including the delayed peak relative to the IRD model, the increased spatial gradient for the peak, the wide plateau at the peak the increased spatial gradient at the peak compared to the onset, and now the longer latencies for low-contrast stimuli. Note that these three experimental plots do not show the raw data, but instead present the best logistic function fit to the time taken to reach the $10\%, 50\%$, and $90\%$ response levels within each ROI. Applying the logistic fits to the SIRD model output would further improve the match with the experimental data shown.

### 4.7.1 Model summary

The entire model can now be summarized by one equation where the latency scatter distribution $\boldsymbol{\tau}_S$ and compensation shift $k_S$ can be folded into the definition as $(\boldsymbol{\tau}_S - k_S)$. The original $\tau_c(c)$ term of the original IRD model is augmented with $\tau_{cd}(c,d)$ as follows:

$$\tau_{cd}(c,d) = \overbrace{\underbrace{\tau_c(c)}_{\text{Contrast shift}}}^{\text{IRD Model}} + \overbrace{(\boldsymbol{\tau}_S - k_S)}^{\text{Latency scatter}} + \underbrace{\boldsymbol{w}_i}_{\text{Weighting spread}} + (\overbrace{\tau_L}^{\text{Delay}} + \underbrace{\sigma_L d}_{\text{Spatial shift}} + \overbrace{\boldsymbol{\tau}_T}^{\text{Tuning spread}}) \tag{4.6}$$

The constants are $\tau_L = 30\text{ms}$, $\sigma_L = 80\text{ms}\,\text{mm}^{-1}$, and $k_S = 30\text{ms}$. Here $\boldsymbol{\tau}_S$ is a Gaussian distribution, $\mathcal{N}(\mu,\sigma)$ with $\mu = 60\text{ms}$ and $\sigma = 30\text{ms}$ that is clipped into the range $[0,120]\text{ms}$, $\boldsymbol{w}_i$ is a Bernoulli distribution, $p = 0.5$ and $\boldsymbol{\tau}_T$ is an exponential distribution, $\exp(\beta)$, with $\beta = 30\text{ms}$. The only other component is the spatial Gaussian input used to drive the $500 \times 500$ units over a $10 \times 10$ millimeter cortical area, used to define $d$ which is the distance of the unit from the center of the sheet.

### 4.7.2 Lateral interactions

Cortical neurons have lateral as well as afferent inputs. Lateral connectivity and the finite speed of lateral propagation is a natural mechanism to link the responses of cortical units across cortical space. There are several reasons lateral connectivity may be particularly important to understanding the VSD signal. Firstly, lateral connectivity is more superficial and therefore more likely to be represented in the signal. If this layer is heavily influenced by lateral interactions, then this is likely to be reflected in the signal, either because the neural response at the cell bodies are modulated by lateral interactions or because the voltage sensitive dye binds to the lateral connectivity itself.

The second reason lateral connectivity is relevant is that the VSD signal records activity over a large cortical area and will pick up the response from all cells, whether the response is afferent driven or laterally driven. This consideration establishes an important link between model definition and the VSD signal.

The hypothesis of Bringuier et al. (1999) is that peripheral subthreshold depolarizing responses are relayed by horizontal connections within area 17 of cat primary visual cortex. As the eccentricity of the stimulus increases from the receptive field center, the stimulus will eventually fall out of the classical receptive field which means

that if the cell continues to respond, it is must be driven by the extra-classical receptive field. In this regime, lateral interactions become particularly important and the hypothesis is that the finite speed of lateral propagation may explain the data shown in Figure 4.5. The idea is that when the stimulus is further from the receptive field center, the cell will reach its peak response only once lateral interactions reach it, which are relatively slow because they need time to propagate over cortical distance.

In the VSDI signal, there is bound to be a mix of contributions from cells driven from within their classical receptive fields as well as from outside it. What really matters is that the vast majority of cells in layer 2/3 will be strongly represented in the VSD signal because they are more superficial and nearly all these cells will receive lateral inputs, regardless of the ratio of afferent to lateral drive. Assuming this basic hypothesis, what Figure 4.5 then shows is how lateral interactions affect latency across space in a way relevant to the VSD signal.

Assuming this interpretation, the spatial-dependent latency term, $\sigma_L d$ is implicitly capturing properties related to lateral connectivity, with the value of $\sigma_L$ being related to the average speed of lateral propagation. If this is true, the SIRD model only captures the coarse features of lateral connectivity which in reality is structured and patchy.

Lastly, the lateral interactions may be involved in explaining the $\tau_L$ constant that defines the delay between the two populations of response. For instance, there could be an additive delay due to the multi-synaptic nature of lateral interactions although the time constants involved make this explanation implausible as the $\tau_L$ value used in the SIRD model is 30 milliseconds. A more plausible account for this delay is described in the next section, where laminar differences are considered.

### 4.7.3 Laminar organization

Structurally, the afferent cortical input generally arrives in layer 4, and long-range lateral connectivity is mostly confined to other, more superficial layers, such as layer 2/3. This anatomical structure suggests a natural explanation for Equation 4.3 even though it was not formulated with reference to laminar differences.

The basic idea is that the non spatially dependent response corresponds to the earlier input layers of the cortex, starting with layer 4 where the cortex first receives afferent input. This population of cells in the cortex then corresponds to the fast responding population 0 of the SIRD model.

In addition, there is little lateral connectivity in layer 4 which explains why these

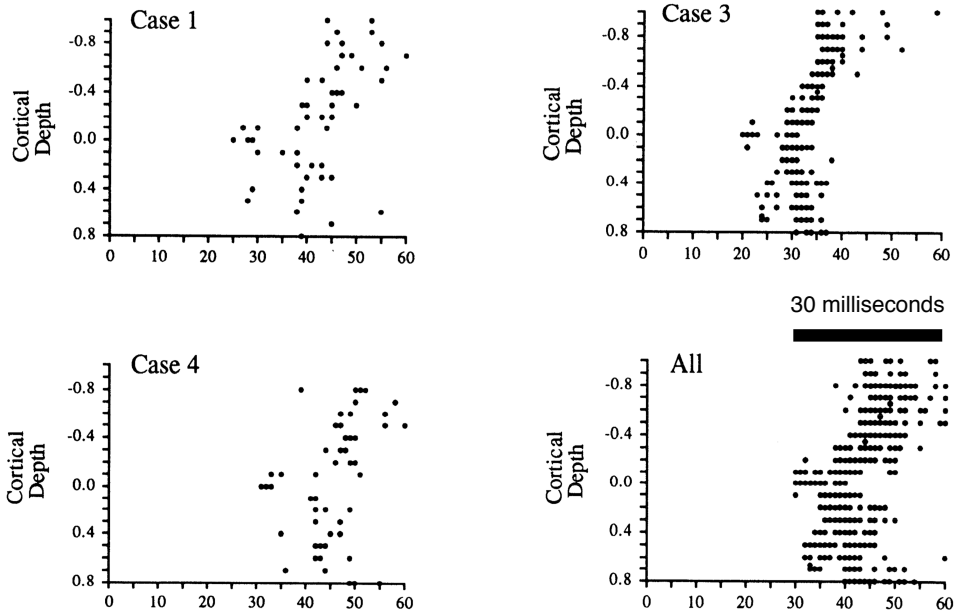


Figure 4.12: **Scatter plots of latency as a function of cortical depth in macaque V1 showing multiunit recordings in three cases**. Zero depth was physiologically determined layer 4C and negative values indicate more superficial layers. The condition marked "All" combines the three cases by aligning the earliest latency observed in each animal with 30 milliseconds. As expected, shortest latencies are in the geniculate-recipient layers and latencies become longer as cortical depth decreases. Additional scale bar shows that a $\tau_L$ value of 30 milliseconds is plausible given these results. Data reproduced from Figure 12 of Maunsell and Gibson (1992).

afferently driven neurons all respond with similar latencies. The more superficial cells in layer 2/3 will only receive afferent input after a delay and these are the cells that have horizontal connections that would explain the spatially dependent component to the response. This would mean that population 1 of the SIRD model corresponds to layer 2/3 cells that are not strongly driven enough by their afferent inputs to respond immediately, and that have to wait for laterally mediated depolarization before they start to respond.

The 30 millisecond gap between the two populations is also plausible in the context of this laminar explanation. Figure 4.12 shows latency as a function of cortical depth presented in Maunsell and Gibson (1992). This figure shows that a 30 millisecond spread between the geniculate-recipient layers and the superficial layers is reasonable.

The last thing that can be explained by the laminar interpretation is why the value of 80 $\mathrm{ms\,mm^{-1}}$ is so high in the SIRD model, corresponding to a propagation speed

around 8 times slower than the mode speed in Figure 4.5C. The stated $\sigma_L$ suggests it takes 80 milliseconds to traverse one millimeter of cortex is very slow given that the data suggests it should take around 10 milliseconds. The key is to realize that the high value reflects the simplified, even weighting between the two populations which in this interpretation means both populations contribute to the signal equally.

There is in fact a tradeoff between the value of $\sigma_L$ required in the SIRD model to account for the VSD signal and the weighting of the two types of response, determined by the probability $p = 0.5$. This effect was first noticed in Figure 4.9A where the spatiotemporal gradient was observed to be decreased relative to the response for the same $\sigma_L$ value ($80\text{ms}\,\text{mm}^{-1}$) when it was applied to the entire population. In other words, increasing the weighting given to the non-spatially dependent population reduces the spatiotemporal gradient for any chosen value of $\sigma_L$.

Given a lack of information about how strongly the two populations are represented, a value of $p = 0.5$ is used, weighting both populations equally. In the laminar account of the model, it is clear that the fast, spatially independent population should be weighted far *more* heavily than the slower population, allowing for a lower, more realistic value of $\sigma_L$. This is because layer 4 is not only deeper, making it more difficult for the photonic signal to emerge from the tissue, but also because less voltage-sensitive dye will be unable to bind to very deep cells.

This fact that in the SIRD model, splitting the population of units in two halves the spatiotemporal gradient suggests that to achieve a suitable spatiotemporal gradient, the equation $\sigma_L \times p = 40$ needs to be satisfied, which means the approximate biophysical value for lateral propagation in the SIRD model is $\sigma_L = 40\ \text{ms}\,\text{mm}^{-1}$. This is closer to the mode value, as shown by the speed value of $\sigma_L = 10\ \text{ms}\,\text{mm}^{-1}$ in Figure 4.5C but this still is not weighted correctly. The SIRD model has a very simple averaging process that ignored many important details. In reality cortical cells are not perfectly segregated into two neat layers, there are many intermediate cells, there is a variable signal loss as a function of depth depending on how well photons can exit the tissue, less voltage sensitive dye will bind to deeper layers and so on.

The last thing the laminar account can explain is why the effect of $\boldsymbol{\tau}_T$ is applied to the second population only. Firstly, any effect that occurs in the second population will be more strongly represented in the VSD signal if the correspond cells are in a more superficial layer. Secondly, $\boldsymbol{\tau}_T$ will be a more important term for neurons that are selective, i.e., that have narrower tuning curves. There is some evidence that neurons of layer 4 are less selective than those of layer 2/3, at least where orientation selectivity

is concerned (Ringach et al., 2002; Gur et al., 2005; Snodderly and Gur, 1995).

Additionally, the "All" case of Figure 4.12 indicates that the latency spread in the more superficial layers is greater than at cortical depth zero which corresponds to layer 4C. This matches the idea that there is additional, tuning-dependent latency variation in the second population of the SIRD model and that this population corresponds to layer 2/3.

In conclusion, the laminar account may be able to explain several features of the SIRD model (1) the reason for the delay between the two populations (2) that the value of this delay is reasonable (2) why it is the slower population that is associated with the spatially dependent latency shift (3) why the $\sigma_L$ value needed to match the VSD response does not correspond directly to lateral propagation speed and that a high $\sigma_L$ value is reasonable (4) why the $\boldsymbol{\tau}_T$ distribution is applied to the slower population.

### 4.7.4 Photonic Scatter

One other mechanism was tested and calibrated for SIRD, and it does make the simulated response slightly more realistic, but it was not of sufficient value to include in the final model. This mechanism is photonic scatter, implemented as a simple Gaussian blurring step applied to the sheet of units.

What photonic scatter is designed to simulate is the spatial blurring of the signal as the photons exit the neural tissue. This smoothes out the spatial response profile and helps eliminate the spatial noise effect that results from latency scatter as seen in Figure 4.6C. If photonic scatter had been applied, 4.6C would have been smoothed out to look more like Figure 4.2A.

It is worth emphasizing that the noisy appearance of Figure 4.2A is only visible as a consequence of the low-density spatial sampling we are using, and is not something that would appear in either a high density SIRD model or in the real experimental signal. In a population of real cortical neurons, the spatial response profile would be smooth due to the latency scatter of inputs across cortical depth as well as the high spatial density of cells in the neural tissue, even without the introduction of photonic scatter.

Unlike the latency scatter, the spatial blurring when the photonic scattering mechanism was added smooths the signal as a smooth Gaussian kernel was used. Knowing the spatial scale of the $10 \times 10$ SIRD model sheet, it was straightforward to find the appropriate kernel size, knowing that the blur is approximately 200μm in size (Orbach

and Cohen, 1983).

Although the photonic scatter smoothed the spatial signal, it only smoothed the spatiotemporal plots a little, by spreading response signal between the ROIs. As these plots were the main tool used in the analysis of the SIRD model, it was decided that although the photonic scatter works and helps smooth the signal a little, it was not worth introducing as another mechanism to the model.

### 4.7.5 Computational cost of the model

At a density of 500 there are 250000 IRD units in the SIRD model simulations shown. Each unit samples the IRD model at a particular position in order to simulate a 200 millisecond response.

Even though a simple IRD profile takes around 2.5 milliseconds to compute, it requires around 10 minutes to generate a quarter of a million PSTH profiles. Once the various random distributions are also introduced to the SIRD model, there is also a lot of random number generation involved which is also computationally expensive.

As a result, it takes around 15 minutes to run a SIRD simulation and analyze and visualize the output using the HoloViews tool described in Chapter 3. If faster performance were required, there is certainly plenty of scope for aggressive performance optimization and numerical vectorization. What is clear is that the computational cost of an equivalent spiking simulation would be much higher, and quite possible prohibitive.

### 4.7.6 Limitations

In this section, the limitations of the SIRD model are acknowledged. Some of these limitations are useful pointers to future work that could be done based on the SIRD model.

#### Fixed firing rate profile shapes

One of the most obvious limitations of the SIRD model may also be seen as one of its most surprising strengths. The VSDI signal is a very indirect signal of population activity but the voltage sensitive dye itself serves to reflect membrane voltage, not firing rates.

In this sense, it is perhaps surprising that the SIRD model can get approximate VSD signal onsets using plausible biophysical parameters while only considering the firing

rates of the IRD model. This may reflect the fact that it is the structure of the tissue, in terms of laminar organization and the various causal delays between signals that are dominates the population response and not the difference between spiking activity and membrane voltage.

The main use of the IRD model is to supply the response shapes, defined by Equation 2.1) and then to scale the response as a function of contrast. It is possible that the equations of the IRD model, namely the Naka Rushton equation for response saturation and the inverted Naka Rushton equation for the contrast-dependent latency shift would be just as applicable to a the membrane voltage template instead of a firing rate template.

Perhaps the main reason the SIRD model works is because all the mechanisms introduced apply to membrane voltages as well as firing rates. The spatially dependent latency shift used membrane voltage recordings directly, so at least this mechanism is appropriate, even though the data was obtained in cat and not macaque.

The concept of latency scatter is applicable when considering the membrane voltage, as is the idea that the diversity of tuning may impact latency. The form of the equations as well as the laminar explanation are also applicable whether membrane voltage or firing rates are considered.

A more important problem with the SIRD model may be that the shape of the response is based on the IRD model that applies for optimal stimuli. Many of the latency shifts that have been introduced, including the spatial-latency term and $\boldsymbol{\tau}_T$ correspond to the responses to non-optimal stimuli. One clear way to improve the SIRD model would be to investigate how the response template given by 2.1 changes when the stimulus is sub-optimal.

The firing rate representation of the SIRD model also has advantages. Not only firing rate profiles nicely modeled buy the IRD model for single units (and for appropriate stimuli) but they also allow the SIRD model to connect with other firing rate models, such as developmental map models. This is the direction taken in the next two chapters of the thesis.

#### Unknown dependence on tuning

Of all the parameters of the SIRD model, $\boldsymbol{\tau}_T$ is by far the least constrained. This simple exponential distribution served only to indicate the qualitative effects this sort diversity has on the VSDI signal as the true distribution of $\boldsymbol{\tau}_T$ is unknown.

What $\boldsymbol{\tau}_T$ represents are all the unknown dependencies between onset latency and

various visual features, namely orientation, spatial frequency and phase. Trying to determine the appropriate form of $\boldsymbol{\tau}_T$ would be an interesting experimental challenge, if it is possible to measure the properties of cells in such a high dimensional feature space. This is suggested as future work in section 7.5.2.

#### Lack of detailed mechanism

Even though the SIRD model discusses the tuning of cells, none of the units actually have their own defined tuning properties. This is because the SIRD model may be described as mechanistic only at the cortical level and it lacks a subcortical pathway. In addition, a lot of the interpretation of the model is related to laminar structure which it lacks.

As a consequence, it would be useful to have a more mechanistic model that can incorporate these features. A model with specific lateral connectivity, different cortical layer, spatially structured receptive fields and varied tuning properties across the population would be able to account for all the components of the SIRD model is greater detail. Some suggestions for how this could be achieved are described in the discussion chapter in section 7.5.11.

A step towards such a model is presented in the last chapter of this thesis by making a link from a more complete, mechanistic, and developmental model to the SIRD model. The idea is to bring all the calibration embodied by the SIRD model into to mechanistic developmental, which is one of the core contributions of this thesis.

## 4.8 Conclusion

The SIRD model is a spatial extension of the well-validated IRD model that is able to account for qualitative and quantitative features of the spatiotemporal VSDI response. This extension is packed into a one line equation that extends a single term of the IRD model that concerning latency shifts. The key feature of the SIRD model is that it accounts for diversity in neural responses that are often missed when recording from single neurons with optimized stimuli, and it explains each qualitative property of the VSD signal in terms of these different forms of response diversity.

Although the SIRD model has a very simple mathematical definition, justifying the form of this equation has involved many careful calibration steps and lots of detailed discussion. It has synthesized the data of three different single-unit experiments

together into a single framework that can explain the population response.

Even though the SIRD model does not have a subcortical pathway, all the mechanisms have a clear mechanistic role and the equations may be understood with reference to the anatomy. This makes it the first, mechanistic account of the spatiotemporal properties of the VSDI signal.

Lastly, as the SIRD model is based on firing rates, it links naturally to other firing rate models such as developmental map models. This suggests that the origins of the various forms of diversity requires by the SIRD model may be understood in terms of the developmental process e.g. in terms of specific lateral connectivity. In particular, the SIRD model serves as a reference for the spatiotemporal developmental model to be presented in Chapter 6.

# Chapter 5

# Quantifying the dynamics of cortical development

The ultimate goal of this thesis is to build a mechanistic model of visual response dynamics in the primary visual cortex. The first step towards this goal was taken in the previous chapter, where it was shown how the observed evoked response at the single-unit level relates to the evoked response observed at the population level observed as recorded with voltage-sensitive–dye imaging.

Although the spatially extended, invariant response descriptive (SIRD) model bridges between spatial scales and across experimental techniques, it is not a mechanistic model of the visual system. Instead, it is a description of how the firing-rate responses of individual cortical neurons relate to the dynamics at the population level. In order to account for the origin of the single-unit cortical response at a mechanistic level, it is necessary to follow afferent activity back through to the lateral geniculate nucleus and back to image on the retina.

Each stage of visual processing starting from the image on the retina has a temporal response and involves the spatial integration of activity. The neurons of the LGN have their own temporal responses coupled with corresponding spatial responses determined by center-surround receptive fields. Cortical neurons, have their own spatiotemporal response, some component of which is inherited from the dynamics of the LGN. Cortical neurons have their own diverse, orientation-selective receptive-field structure that integrates spatially across the LGN. The afferent dynamics of the evoked response in V1 is a property of the entire visual system that is conveying activity from the retina.

Each stage in the feed forward pathway is causal and the spatiotemporal activity dynamics at each stage is a crucial factor in determining the dynamics of the response

downstream. In order to decompose the evoked activity into its individual mechanistic components and identify each causal contribution to the evoked response, it is necessary to trace activity through the visual pathway, from the retina, through the LGN and to V1.

Building a model that directly incorporates the necessary connectivity and diversity in receptive fields throughout the early visual pathway is a daunting task. The available connectivity data is sparse and there is insufficient data to properly constrain the connectivity of a mature visual system. One way to bypass these difficulties is to work with a developmental model that starts in a simple, randomized state and that acquires the necessary diversity in connectivity and receptive-field structure over time.

Developmental self-organizing map models include all the key stages required to build a mechanistic evoked response model, starting with activity at the photoreceptor level. These developmental models have a simple initial condition and self-organize via Hebbian learning, developing diverse receptive-field structure that is organized into spatially extended feature maps. Once the process of development is complete, these models include specific lateral connectivity, orientation selective receptive fields, contrast invariant tuning curves and all the other essential features necessary to construct a mechanistic, evoked response model.

There is one other, more fundamental reason why developmental timescales are relevant to the evoked response. Only through development does cortical structure come into correspondence with the visual environment. In the SIRD model, firing-rate profiles were fixed but real neurons are not static entities. The evoked response changes over time as a function of the synaptic and network structure which is plastic and shaped by the entire history of previous evoked responses. There is a reciprocal connection between long and short timescales and it is this connection that allows an evoked response to encode information about the visual world.

In this chapter, a developmental, self-organizing map model will be analyzed in terms of the development of orientation-selective cortical receptive fields that are organized into smooth orientation maps. A new map quality metric will be used to quantify the biological plausibility of orientation map formation throughout development in order to validate the model. This model will then form the basis of the mechanistic spatiotemporal response model that bridges across spatial and temporal scales within a single, consistent framework, as described in the next chapter.

## 5.1 Activity dynamics and development

The SIRD model of the last chapter made use of the invariant response descriptive model of cortical firing-rate profiles to try to understand the voltage-sensitive–dye imaging signal. There are many possible ways such a model could be extended, making it important to highlight the properties of developmental map models that are particularly relevant for understanding the properties of the evoked response.

The IRD model is a direct description of firing rate at the V1 level and does not explicitly account for feature selectivity or the receptive field of neurons and it does not include any details regarding the subcortical visual pathway. Real evoked responses in the cortex are the result of complex, mechanistic interactions as the activity propagates from the retina to V1 via the LGN.

Any reasonably complete mechanistic account of activity in V1 needs to start with the projection of a visual stimulus onto a simulated retina. The retina then performs various forms of visual processing before the activity is projected to the lateral geniculate nucleus via the retinal ganglion cells. There, cells are observed to have ON and OFF center-surround receptive fields which then project to cortical neurons which typically have orientation-selective receptive fields.

A large, sophisticated compartmental model extending the SIRD model would be able to include all of these features in great detail whereas the developmental models we will be considering include far less detail. The key advantage of developmental models when it comes to understanding evoked response dynamics is their ability to address the effect of activity-driven plasticity on long time scales alongside receptive-field and feature map formation (Bednar, 2012).

The receptive fields of cortical neurons have been known to be orientation selective since the work of Hubel and Wiesel over 50 years (Hubel and Wiesel, 1959). These orientation preference are smoothly organized across the cortical surface in many primate species, as shown by the orientation map in Figure 2.3B. This spatial, orientation preference structure is crucial to understanding the evoked response as a receptive field represents the relationship between some property of a visual stimulus and the neural response.

Self-organizing developmental map models demonstrate receptive-field formation and orientation map formation, expressed in terms of activity driven changes to synaptic strength. These changes occur very slowly and given that receptive-field structure is an important component in determining the evoked response, it is natural to make use

of such developmental models when trying to account for the spatiotemporal dynamics of evoked activity across timescales. For an overview of the self-organizing network models we will be considering, see section 2.5.

Calibrating and validating developmental self-organizing models poses some unique challenges, most notably due to the difficulty in reasoning about the model behavior at the end of development given only a specification of the initial conditions and training stimuli. Although there are many different measurements that can be performed at the end of development, such as establishing whether or not the model exhibits contrast-invariant orientation tuning, the most striking outcome of the development process is an orientation preference map.

Measuring orientation preference and selectivity maps is relatively straightforward using the standard vector averaging method (Miikkulainen et al., 2005; Blasdel and Salama, 1986) whereas quantitatively establishing the plausibility of these maps is much more difficult. Although selectivity and stability measurements of an orientation map are useful indicators, they are insufficient to gauge map quality. An additional map quality metric is required which is introduced in this chapter to evaluate the plausibility of the spatial organization of orientation preference.

The GCAL model (Stevens et al., 2013b) is the developmental model that will be analyzed in this chapter. This model was the product of several years of collaboration prior to me carrying out the novel map quality analysis presented here, leading to the publication with Judith S. Law, Jan Antolik, and James A. Bednar. The suite of metrics described in this chapter were used to show that GCAL features realistic map development and the GCAL publication is included in the Appendix.

## 5.2 Evaluating orientation map quality

An experienced neuroscientist who has inspected many different experimentally recorded orientation maps will eventually develop a set of subjective heuristics regarding what a smooth orientation map should look like. In a similar way, a computational modeler who has examined the output of developmental simulations across many different runs will learn to judge relative stability and robustness between developmental models. Although these types of informal heuristics are useful in the research process, they are unsuitable for communicating results to the wider scientific community.

In the absence of objective metrics, it is sometimes permissible to publish a small number of results and claim that these examples are representative of the whole. This

approach tends to be problematic in that the selection process itself is a subjective judgment that may in turn be biased. To show that any set of supposedly representative examples is in fact unbiased, an appropriate objective metric is once again required.

Well-defined map metrics are therefore needed by both experimenters and modelers, with many different measures proposed in the literature. The way a metric is used is an important consideration, as both experimenters and modelers may use the same metric to achieve very different goals. An experimental scientist will typically use a metric to summarize data recorded from a live animal. In contrast, the aim of a computational modelers is to show that a mathematical abstraction captures the relevant properties of the biological system under study. Unlike the experimenter, a modeler is concerned whether a simulated orientation map is biologically plausible.

We will start by defining selectivity and stability measures, which are valuable, well-defined quantities, and yet we will show without an additional measure of the map's spatial organization they cannot rigorously constrain map quality. Existing metrics that are influenced by the spatial arrangement of orientation preferences are then considered and shown to impose only very weak constraints on map structure. Finally, a new map quality metric based on $\pi$ pinwheel density is introduced which not only imposes strong constraints on what constitutes a realistic map but does so in a way that is species independent.

### 5.2.1 The need for a new map quality metric

Two of the simplest metrics for evaluating the development of simulated orientation maps in relation to the experimental data are (1) the average orientation selectivity and (2) the map stability index. These two measures are now described before it is shown why they are insufficient for assessing the plausibility of a map without also including (3) a map quality metric.

#### Selectivity and stability

The overall selectivity of an orientation map is simply the mean selectivity across all the elements of the map, where an element may be a pixel in an experimental recording or a computational unit in a model. The orientation selectivity is defined using the standard vector averaging method (Miikkulainen et al., 2005; Blasdel and Salama, 1986) as the magnitude of the summed vector.

The orientation map similarity index for evaluating stability is described in section

2.4.1. The similarity index ranges from zero to unity where a value of zero similarity indicates *anti*correlation and a value of one indicates a perfect correlation. As two random maps will typically be uncorrelated, a closely related measure is introduced called the *stability index*. This metric is a trivial rescaling of the similarity index metric such that zero on the stability index scale indicates a lack of correlation between maps instead of an anticorrelation.

The stability index (SI) measures how closely an orientation preference map $O$ at a particular stage of developmental process resembles the final available orientation map recording $F$ obtained at the end of development. The mathematical definition is expressed as the following sum over $n$ units indexed by $i$:

$$SI = 1 - \frac{4}{n\pi} \sum_i \left| (F_i - O_i) \mathrm{mod} \left( \frac{\pi}{2} \right) \right| \tag{5.1}$$

The chronic experimental recordings in ferret V1 of Chapman et al. (1996) show that orientation selectivity increases as the animal matures but the map structure is stable. That is to say that the preference organization across the cortical surface does not change after it initially emerges, resulting in a high SI value throughout development. These results are summarized in the Background chapter in section 2.4.1.

#### The need for a spatially sensitive metric

Both stability and orientation selectivity are defined at the level of individual units and map stabilities and selectivities found by computing the average over all units. This means that these two metrics are locally defined measures that are able to assess whether a simulated map has suitable properties *on average* but they are insensitive to the spatial properties of the map. In other words, these measures do not take into account the way the orientation preference might change from one unit to the next.

This problem can be illustrated by considering the differences between a salt-and-pepper organization typical in rodent V1, such as the one shown in Figure 2.3A with the smooth, structured orientation maps observed in primate V1, such as the one shown in Figure 2.3B. These two types of map pattern are clearly distinct with a salt-and-pepper arrangement highly unlikely to be found in an adult primate and smooth map highly improbable in the rodent. Nonetheless, selectivity and stability measures alone are unable to distinguish between these two cases.

Mathematically, the computation of map stability and selectivity values of a large primate orientation map is identical to any spatially scrambled version of itself. It is clear that to properly assess developmental models of orientation map formation, an

additional metric is needed that can take the global spatial organization of the map into account.

### Existing orientation map structure metrics

The biological plausibility of an orientation map is not only defined by the properties of individual units but is a function of the overall spatial map organization. Some existing non-local measures will be considered before the novel $\pi$ pinwheel map quality metric is introduced.

Local smoothness is one property that could easily distinguish between how orientation maps are spatially organized in primates from the salt-and-pepper arrangement in rodents. Several measures of smoothness have been used in the literature including the "local homogeneity index" (Nauhaus et al., 2008) and the "local input orientation selectivity index" (Schummers et al., 2004). These measures express how rapidly orientation preference change across the cortical surface in a small region around a particular cortical location.

Although these smoothness metrics can distinguish salt-and-pepper organizations from primate maps, they fail to distinguish between two maps that are equally smooth. In other words, there are many possible maps with an unrealistic spatial organization for any given chosen of smoothness. To illustrate, a planar sinusoidal wave pattern of orientation preferences is a pattern that is both very smooth and biologically implausible. Autocorrelation is a similar measure that acts as as another smoothness criterion which is just as weak when assessing the plausibility of map structure.

Another approach is to examine the statistical properties of population distributions instead of averaging values across all units and reducing the measurement to a single scalar. For instance, it has been empirically shown that the representation of orientation preferences across a map deviates from a perfectly uniform distribution (Coppola et al., 1998; Müller et al., 2000; Tanaka et al., 2009).

A simulated map with a plausible orientation distribution histogram will be arranged such that each preference value has the appropriate overall level of representation across the map. Although this is an important criterion, species with smooth map organization, it is only applicable over large cortical areas covering enough different orientation hypercolumns to achieve an unbiased average.

Conversely, in a species with salt-and-pepper organization, the orientation distribution histogram can take any shape once the preferences of enough cells are collected together. Once again, this measure cannot detect the difference between an appropri-

ate balanced smooth map and a salt-and-pepper arrangement although this metric can detect when a particular orientation preference is under- or overrepresented.

A more promising approach is to examine the spatial properties of the orientation hypercolumns themselves. Different species have different hypercolumns sizes and estimating the typical distance between orientation hypercolumns is a crucial statistic for describing the spatial organization of the map. Generating a good distance estimate involves analyzing the spatial distribution of preferences in a way that offers some useful insights into what constitutes a realistic map.

## 5.3 Constructing a map quality metric

In this section, a simple approach for estimating the hypercolumn distance is first described, followed by a description of the pinwheel density, and more specifically, the phenomenon of $\pi$ pinwheel density.

The hypercolumn distance offers a natural species-dependent spatial feature to measure, whereas pinwheel density is normalized by the hypercolumn distance to have a species-independent reference value. The novel use of $\pi$ pinwheel density for assessing orientation map simulations is discussed and the heavily tailed gamma squashing function is then introduced.

### 5.3.1 Estimating the hypercolumn distance

There are several different approaches to estimate the hypercolumn distance, $\Lambda$, based on the spatial periodicity of the orientation preferences across the cortical surface. The simplest approach is to examine the polar Fourier power spectrum which tends to be ring-shaped in experimentally measured maps (Erwin et al., 1995; Blasdel, 1992a). In Fourier space, a periodic, spatially extended signal with an isotropic characteristic distance is mapped to a fuzzy ring, as shown by the middle column of Figure 5.1.

In addition to generating an estimate from the size of the Fourier power spectrum ring, a more advanced wavelet-based analysis technique is sometimes used (Kaschube et al., 2010). Wavelet analysis offers an improvement over the Fourier approach when it is necessary to account for retinotopic anisotropy in V1 that results in a distortion of the ring in the Fourier domain. We will focus on the Fourier approach here, as wavelet analysis is significantly more complicated, the quantitative difference from the Fourier approach is small, and all the simulated maps that we will be considering are perfectly

isotropic.

An example polar Fourier power spectrum of an orientation map is shown in Figure 5.1B. The simplest way to find the radius of a circular ring (i.e., for an isotropic map) is to average the power spectrum radially to build a one dimensional histogram, as shown in the column on the right side. Locating the tallest bin is one simple way to estimate the ring radius but this approach ignores the information contained in the overall shape of the distribution.

Following the approach used (Kaschube et al., 2010), an estimate of the ring radius can be obtained from the $a_1$ coefficient computed using a least-squares with respect to the wavenumbers ($k$) using the following fitting function:

$$f(k) = a_0 \exp\left(\frac{(k-a_1)^2}{2a_2^2}\right) + a_3 + a_4 k + a_5 k^2 \tag{5.2}$$

This fitting procedure makes use of the additional information in the ring cross section and these fits are shown by the red traces in the right column of Figure 5.1. A perfectly thin ring is impossible, as this would require an exact periodicity in all directions at all points in the map. At the other extreme, the sort of salt-and-pepper arrangement shown in Figure 5.1D will have long-range periodicity and thus no ring structure. A plausible simulated orientation map must therefore have a noisy ring with some finite thickness, due to the quasi-periodic repetition of the orientation hypercolumns.

In addition to allowing the hypercolumn distance to be estimated, the polar Fourier power spectrum helps reveal spatial properties of the map without uninformative phase information. Although the use of the Fourier power spectrum help simplify the analysis, hypercolumn distance alone does not offer an objective way to distinguish a high quality map from a poor one. In particular, the size of the Fourier ring is species dependent, with different species having different characteristic hypercolumn sizes. It may be possible to express a criterion in terms of the tightness of the ring (i.e., how ring-shaped it is), but as we will see in the next section such a criterion is subsumed by the proposed $\pi$ pinwheel density metric.

### 5.3.2 Pinwheels and $\pi$ pinwheel density

In addition to orientation hypercolumns, another natural spatial feature that can be identified and quantified are the pinwheel singularities. A pinwheel is a point in the map around which all possible orientation preferences are represented. Pinwheels have one of two polarities, depending on whether the orientation preference increases in the clockwise or counterclockwise direction around it.

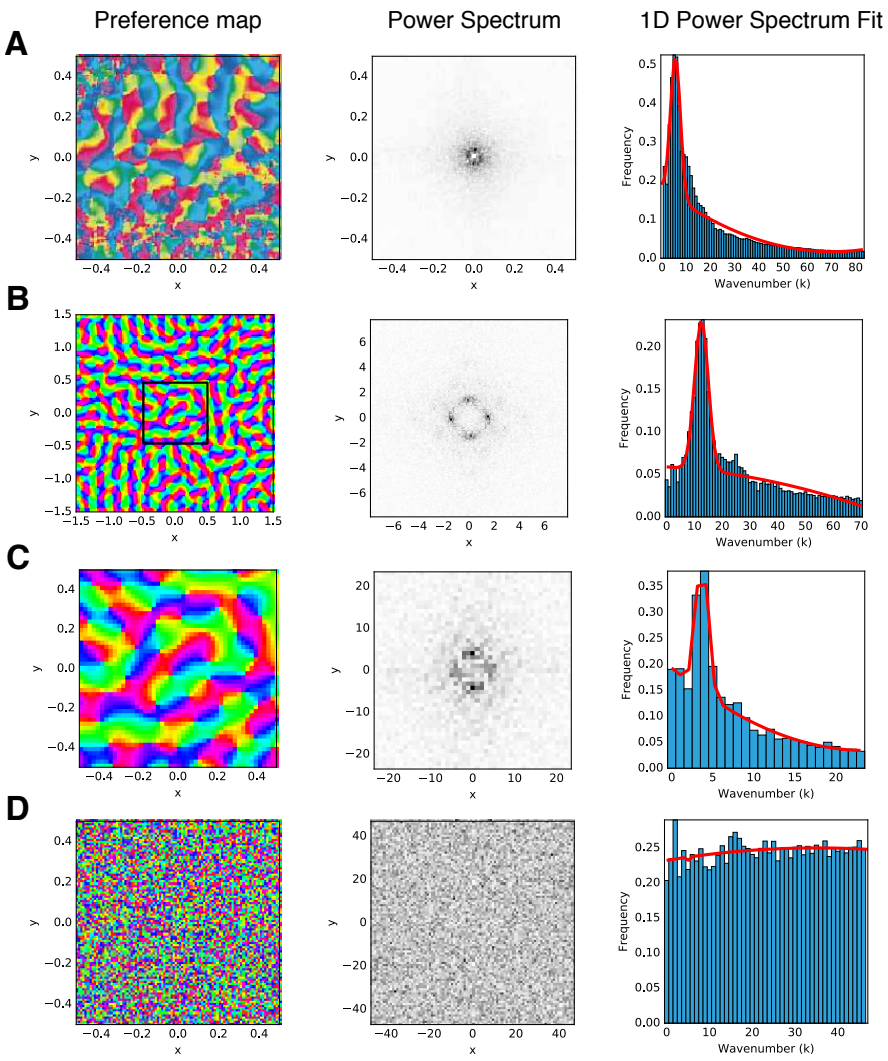


Figure 5.1: **Polar power spectrum analysis of a range of experimental and artificial orientation maps.** Left column shows orientation preference maps. Middle, the corresponding polar Fourier power spectrum. Right, the 1D power spectrum histogram (blue) together with the fits of Equation 5.2 (red) used to estimate the hypercolumn distance following the methods of (Kaschube et al., 2010). (A) Ferret orientation map reproduced from Chapman et al. (1996). Note that the central $4 \times 4$ area of the polar power spectrum is set to zero to eliminate the DC component and better illustrate the power spectrum fit on the right. (B) Artificial orientation map (GCAL model) with a larger ring in the Fourier power spectrum. The stronger power in the cardinal directions is an artifact of running a simulation on a Cartesian grid. (C) The central area of (B) marked in the black square. Note that with fewer samples, the ring is noisier but approximately the same relative size as before. This is because the increased spatial frequency of the hypercolumns in the available area is offset by the reduced number of samples necessary to compute the high frequency components of the image. (D) Example of a map generated by random noise, similar to the salt-and-pepper arrangement in rodents. The ring is no longer present in the polar power spectrum and the 1D histogram is now flat.

In smooth, well-organized primate map, pinwheels tend to be fairly evenly distributed over the cortical surface. The spatial distribution of pinwheels has been analyzed and it has been found that the average distance between nearest-neighbor pinwheel pairs tends to be lower between pinwheels with opposite polarity (Müller et al., 2000).

Automatic identification of pinwheel locations is possible by finding the intersection between the zero contours in the real and imaginary components of the polar representation of orientation preference (Löwel et al., 1998). Correctly identifying pinwheels in experimental maps is non trivial, requiring a careful filtering procedure that will be discussed shortly.

The orientation map quality metric for developmental simulations described in the next section is based on the remarkable empirical discovery of $\pi$ pinwheel density across carnivorans, primates, cats, and tree shrews (Keil et al., 2012; Kaschube et al., 2010). This observation relies on the linear relation between the size of orientation hypercolumns across different species, and the pinwheel count per unit area. As map size increases, the average number of pinwheels in a unit hypercolumn area ($\Lambda^2$) tends to the mathematical constant $\pi$. The pinwheel density, $\rho$, is simply the number of pinwheels found on average in area $\Lambda^2$, where the $\Lambda$ value may be computed using the Fourier power spectrum method described in the previous section.

To support the claim that $\pi$ pinwheel density is a universal property, Kaschube et al. (2010) analyzed hundreds of orientation maps in three species, namely tree shrew, galago, and ferret. Figure 5.2A shows the linear relationship between pinwheel count per unit area and the hypercolumn distance and Figure 5.2B shows that this gradient centers around $\pi$.

These maps used in this analysis had to be carefully filtered, using the methods described in the supplementary materials of Kaschube et al. (2010). In particular, a Fermi filter was used with species-dependent cutoff frequency values in order to ensure the filtering process did not mask or distort the computed pinwheel density. For instance, Gaussian filtering does not allow an objective definition of the pinwheel density (Kaschube et al., 2010) as it obscures the real structure of the underlying map.

The surprising empirical observation of $\pi$ pinwheel density across species has the necessary attributes to build a suitable metric. It defines a unique reference value that can be used to constrain artificially generated maps in relation to experimental measurement.

The dimensionless constant of $\pi$ is not selected arbitrarily but emerged from a

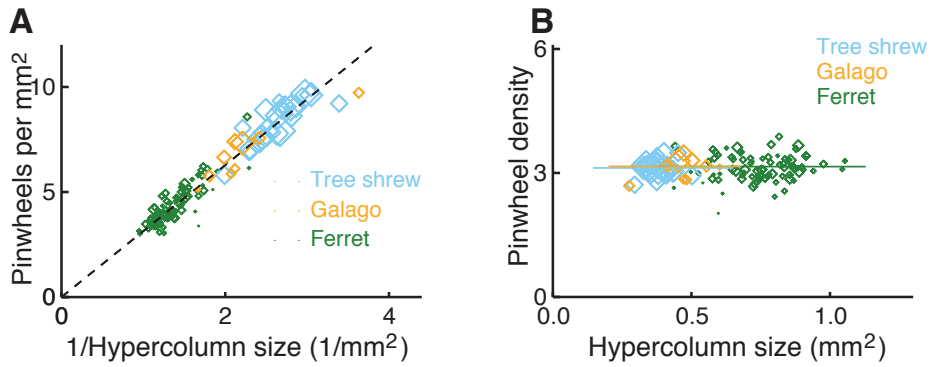


Figure 5.2: **The relationship between absolute pinwheel density and hypercolumn size.** (A) A linear relationship between number of pinwheels per $\mathrm{mm}^{-1}$ of cortex and the inverse hypercolumn size in ferret, galago, and tree shrew. (B) The gradient of this relationship is a unitless constant shown on the $y$ axis, with pinwheel densities centered on $\pi$ . Figure reproduced from Kaschube et al. (2010).

theoretical analysis in terms of the universality class of self-organizing phenomena to which orientation map formation is thought to belong (Kaschube et al., 2010). As pinwheel density can be computed automatically when considering simulated maps, it offers a solid basis to build a species-independent map quality metric for use in orientation map models.

### 5.3.3 An orientation map quality metric for simulations

The pinwheel densities shown in Figure 5.2B have been computed from experimentally recorded orientation maps which have a small spread around the value $\pi$. To build a map quality metric based on pinwheel density it is necessary to consider the potential range of pinwheel densities of orientation maps that may be generated by a simulation, from the highly implausible maps to highly realistic ones.

As pinwheel density is simply the average number of pinwheels per hypercolumn area, it can be any positive value. An arbitrary number of pinwheels may reside within a hypercolumn area and the hypercolumn area itself is also a positive and potentially unbounded quantity. It follows that the only mathematical constraint on a pinwheel density value is that it must be positive.

Figure 5.3 shows a few concrete examples pinwheel densities for some simple patterns as well as the pinwheel density for some example orientation map simulations. We see that the synthetic maps shown have varying pinwheel densities that are easily distinguished from $\pi$, varying between 0.3 for the stripe pattern and 4.0 for the periodic

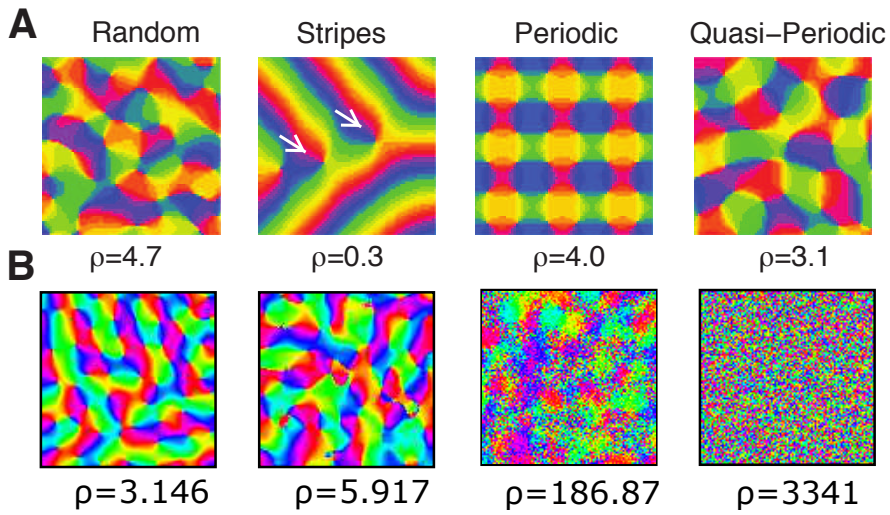


Figure 5.3: **Example pinwheel densities for synthetic and simulated orientation maps.** (A) Example pinwheel densities for synthetic maps reproduced from Kaschube et al. (2010). (B) Example pinwheel densities for a range of simulated orientation maps of varying quality. We see that low quality simulated maps tend to have high estimated pinwheel density values.

tiling. The pinwheel densities for the simulated orientation maps, in contrast, range from realistic values of $\pi$ to very large values, with the worst simulated map example shown with a pinwheel density of 3341. The reason for such high potential values when dealing with simulated maps will be discussed after a unit metric has been established.

#### Normalizing the metric

Normalized metrics that take a value between zero and one are particularly useful, where zero denotes the worst possible metric value and one denotes the highest possible value. Normalized metrics can be multiplied together to create other normalized metrics and it is useful to have a well defined upper and lower bound.

Given that pinwheel densities are positive and unbounded in principle and the high values of pinwheel density observed in the worst orientation preference maps shown in Figure 5.3, it would be useful to find a suitable way of normalizing pinwheel density. Such a metric would have the following six properties:

(1) The metric should vary smoothly with pinwheel density, because functions with smooth gradients are easier to reason about. Functions with arbitrary boundaries are more difficult to work with and are tricky to optimize. (2) If the pinwheel density is $\pi$, the metric should have a value of unity, indicating a structure that is indistinguishable

from an ideal, biological map with assumed $\pi$ pinwheel density. (3) If there are no pinwheels at all, the metric value should be zero, as a metric based on pinwheel density is only applicable when there are sufficient hypercolumns for a good hypercolumn distance estimate and sufficient pinwheel singularities. (4) If there are many pinwheels and the pinwheel density is high, the metric should tend to zero, reaching zero at infinity. (5) Given that only pinwheel densities near $\pi$ correspond to biologically plausible maps, the metric should converge to nearly zero quickly, even if it only asymptotically approaches a truly zero value. (6) Given that the modal value $\pi$ is much closer to zero than infinity, this function will necessarily be heavily tailed.

From these six constraints, the goal is to find a unimodal, heavily tailed distribution. The distribution must start at the origin, peak with unit height at $\pi$ and then rapidly fall back to zero as pinwheel densities approach infinity. For these purposes, the gamma distribution was chosen, as it is defined by two arguments that will be reduced down to a single parameter.

The gamma distribution is a probability distribution that is defined as follows:

$$P(X = x) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} \exp^{\frac{-x}{\theta}} \tag{5.3}$$

the gamma function. It is now necessary to find the appropriate value of $\theta$ given $k$ to ensure that the peak always appears at $\pi$, and then it is necessary to normalize the distribution so that this peak value is unity.

Finding the expression for $\theta$ in terms of $k$ is straightforward given that the mode of the distribution is expressed by $(k-1)\theta$ for values of $k$ greater than one. Constraining the desired mode value to the value $\pi$ and rearranging yields $\theta = \frac{\pi}{(k-1)}$. This leaves $k$ as the remaining free parameter that is used to determine how heavily tailed the distribution is. Finally, to normalize the metric, it is sufficient to divide the output of this mode-constrained gamma distribution by its value at $\pi$.

The map quality metric (*mq*) can now be expressed in terms of the pinwheel density $\rho$ and the $k$ parameter as follows:

$$mq(\rho;k) = \frac{\rho^{k-1} \exp^{\frac{(1-k)\rho}{\pi}}}{\pi^{k-1} \exp^{(1-k)}} \tag{5.4}$$

The normalization term that was required to make gamma function into a probability distribution cancels out in this formulation, to ensure a value of unity at $\rho = \pi$. For a given value of $k$, the denominator in the expression above only needs to be evaluated once in order to obtain the appropriate normalization constant. As shown below, the

value of $k$ can be set by the modeler to ensure that the metric is sensitive to the range of maps typically needing to be distinguished in practice.

#### Applying the map quality metric to map simulation

The apparent convergence of the pinwheel density to a value of $\pi$ is an empirical finding, discovered by analyzing a large number of carefully recorded orientation maps across species. The appearance of the mathematical constant $\pi$ is a surprising property of cortical organization, and it remains to be shown that $\pi$ pinwheel density is a suitable criterion to assess simulated maps.

It is particularly important to understand how the pinwheel density estimation algorithm behaves when given poor quality, simulated maps as input instead of a biologically plausible map of the sort analyzed by Kaschube et al. (2010). In order for the metric to be a useful guide in computational neuroscience, it needs to work reliably when supplied an arbitrary input that bears no resemblance to real orientation maps. There is no *a priori* reason to assume that a preference map generated by an arbitrary simulation should have any relation to the biology.

Considering the synthetic patterns shown in Figure 5.3A, it is clear that some patterns can have a pinwheel density less than $\pi$, such as the stripe pattern with $\rho = 0.3$. If these low pinwheel densities were common in the sort of simulated maps we will be considering, the metric would face the difficult task of distinguishing between maps based on densities in the narrow range $[0..\pi]$.

The evaluation of pinwheel density for the simulated maps shown in Figure 5.3B, it is apparent that simulated maps tend to have pinwheel densities greater than $\pi$. This can be understood by thinking about the space of all possible preference maps, as there are fewer smooth arrangements than noisy arrangements, assuming that each unit is free to adopt any preference value without constraint from its neighbors. When maps are noisy or distorted, the pinwheel density estimation algorithm has a strong bias towards high pinwheel density estimates.

This bias is a consequence of how the pinwheel finding algorithm begins to fail as the concept of pinwheels is rendered meaningless. By definition, a pinwheel is a point surrounded by all possible orientation preferences that are varying in a smooth and monotonic way. Noisy maps fundamentally violate this smoothness assumption, which will cause *any* pinwheel finding procedure to fail.

In particular, consider the method of locating pinwheels via contour intersection (Löwel et al., 1998) used here. The very concept of a tracing a contour relies on the

assumption that there is a smooth underlying function that is being regularly sampled. As a result, when the orientation preference of the samples fluctuates rapidly, the contour tracing process infers contour lines running along a rapidly undulating surface that is assumed to smoothly vary between the available sample positions.

Tracing contours over a rapidly undulating and twisting surface will then result in a very high density of contours lines in both the real and imaginary components of the polar representation. A large number of superfluous contours results in an even larger number of intersections between these contours where each intersection corresponds to a false positive for a pinwheel location.

Considering the effect of noise on preference maps offers an additional insight into how the hypercolumn distance estimate will be affected by poor quality, noisy, or otherwise distorted inputs. In the case of white noise, all spatial frequencies are represented equally and the fitting procedure described in section 5.3.1 will fail. This is shown in Figure 5.1D, where the power histogram is flat—there is no peak position to estimate, allowing the estimated hypercolumn distance to take on any arbitrary value, from zero to the entire width of the map.

Now we have considered the effect of noisy input on both pinwheel finding and the hypercolumn distance estimation, we can now consider how the pinwheel density is likely to be affected, given that it is computed as the average number of pinwheels per hypercolumn area. Noise will result in a large number of pinwheel false-positives and we have shown why the estimated hypercolumn distance can take any value from zero to the map size.

In the overwhelming majority of cases, this procedure will result in a very high estimated pinwheel density given noisy, non-smooth inputs. It is conceivable that if the estimated hypercolumn distance happens to fall on a suitably low value, a hypercolumn area might be computed that happens to contain $\pi$ false positive pinwheel locations on average. Statistically, such an occurrence is extremely unlikely and empirically we have not found any poor quality preference map that also has roughly $\pi$ pinwheel density. Just in case, however, the analysis in the next section relies on multiple runs to ensure a high-quality pinwheel density estimate that further reduces the chance that any such fluke occurrences will affect the results.

This analysis helps determine the appropriate shape for the heavily tailed distribution mapping the pinwheel density into unit range. In the previous section we saw that the shape of the distribution is determined by the parameter $k$, with high values of $k$ resulting in a broader distribution. In other words, the $k$ parameter is used to determine

the penalty in map quality associated with a deviation from $\pi$ pinwheel density, where the metric becomes more selective the closer the $k$ value is to one.

Given that extremely high pinwheel densities are likely when processing poor quality input maps, a good metric will not tail off to zero too quickly, so that two maps with differing, but large, pinwheel densities can still be compared. From the range of pinwheel densities observed in the analysis presented in the next section, it was found that a value of $k = 1.8$ offers a reasonable level of discrimination.

Lastly, it is important to note that in computational models and simulations, that any observed noise is generally *not* measurement noise, although there can be smooth, systematic errors due to insufficient sampling. This is a crucial step in the analysis, as any simulated maps that are analyzed are expected to correspond directly to the ground truth in the model. In other words, any noise and distortion in the input is assumed to reflect real imperfections in the true map structure, though they may be only slight (but real) differences. Any filtering procedure would be difficult to automate, and there is typically no meaningful way in which such a filtering process would operate to extract the "real" underlying hypercolumn and pinwheel organization when considering model maps. In the models we will analyze in the next section, possible sources of disruption include distortions resulting from excessively fast Hebbian learning, border effects at the simulation boundaries, and, given particular initial conditions, it is even possible to explicitly achieve a salt-and-pepper organization (Law, 2009).

## 5.4 Evaluating developmental map models

In this chapter so far, we have discussed why developmental map models are relevant when attempting to understand the evoked response and have presented a novel map quality metric to assess the biological plausibility of simulated orientation maps. In this section, this map quality metric is used to analyze the behavior of the GCAL developmental map and the three submodels leading up to it (Stevens et al., 2013b). The goals are to (1) show why a map quality metric is essential for evaluating map forming models, (2) briefly motivate the mechanisms of the GCAL model as they are used again in the next chapter, and (3) validate the GCAL model as the basis of the unified spatiotemporal model presented in this thesis.

In order to obtain a good estimate of the pinwheel density, a large enough cortical area needs to be simulated in order to capture enough pinwheels and collect sufficient statistics. This was achieved by running multiple simulations using different random

seeds which control the randomized training patterns and weight initialization, resulting in different orientation maps after development. In addition, a cortical area of size $1.5 \times 1.5$ (arbitrary units) was simulated, in order to crop off the borders and restrict analysis to the central unit area only. This helps reduce border effects that act to distort the map structure towards the simulation boundaries.

The models that shall be analyzed in turn are called L, AL, GCL, and GCAL. Discussion surrounding the various mechanisms involved at each stage will be kept to a minimum, as the goal is to demonstrate application of the map metric and to validate GCAL. Detailed discussion regarding the effect of the homeostatic threshold adaptation and contrast-gain control mechanisms can be found in Stevens et al. (2013b), included in the Appendix.

### 5.4.1 Empirical constraints on map formation

Before evaluating the development of simulated orientation maps, it is necessary to understand the observed developmental properties of real, biological maps. Most of these empirical constraints are derived from the chronic study in Chapman et al. (1996), discussed in section 2.4.1 and summarized by Figures 2.8 and 2.9.

In short, orientation maps begin with low initial selectivity which rapidly increases and plateaus by the end of development. Experimental orientation maps are stable, in that any orientation maps after the earliest stages of development are highly similar to the final recorded map. This indicates that the preference is determined early on, while the orientation selectivity is still increasing. The new constraint introduced in this chapter is the requirement for $\pi$ pinwheel-density maps by the end of development, corresponding to a map quality metric close to unity. I.e., simulated maps will be considered plausible only if their development process is stable and results in selective maps that also have close to $\pi$ pinwheel density. We can now use these metrics to evaluate the plausibility of a series of models gradually increasing in complexity and realism.

### 5.4.2 The L model

The L model is designed as a baseline model stripped of everything except the core features shared by all self-organizing map models with plastic lateral connectivity. This model includes the minimal set of mechanisms necessary for smooth orientation map formation: fixed difference-of-Gaussian receptive-field profiles in the ON and OFF

sheets, static firing thresholds, and cortical Hebbian plasticity, including postsynaptic weight normalization.

Such a model is extremely fragile, and the goal is to show that there is some training regime where smooth maps can develop, in order to show that the essential process of self-organization is still intact. This regime was found using the map quality metric presented in this chapter. Figure 5.4 shows the mean map stability (green), selectivity (red), and map quality across contrasts across 10 different simulation runs initialized with different random seeds. It is clear that stability drops rapidly as contrast increases while selectivity slowly increases. It is the $\pi$ pinwheel density metric that highlights the region where the L model performs best, shown by the green curve in Figure 5.4A. From the peak value of the map quality metric, it can be seen that L maps are most similar to biological maps around 25% contrast, with an example of such a map shown in Figure 5.4C.

Having an automated metric that can be used to summarize the quality of many maps at once, allows this sort of observation to be made easily. Without a metric, not only would it be difficult to justify any particular subjective judgment but all 200 orientation map simulations that are succinctly summarized by the green trace would have to be inspected manually.

Figure 5.4B shows an example orientation map illustrating how the L model behaves at low contrast. In this regime, map organization did not take place, as there was no activity in V1 to drive Hebbian learning and self-organization. In other words, the afferent input to each V1 unit was insufficient to exceed the fixed activity threshold. As a consequence, all units retained the initial random orientation preferences assigned by the initial random weight generation, resulting in a correspondingly low orientation selectivity. As the preference structure does not change over development, the stability metric is high. The lack of smooth organization due to a lack of self-organization is responsible for the low value for the map quality metric as a large number of false pinwheels being detected, indicated by the white dots.

At high contrast, V1 suffers the inverse problem, with rapid changes to the weights due to high pre- and postsynaptic activity, resulting in a rapidly changing orientation map and consequently, low stability. There is high orientation selectivity, but only because the last Gaussian training stimulus is imprinted in the network weights. These distortions greatly increase the pinwheel density, with many pinwheels found at the noisy boundaries where the orientation preference values change abruptly.

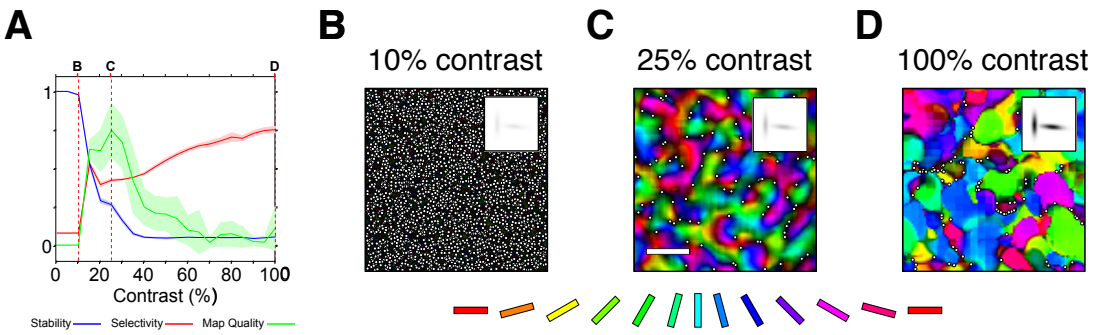


Figure 5.4: **Analysis of the L model robustness in terms of selectivity, stability, and the new map quality metric.** (A) Mean map stability (green), selectivity (red), and map quality (blue) as a function of contrast, across 10 simulations with randomized input sequences where the shaded areas indicate the 95% confidence interval for each metric. The L model has a peak in the map quality for a narrow range of training contrasts. Lines marked B, C, and D indicate the contrast levels used to train the example maps shown in this figure. (B) Example orientation map trained at 10% contrast, where no development occurs in this case. Inset shows example training pattern and white points indicate false-positive pinwheel locations resulting in a high pinwheel density and correspondingly low value for the map quality metric. (C) Example orientation map trained at 25% contrast, where the map quality metric peaks. White bar shows the hypercolumn distance, when it is possible to estimate from the Fourier power spectrum. At this contrast, many pinwheels are correctly identified. (D) At 100% contrast, the map is distorted, resulting in clusters of false pinwheels and a correspondingly low map-quality metric.

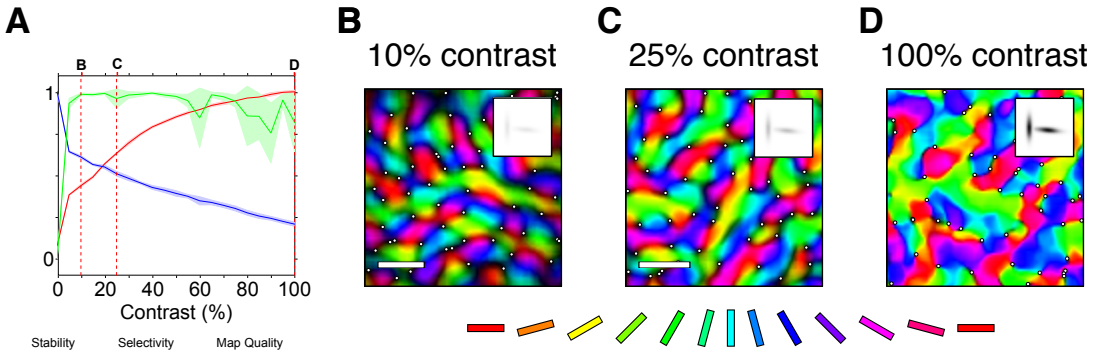


Figure 5.5: **Analysis of the AL model robustness in terms of selectivity, stability, and the new map quality metric.** [Same conventions as in Figure 5.4.] The AL model has develops high-quality maps for low-contrast inputs (B and C, with high map metric, well-identified pinwheels, and accurately estimated hypercolumn distance). However, at higher contrasts (e.g. C), it gives highly variable performance and low-quality maps, with clusters of false pinwheels that resemble the L model.

### 5.4.3 The AL model

The AL model augments L with an adaptive, homeostatic threshold which restores self-organization at low contrasts. If the initial threshold is too high for a V1 unit to respond, the lack of activity will gradually drop the threshold till the response is restored, allowing map development to proceed.

With adaptation included, the map quality metric is close to one for low contrasts but begins to drop as the contrast increases, as seen in Figure 5.4A. This is similar to the problem with the L model when high contrast training stimuli are used. Although the threshold of the V1 units rises due to the strong input, keeping the postsynaptic activity within a suitable range, the presynaptic activity is unaffected by this mechanism, resulting is rapid Hebbian learning that distorts the map. The poor map organization results in clusters of false pinwheels as well as unreliable hypercolumn distance estimates as the size of hypercolumns begins to vary wildly. This unreliability in the hypercolumn distance is a major contribution to the variance in the map quality metric at high contrast.

Without the map quality metric, it would be difficult to judge whether the AL model performs well at high contrast from the mean stability and selectivity curves, given that stability decreases but selectivity increases. The example map in Figure 5.4D does illustrate the sort of distortions that are occurring but it is the map metric that helps conclusively show that these distortions are occurring across all the high contrast

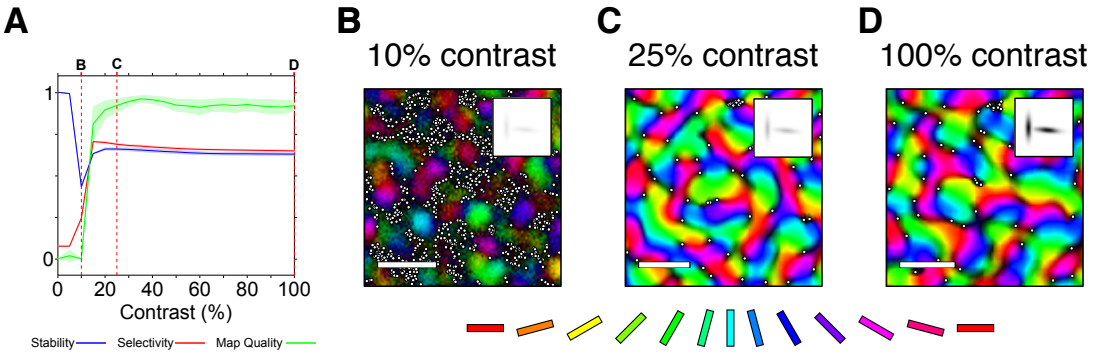


Figure 5.6: **Analysis of the GCL model robustness in terms of selectivity, stability, and the new map quality metric.** (A,B,C) [Same conventions as in Figure 5.4.] For relatively high-contrast inputs, the GCL model has a consistently high map quality with relatively low variance. The metric fails to reach 1.0 because some small areas of the map never self-organize, consistently being inhibited by their sooner-developing neighbors and thus unable to learn the input patterns.

simulations.

### 5.4.4 The GCL model

The GCL model removes the homeostatic adaptation mechanism and implements contrast-gain control in the ON/OFF sheets instead. This results in the regulation of the presynaptic activity which also boosts map quality at low contrasts, with the gain control mechanism boosting weak stimuli to the point where V1 units receive sufficient input to respond even with a fixed threshold.

The GCL model is more robust than the AL model at high contrasts, with contrast-gain control regulating the presynaptic response of the afferent projections. For the training patterns strong enough to drive development, as shown in Figure 5.6C and D, the pinwheel density of the GCL model is close to $\pi$ with some small additional variance.

The map quality of the GCL model is consistently high, but there are some noisy regions that remain, in low selectivity areas of the map. These artifacts correspond to areas of the map that have failed to develop properly due to strong lateral inhibition by other highly responding units in the vicinity. These noisy regions are not smooth, resulting in small clusters of false pinwheels that lower the map quality and result in the variance of the green curve in Figure 5.6A.

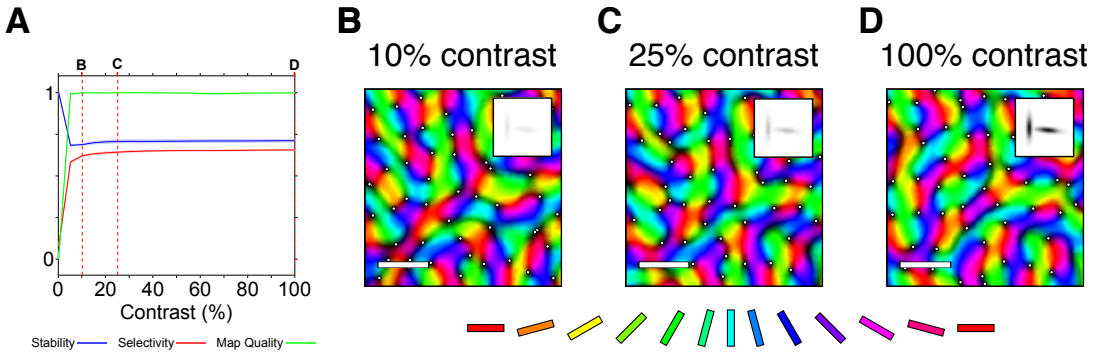


Figure 5.7: **Analysis of the GCAL model robustness in terms of selectivity, stability, and the new map quality metric.** [Same conventions as in Figure 5.4.] The GCAL model has high selectivity, high stability and a nearly perfect map quality metric value ($\pi$ pinwheel density) for all contrasts where any self-organization occurs.

### 5.4.5 The GCAL model

GCAL is the final developmental model that will be analyzed with the map quality metric and this model is the one that will be extended in the next chapter. GCAL simply combines the cortical adaptive threshold mechanism of the AL model with the subcortical contrast-gain control mechanism of the GCL model.

As Figure 5.7 shows, GCAL is an incredibly robust model that is both stable and selective, and also very reliably develops maps with close to $\pi$ pinwheel density. As soon as GCAL is driven strongly enough for development to occur, the final map selectivity and developmental stability is consistently high across contrast. Furthermore, the map quality metric is nearly indistinguishable from unity, with very little variance.

As GCAL model was developed prior to and independently of the map quality metric, the discovery that GCAL orientation maps are close to $\pi$ pinwheel density is an important validation of both the model and the metric. The map quality metric served its purpose well, by quantitatively supporting earlier subjective judgments regarding map quality in the L, AL, GCL, and GCAL models that led to the design of GCAL.

## 5.5 Discussion

The map quality metric introduced in this chapter has now been applied in the context of a real analysis that led to the publication of the GCAL model (Stevens et al., 2013b) attached in the Appendix. Although the GCAL model existed prior to this work, in the absence of a map quality metric, it was difficult to conclusively demonstrate that

GCAL formed high quality orientation maps. In other words, it was not possible to perform the kind of quantitative analysis that has been presented here.

The map quality metric proved vital in analyzing the L, AL, GCL, and GCAL models for four reasons: (1) it is an orthogonal measure to the selectivity and stability measurements, which cannot be omitted in a detailed analysis of how the mechanisms in the GCAL model affect the process of development, (2) it is independently informative, revealing information that would be otherwise be difficult to assess, such as the contrast value at which the L model has the highest quality maps (a reference point that was then used throughout the analysis), (3) it offers a well-defined, unambiguous, species-independent reference value, unlike the mean map selectivity, for instance, and (4) it quantitatively validates the primary output of the GCAL model, allowing it to be presented to the scientific community, without having to resort to subjective quality assessments.

To further illustrate the point regarding the orthogonality of the map quality metric, it is feasible for a model to have exactly the same selectivity and stability profiles as presented in part A of Figures 5.4–5.7 without ever having organized into something other than a salt-and-pepper arrangement. If this had been the case, GCAL would have failed as a model of smooth orientation map formation, demonstrating that a high pinwheel metric value is not simply desirable, it is *necessary*.

Although the concept of pinwheel density and $\pi$ pinwheel density is not novel, this is the first time it has been used as a metric to assess this type of self-organizing map model. Pinwheel density has now been used in the evaluation of other map-forming models, typically by showing a violation of the $\pi$ pinwheel-density criterion using analytical approaches. For instance, the stochastic wiring model proposed by Paik and Ringach (2011) has been demonstrated to be unable to satisfy $\pi$ pinwheel density (Schottdorf et al., 2015). In addition, the elastic net model is also unsatisfactory, as $\pi$ pinwheel density can only be achieved by freezing development at a particular point in time (Keil and Wolf, 2011).

Unlike these other classes of model, there is currently no way to reason about the eventual properties of orientation maps during development when working with self-organizing map models, apart from running a numerical simulation. As a consequence, the map quality metric has been used to directly compare the properties of the observed model map at some stage of development with the experimental constraint derived from maps in mature animals.

The use of $\pi$ pinwheel density as an automated metric to assess these particular

models also had to be justified carefully. In section 5.3.3, it was necessary to reason about the way in which the pinwheel location and hypercolumn distance estimation algorithms fail for non-biologically plausible inputs. This showed that poor quality maps tend to fall into the long tail of the gamma squashing function where pinwheel density values are high. In other words, poorly organized or undeveloped maps will have high pinwheel density and it is the self-organization process that brings the value closer to $\pi$.

This is partly because the developmental map simulations we are considering have a randomized initial configuration and partly because map distortions and other defects tend to appear noisy. A metric based on $\pi$ pinwheel density may not be suitable when assessing other classes of model. For instance, suppose a model where the orientation maps are always guaranteed to be smooth and where striped configurations are possible, such as the one shown in Figure 5.3A. In this case, there a possibility that an implausible map might have close to $\pi$ pinwheel density purely by chance. An underlying striped configuration could have a pinwheel density below $\pi$ and some other effect might then act to bring the pinwheel density closer to $\pi$ reference value. Thus even though the pinwheel-density metric has been very successful for these models, it may not be appropriate for every type of model.

## 5.6 Conclusion

GCAL was a simple, stable, and robust model of orientation map development prior to the start of this research project. The map quality metric and analysis approach developed in this chapter was necessary to validate the model by demonstrating these properties in an objective way. Establishing a convincing way to analyze the plausibility of map development in models such as GCAL was a key part of bringing this model to publication.

The map metric based on pinwheel density allows map simulations to be evaluated in terms of the spatial organization of orientation preference in way that is independent of the species targeted by the model, so long as it is a species with smooth maps. By showing GCAL has close to $\pi$ pinwheel density, this metric demonstrates that GCAL has nearly perfect orientation maps once border effects are reduced with respect to this empirically determined criterion. Validating GCAL in this way in order to get it published is also important for the work in the following chapter that extends GCAL to construct a mechanistic and spatiotemporally unified model.

# Chapter 6

# Unifying developmental and evoked response dynamics

The evoked response in the primary visual cortex is a reflection of the ongoing state of the visual world in the context of long term visual experience. For the activity of a neuron to represent meaningful information, it must capture the state of a visual feature within a space of possible outcomes, defined by the long-term statistical properties of the environment in which the organism lives.

The response of an individual cortical neuron is only meaningful and biologically relevant when considered in a broad context spanning time and space, as discussed in Section 2.1.2. The activity of a particular neuron isolated from its surroundings is not the same as the visual response, which only emerges from the dynamics of a large population. Similarly, a network of neurons that has not captured the overall, long-term statistics of the visual environment cannot perform any meaningful computation, as the information content of a visual feature is only defined with reference to the expected or learned properties of the environment.

This means that the dynamics of visual perception is determined by the entire history of interaction between an extended volume of neural tissue and the environment. As a result, any complete and mechanistic model of vision must simulate a large population of neurons over a long period of time. Unfortunately, even with state-of-the-art supercomputing resources, it is not yet possible to run a detailed cortical spiking model over the necessary spatial and temporal scales. Moreover, the vast number of parameters such a model would require would be greatly underconstrained by the data, making the results unlikely to correspond to meaningful aspects of the system being simulated.

Building a manageable model that spans the large spatial and temporal scales rele-

vant to the visual process must therefore use appropriate simplifying approximations to reduce computational and parameter-space complexity. One way this can be achieved is to ignore the detailed biophysical properties of individual neurons and use rate-based approaches to approximate the spiking response of small collections of neurons at once. The SIRD and GCAL model of Chapters 4 and 5 are both rate-based models, but they cover very different timescales, i.e. 200 milliseconds vs. days or weeks, respectively. The goal of this chapter is to build a single model that can bridge these levels of description.

The approach used to construct this model is to gradually extend GCAL, improving its temporal properties until it can begin to incorporate the insights of the SIRD model. This progression of features is shown in Table 1.1, starting with a new continuous version of GCAL (CGCAL) that shows how developmental map models can operate using a more realistic model of temporal processing. Next, this continuous GCAL model is calibrated against PSTH profiles in the LGN and V1 to form the TCAL (Temporally CALibrated GCAL) model. This yields a developmental model that has appropriate single-unit evoked responses and that can incorporate the mechanisms suggested by the SIRD model.

The final result is a new type of model that can simulate cortical development as well as the detailed temporal dynamics of the evoked response, within a single conceptual framework. Using this approach, it is possible to explore how gradual activity-dependent learning processes shape the network structure, which in turn determines the cortical response dynamics. This new modeling platform will allow neocortical operation to be explored in a much more unified and holistic manner.

## 6.1 Discontinuous temporal processing in GCAL

Self-organizing map models such as GCAL account for feature map formation via activity-dependent synaptic plasticity mechanisms. The shortest natural unit of time in these models when considering naturalistic behavior of an animal is the visual fixation. Only within a single fixation is it reasonable to assume that the image on the retina will remain in a constant position, assuming that the visual scene is also static.

In GCAL and related models, multiple simulation steps are computed per fixation in order to settle the activity bubbles in the cortical sheet that drive smooth map development. The key assumption made by this class of model is that activity bubbles form within each fixation before learning, so that the learning step and weight update can be

applied at one instant, just before the next training pattern is presented.

In an awake, behaving monkey viewing a natural scene, a fixation lasts between 100 to 400 milliseconds in duration (Gallant et al., 1998). As the GCAL learning rule is applied once per training pattern, these values specify the bounds on the shortest meaningful time period in these types of map forming model. Although there are simulation steps needed to settle the activity bubbles, it is only the final activity profile at the end of a fixation that has any impact on the weights and therefore the eventual model state.

In short, this means that the activity in self-organizing map models such as GCAL within a fixation are only calibrated with respect to the final activity state at the end of each fixation. The evolution of activity in the subcortical pathway as well as the cortical sheet from the initial presentation of the training stimulus up to this point has no direct effect on the developmental process. Expressed in real world units, this means that the highest meaningful temporal resolution of GCAL with respect to the developmental process is between 100 to 400 milliseconds.

### 6.1.1 Discontinuities in the time domain

There are three particular features of GCAL that are appropriate optimizations for a developmental map model yet are inappropriate for a temporal model of activity that has a high time resolution. All of these features result in non-uniform handling of events with respect to time, whereby particular steps in the simulation have a special significance or are otherwise processed differently.

Any non-uniform processing in the model as a function of time points to a potentially implausible mechanism, since the components of real biological systems tend to have a smooth time evolution without dramatically discontinuous behavior. A model without these discontinuities will now be described as “continuous” even though the numerical simulation involves discrete spatial sampling and discrete time steps. As will be discussed shortly, this approach is a step towards building a truly continuous model that would be expressed with differential equations instead of difference equations.

Using this sense of “continuous”, the Continuous GCAL model (CGCAL) introduced in the next section eliminates the temporal discontinuities in the GCAL model, namely the non-uniform timesteps, the elimination of the initial LGN activity as projected to V1, and the periodic scheduling of processes like the Hebbian learning step,

the adaptive threshold adjustment, and artificial activity reset.

#### Non-uniform time intervals between events

The GCAL model is implemented as an event driven neural simulation whereby events are scheduled to run after a specified delay. Each event can trigger some processing which may in turn may trigger further events. This is an efficient scheme for implementing developmental models as different sheets are allowed to update at different rates, and sheets where the activity is known to be held constant (such as the photoreceptor sheet during fixation on a static scene) do not need to be updated. For instance, in GCAL there are 16 settling events in the V1 sheet for every 2 events in the ON and OFF sheets which in turn are triggered by a single event driven by the photoreceptor sheet as the training pattern changes.

In a continuous model of response dynamics where the activities of all the neural sheets are expected to always be varying with time, the benefits of an event driven scheme are reduced and a clocked simulation scheme becomes conceptually simpler. Using a clocked model, newly computed activities are generated for all the units in every sheet of the model at every timestep. It is then possible to assign a real world time value to the timestep as we shall see shortly, making it easier to compare the model responses with experimental recordings.

#### Snapshot learning, homeostatic threshold, and activity reset

In order to be efficient, GCAL executes certain processes at a lower rate than the maximal event rate associated with the recurrent lateral settling in the V1 sheet. These processes must then be adjusted when implementing a continuous model to approximate smooth temporal evolution. Although there are events that appear discrete in the nervous systems, such as action potentials or vesicle release, realistic processes at the level of an individual GCAL model unit are likely to act continuously, whether or not the time constant of the process is fast or slow.

Snapshot learning refers to the way developmental self-organizing map models wait for activity bubbles to emerge before the Hebbian learning step is applied in order to form smooth feature maps. In addition to the synaptic weight adjustment, the homeostatic threshold update is also executed once the 16 lateral settling steps in the V1 sheet are completed for any given training presentation.

This arbitrary waiting period of 16 settling steps before update is difficult to jus-

tify when attempting to model the response on a millisecond timescale. There is an additional process that is also clocked at a different rate, namely the activity reset that is triggered when new afferent input is received. This reset to zero activity is useful when trying to reason about developmental models as it ensures that the dynamics within each fixation duration are independent of each other for a given set of model weights.

When considering a continuous model, there is no reason to think that activity in the cortex cannot persist from one moment to the next. In a naturalistic free viewing condition, saccades that last 20-40 milliseconds do serve as a natural separation between fixation (Gallant et al., 1998) events. Although these intervals can help justify activity resets, the correct approach in a continuous model is to appropriately modify the cortical input and not to instantaneously eliminate all activity between fixations. As we will see next, there is another way activity is instantaneously suppressed in the GCAL model.

#### Suppression of initial LGN activity to V1

In GCAL there are two steps involved when processing activity in the ON and the OFF RGC/LGN sheets. First the initial activity is projected from the photoreceptor sheet to the LGN sheets via the difference-of-Gaussians center-surround weights. This initial activity is *not* projected onwards to the V1 sheet but is then used to compute a new activity in the RGC/LGN sheet in a single step, via the action of the lateral divisive projections used to implement contrast-gain control. Only after the application of this contrast-gain control is the activity projected to the cortical sheet.

In other words, the dynamics of subcortical contrast-gain control are implemented as a one step process in the ON/OFF sheets and entirely invisible to the cortical sheet. In this way, whatever transient response of the gain control mechanism may have is ignored. Although this simplification is appropriate for the purposes of developmental modeling, it is not appropriate in a model explicitly designed to explore the properties of transient responses.

### 6.1.2 Converting GCAL into a continuous model

The Continuous GCAL model (CGCAL) is designed to be as similar to GCAL as possible while removing the issues with discontinuous temporal processing mentioned above. All spatial parameters of the GCAL are left unchanged and the settings of

CGCAL are chosen to mimic those of the original model as closely as possible. Unlike GCAL, CGCAL is a clocked model without activity resets, featuring continuous learning, continuous homeostatic adaptation and continuous contrast-gain control in the ON/OFF sheets.

Each of the following subsections detail the corresponding fix to each of the issues listed in the previous section. Some of the necessary changes were first introduced in Stevens (2011) in an early non-developmental version of TCAL that did not include a continuous version of GCAL, but the formulation developed in this thesis helps separate the effect of each mechanism. CGCAL and TCAL are also now complete developmental models that support scalable timesteps by automatically recalculating the various learning rates, allowing simulations to be run using different temporal resolutions.

#### Clocked simulation

In order to be able to sample unit responses across all stages of the model on a consistent timebase, it is necessary to use a clocked simulation. The original event driven model can be adapted by emitting new events every 0.05 time units from the photoreceptor sheet to drive the activity in the rest of the model. Then it is necessary to ensure that all the projection delays in the model are integer multiples of this timestep to make sure that all the sheets will be updated in lock step, without any updates occurring at fractional timesteps.

This conversion to a clocked mode does not have a major impact on the behavior of GCAL, as all projection delays were already multiples of the minimum 0.05 value used to settle activity in the cortical sheet. After clocking the photoreceptor sheet, all stages of the CGCAL model are updated 20 times per training pattern presentation, corresponding to one unit of simulation time which also corresponds to a single fixation. This modification ensures that the LGN sheet will also receive 20 multiple events from the photoreceptor layer per simulation time. This will then reveal the activity dynamics of the contrast-gain control mechanism in the ON/OFF sheet that were previously implemented as a single step process.

#### Continuous sheet model

The original GCAL model makes use of a customized implementation that enforces an exact number of settling steps in the cortical sheet before waiting for a new input event.

This parameter offers extra control that is helpful when optimizing a developmental, event driven simulation, but is unsuitable for a continuously clocked model. There is no reason to expect lateral interactions to stop occurring after a fixed number of steps, when the activity state is not being reset between fixations. This activity reset and the periodic learning are implemented in this sheet and we have seen that these mechanisms pose a problem for a continuous model.

To address this, a simpler, continuous sheet type was implemented for use by the CGCAL model that processes all incoming events in a uniform way. This new implementation removes the activity reset and ensures that the Hebbian learning rule is applied every timestep. Using this sheet in the LGN means that the initial transient response of the ON/OFF sheets is no longer suppressed from reaching V1.

Together the clocked photoreceptor layer, the continuous sheet implementation addresses most of the issues described in the previous section. In the ON and OFF sheets where Hebbian learning is disabled, it results in a more consistent and continuous application of contrast-gain control while in V1 sheet, it enables continuous learning without activity resets or an enforcing number of settling steps.

#### Rescaled learning rates

Using a continuous sheet requires some of the parameters of the GCAL model to be rescaled, as each mechanism will now be invoked more frequently than before. In particular, parameters relating to time constants or learning rates need to be adjusted, as these implicitly depend on the timestep in GCAL.

By default, the V1 sheet of the CGCAL model applies Hebbian learning once every 0.05 timestep, instead of once per fixation as in GCAL. As a result, the learning rates associated with the afferent and lateral projections in V1 must be adjusted to compensate. Now that the model is clocked in increments of 0.05, there are 20 simulation steps per simulation time unit. Dividing the Hebbian learning rates by this number then helps keep them in line with the original model.

The homeostatic threshold adaptation mechanism in V1 also needs to be adjusted as it is also being invoked 20 times more frequently per unit of simulation time. In other words, the per-unit homeostatic thresholds in the CGCAL model are updated over the course of the response within a fixation, and not just at the end of it. Like the Hebbian learning rates, the homeostatic learning rate is simply divided by 20, that is to say the number of timesteps per unit of simulation time (i.e. per fixation).

### 6.1.3 Behavior of the CGCAL Model

The aim of the CGCAL model is to reproduce the behavior of GCAL without the discontinuous temporal properties of some of its mechanisms. As this required some significant changes to the sheet implementation as well as learning rates, it is necessary to examine the behavior of the CGCAL model relative to GCAL. First the subcortical pathway will be considered, in order to ensure that the temporal response profiles in the ON/OFF sheets are still consistent with the original model.

Figure 6.1 compares the temporal responses of the ON sheet, sampled using a regular spatial grid. These temporal response profiles are obtained from the projection activity, i.e., the activity from the ON sheet as it is projected to V1. This is the reason there is no sharp initial peak in activity in the GCAL profiles, resulting in simple step profiles.

In the middle of the figure, the responses are shown for the new Continuous-GCAL model as it has been described so far. The initial transitory activity peak is now clearly visible, driven by the initial activity from the photoreceptor sheet, through the LGN, showing the early response before contrast-gain control in the ON/OFF sheets can be applied. The subsequent oscillatory response reveals that the contrast-gain control mechanism has its own dynamics that was entirely obscured when gain control was implemented as a one step process. Now that there are just as many events occurring in the ON/OFF sheets as the cortical sheet, it is clear that settling dynamics is a feature of all the visually driven sheets as the response approaches a steady state.

#### LGN Hysteresis

The oscillatory temporal profiles of activity in the ON/OFF sheets of CGCAL are now different from the step profiles received by the V1 units of GCAL. This motivates the introduction of the one new mechanism that is present in the CGCAL and subsequent TCAL models but is not already present in GCAL.

Activity hysteresis can be applied to the continuous ON/OFF sheets of CGCAL to dampen the oscillatory behavior and make the profiles more similar to those in GCAL. This mechanism smooths the activity over time using an exponential falloff, defined by the hysteresis time constant $\tau$.

The following equation expresses hysteresis in terms of the unit activities $\eta$ between one simulation timestep and the next. The activity value after hysteresis, $(\bar{\eta}_t)$ is computed using the time constant $\tau$, the current input activity $(\eta_t)$ and the previous

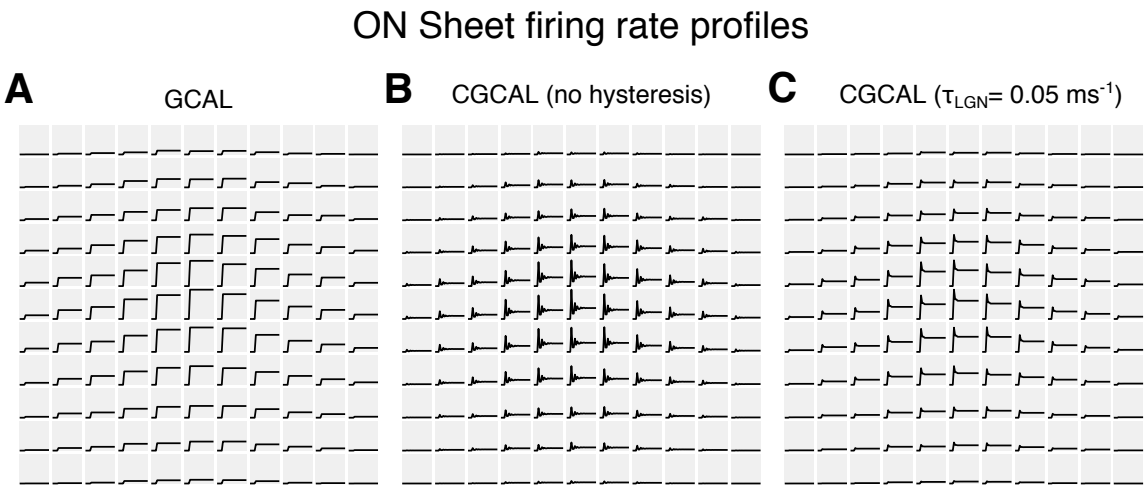


Figure 6.1: **Comparison of ON sheet responses as received by V1 units in the GCAL and CGCAL (with and without hysteresis) in response to an isotropic Gaussian stimulus.** (A) The GCAL profiles are simple step functions as the initial activity before contrast-gain control is not propagated to V1. (B) CGCAL response as a direct analog of GCAL without any additional mechanisms. Responses do not match the profiles on the left due to recurrent action of lateral inhibition within the RGC/LGN sheets. (C) adding a small amount of activity hysteresis brings the CGCAL model closer in line with the GCAL response. Similar results can be obtained by analyzing the OFF sheet responses with profile amplitudes which have an inverse relationship across space.

input activity ($\eta_{t-1}$) where these input values correspond to the half-rectified activity output of the unit, before the action of hysteresis:

$$\bar{\eta}_t = \bar{\eta}_{t-1} + \tau(\eta_t - \eta_{t-1}) \tag{6.1}$$

Using a dimensionless time constant of $\tau_{LGN} = 0.6$ in the new hysteresis mechanism after half-rectification, it is possible to make the ON and OFF responses much more similar to those of the original model. This can be seen in Figure 6.1 by comparing the grid of ON sheet PSTH profiles on the left (GCAL) and right (CGCAL with the hysteresis mechanism). At this point it is worth noting that these are *not* plausible PSTH profiles for LGN cells, illustrating the point that GCAL was only calibrated with respect to the cortical activity at the end of the fixation period. Calibration of these profiles will be done in section 6.2.2.

Equation 6.1 is a difference equation and it is defined in terms of a dimensionless constant, $\tau$. Choosing any specific $\tau$ value means that the response dynamics will

depend on the timestep size, which in this case is 0.05 simulation time units. In order to avoid a strong dependence on the timestep value, it is appropriate to define the hysteresis time constant in $\text{ms}^{-1}$. The hysteresis constant used in the CGCAL ON/OFF sheets is then $0.05\text{ms}^{-1}$, as shown in Figure 6.1C.

Computing the dimensionless $\tau$ value required by Equation 6.1 from the $\tau$ value specified in terms of milliseconds is straightforward: simply multiply by the timestep duration in milliseconds. So far, no real world duration has been assigned to a timestep in GCAL, but this is also easy to compute from the values already stated.

If one simulation time unit corresponds to a single fixation which we will assume to last 240 milliseconds, the 20 simulation steps in this period means that each timestep is separated by 12 milliseconds. We can now compute the corresponding dimensionless hysteresis value using in CGCAL by multiplying $\tau_{LGN} = 0.05\text{ms}^{-1}$ by 12 milliseconds to get $\tau_{LGN} = 0.6$.

This hysteresis constant scaling equation together with the scaling equations for Hebbian and homeostatic threshold learning allows the timestep size to be adjusted, within limits. If the timestep is too large, there will be insufficient simulation steps for these equations to act appropriately. The values may also need adjustment in the continuous limit, as the timestep approaches zero.

As all the dynamical equations of the CGCAL model are now expressed in terms of difference equations that update the model between time $t$ and time $t+1$ without adding discontinuous behavior, it should be straightforward to reformulate the model in terms of differential equations in future. Expressing CGCAL using differential equations might enable the use of analytical techniques that could not be applied before. Such an analysis could prove useful, even if the numerical simulations continue to rely on an approximation based on difference equations.

### CGCAL response

Now that the subcortical responses in the CGCAL model have been adjusted to match those of GCAL, it is possible to explore the difference in behavior between these two models. In particular, it is instructive to visualize how activity bubbles form as the training pattern presented to the photoreceptor sheet changes in spatial position.

Figure 6.2 illustrates one of the main differences between CGCAL and previous developmental map models such as GCAL. As a training pattern is presented, activity bubbles form, as is required for smooth orientation map formation in a self-organizing map model. In CGCAL, unlike GCAL, the Hebbian learning rule is now being applied

on every timestep, as is the homeostatic threshold update.

The difference in activity dynamics between the two models is highlighted when the training pattern changes and new afferent input arrives in the V1 sheet. In GCAL, all the activity in the V1 sheet is reset, with bubble formation starting anew from a condition of zero cortical activity. In CGCAL, there is no artificial temporal boundary between training patterns, and so when the stimulus changes, the existing activity bubbles may either decay, shift position, or get reinforced by additional incoming activity.

The stimulus used in Figure 6.2 corresponds to a fixation switching between two static scenes, in order to allow direct comparison with GCAL. It is worth noting that CGCAL supports more interesting behavior than GCAL, as you could present a moving stimulus within each fixation in order to simulate motion. Motion and direction maps have been simulated in self-organizing map models before (Bednar, 2012; Miikkulainen et al., 2005) but temporal processing in these models was discontinuous, and here a more appropriate model of time could be used in future simulations. Using spatiotemporal stimuli for analysis and training is discussed in section 7.5.9.

#### CGCAL OR map development

Having defined CGCAL and having investigated the response of the ON/OFF and cortical sheets to a single stimulus, it is time to verify that CGCAL still works as a developmental map model. Figure 6.3 shows an example of a GCAL orientation map next to an example of a CGCAL orientation map. Although CGCAL does not have an orientation map as high quality as the GCAL reference, it is clear that smooth map formation is retained in the CGCAL model, despite the numerous changes to the dynamics. At this point, the CGCAL model is complete, having demonstrated the minimal set of changes necessary to redefine GCAL as a continuous, clocked simulation with smooth map formation.

The CGCAL orientation map in Figure 6.3 is clearly less realistic than the GCAL map, which is likely because the activity bubbles are not as clear as they are in GCAL (see Figure 6.2). With parameter tuning to achieve more well-formed bubbles, there is no reason to assume that $\pi$ pinwheel density could not be achieved in CGCAL, as all the relevant mechanisms are inherited directly from GCAL. However, this tuning falls outside the scope of CGCAL, which is explicitly designed to retain all of GCAL's parameters when possible so that we can use it to establish a link between GCAL and the TCAL model to be introduced later in this chapter.

At this juncture, two possible descendents of the CGCAL model can be consid-

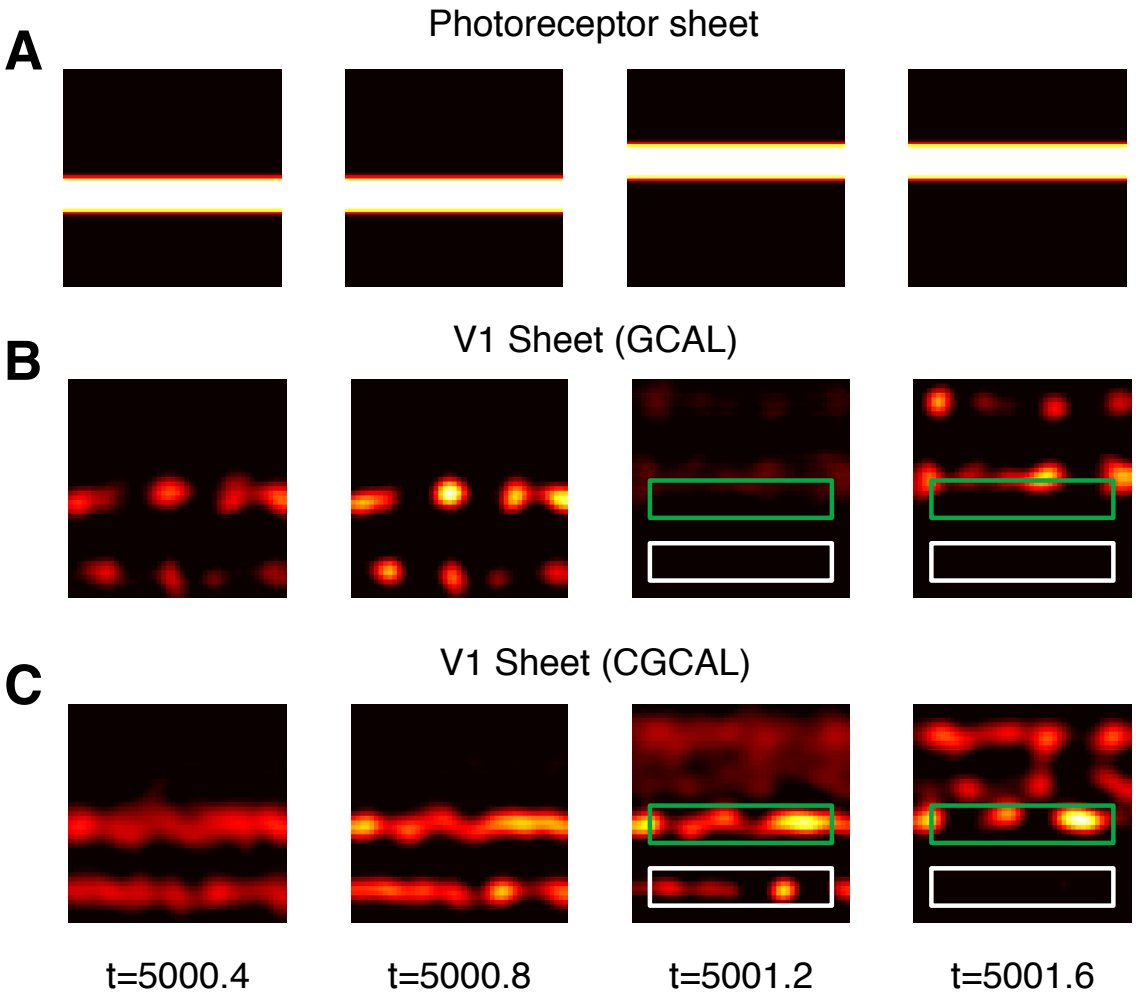


Figure 6.2: **Comparison of responses to horizontal line in GCAL and CGCAL models after $5000$ training iterations.** (A) The horizontal line pattern presented on the photoreceptor sheet at times $5000.4$, $5000.8$, $5001.2$, and $5001.6$ with the pattern appearing at iteration $5000$ with zero activity in the V1 sheet at that time. (B) The activity response in the GCAL model to the horizontal line at the indicated times. Between $t = 5000.4$ and $t = 5000.8$, the activity bubbles are seen to get stronger. As the horizontal line shifts between $t = 5000.8$ and $t = 5001.2$, the activity reset is triggered, eliminating all activity in the V1 sheet. Between $t = 5001.2$ and $t = 5001.6$ the activity bubbles build up in the top half of the cortical area. (C) Corresponding response in the CGCAL model. At the start, between $t = 5000.4$ and $t = 5000.8$ activity bubbles start forming as before. The activity is not reset by the change in line position leaving residual activity in the white rectangle at $t = 5001.2$ which later decays away. In some areas, the residual activity persists from the previous stimulus position only to be is reinforced by the new stimulus (green rectangle).

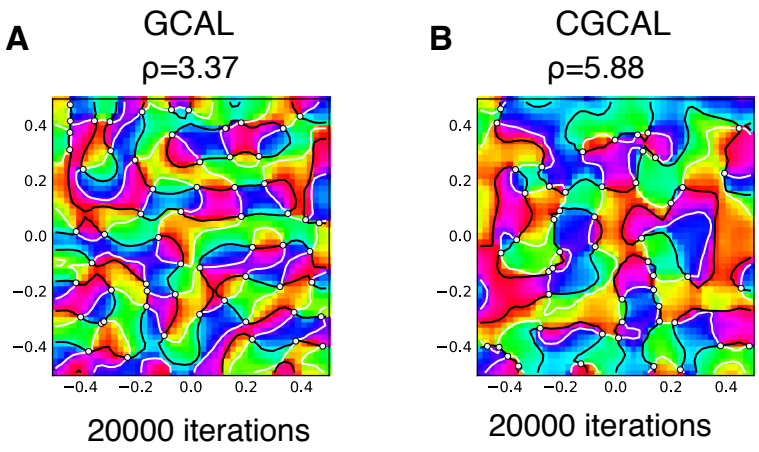


Figure 6.3: **Comparison between GCAL and CGCAL orientation maps after** 20000 **iterations.** On the left is an example GCAL orientation map after 20000 iterations and on the right is the corresponding CGCAL map. Both maps are annotated with pinwheels as well as the real (white) and imaginary (black) contours in the polar representation. Both maps were generated using the same stream of randomized training patterns.

ered. One would explicitly re-tune the parameters to achieve the same, high-quality orientation maps as GCAL, but now in the continuous framework. The second route is to improve the shape of the response profiles in the ON/OFF and V1 sheets. The dynamics of the response in the RGC/LGN sheets are particularly interesting, as CGCAL has revealed dynamics that were previously suppressed. Improving the plausibility of these responses is the goal of the TCAL model, described in the next section.

## 6.2 TCAL: Temporally CALibrated

The argument so far is that CGCAL has a more plausible model of temporal processing than GCAL. As the map quality has decreased and a new hysteresis mechanism had to be introduced to match GCAL, this model does not appear to be an improvement in terms of any concrete results. In addition, by throwing away the optimizations used in GCAL, each simulation run is much slower, with 20× more learning and homeostatic update steps for no obvious benefit in explanatory power. Here we will treat CGCAL as just a step along the way to a more powerful model, TCAL.

TCAL is a direct descendent of CGCAL that, for the most part, only requires some parameter re-tuning to demonstrate something completely new. It is still a developmental model, but unlike previous developmental models, the dynamics of the activity

response within each fixation period are calibrated. Temporally CALibrated GCAL model (TCAL) has exactly the same structure as CGCAL but it makes contact with the experimental data regarding the evoked activity response, as well as with the SIRD model of Chapter 4.

To start with, the timebase will be remapped so that one simulation time unit is intended to correspond directly to a one millisecond duration. Applying this simple, linear transformation will make it easier to compare TCAL responses with the experimental data.

### 6.2.1 Temporal resolution and simulation duration

By convention, one simulation time unit corresponds directly to one millisecond in TCAL. By default, each training pattern is presented to the photoreceptor sheet for the duration of one fixation, lasting 240 milliseconds. This value falls near the middle of the 100 to 400 millisecond range of fixation durations observed in monkeys viewing natural images (Gallant et al., 1998). The default timestep between simulation updates is 5 milliseconds, chosen to offer a good compromise between overall execution time and temporal resolution. As TCAL inherits the timestep scaling equations from CGCAL, it is always possible to adjust the timestep if required.

With 5 millisecond timesteps within each 240 fixation, there are now 48 simulation steps per training pattern in TCAL instead of the 20 steps executed in CGCAL. In line with GCAL, the goal in TCAL is to form orientation maps within $10000 - 20000$ training iterations.

Assuming this constraint is satisfied, it is now possible to compare the time taken for orientation developmental in a real animal to the corresponding simulation time expressed in milliseconds. It is expected that development in the model will happen quickly as using an elevated learning rate is one way to reduce the computational cost of running a developmental simulation. Using TCAL, it is possible to get an idea of how much faster learning occurs in the model than it does in real time.

A simulation of 20000 fixations of 240 milliseconds each corresponds to around 1.3 hours of real visual input, ignoring the duration between fixations. For comparison, orientation map development in ferret as recorded with chronic intrinsic optical imaging takes several days as shown in Figure 2.8. Although an 11 day time period is shown, the initial map structure forms in around two days, between p31 and p33. In addition, the electrophysiological measurements of selectivity shown in Figure 2.9

suggest that map development may be essentially complete by p39.

Even if you consider the lack of visual input from the external world during the periods of sleep, it is clear that the learning rate used in developmental models is very high when compared to the experimentally observed development process. Matching the rate of change induced per fixation would require a learning rate that is a hundreds times slower, or alternatively, simulations that take hundreds of times longer to run.

Achieving such a match is not useful here, as the purpose of a developmental model is to capture the processes involved in forming receptive fields and understanding how they are organized across the cortical surface. If TCAL can demonstrate that units are learning suitable weight patterns and that a self-organizing process is taking place, then it will be successfully modeling a phenomenon that emerges on a timescale of days. The stated time in milliseconds at the end of a simulation run then has to be interpreted cautiously, due to the elevated learning rate.

This example illustrates what is means for a model to span both the timescales of development and the timescales of the evoked response. A 5-millisecond bin is a reasonable interval for collecting action potentials and computing firing rate profiles to be calibrated against experimental PSTHs. If the same model also simulates the emergence of orientation maps, then means that the model is accounting for a developmental process that takes place over several days. Over this timespan, the tuning properties of the units emerge and there are large scale structural changes to the cortex that affect its functional properties.

It is important to emphasize that TCAL can efficiently bridge these timescales using modest computing resources. Compared to an equivalent spiking level simulation of the same cortical area and over the same timescale, TCAL is highly tractable. A TCAL simulation to 20000 steps will complete within around 6 hours using a modern server processor, for the default area and density. It would be entirely feasible to expand on the computational resources to achieve a model that operates in real time with a more realistic learning rate such that a few days running a simulation would correspond to a similar number of days of cortical development.

### 6.2.2 Calibrated single-unit response profiles

TCAL will calibrated in two stages, first by considering the evoked response timescale of milliseconds and then, by calibrating it on the developmental timescale of days. The key question is whether the firing rate responses of TCAL units can be made

sufficiently plausible. In other words, whether TCAL units can have realistic PSTH profiles in relation to the experimental data.

Remarkably, it turns out that highly realistic temporal profiles can be achieved without introducing any new mechanisms at all. The only additional mechanism required by TCAL has already been introduced in CGCAL, namely the hysteresis mechanism applied to the ON/OFF sheets. This is where the first PSTH calibration will occur, and we will see that the dynamics of contrast-gain control revealed by CGCAL are sufficient to generate plausible PSTH profiles.

#### PSTH profiles in the LGN sheets

How can a developmental model achieve plausible LGN PSTH profiles such as the ones shown in Figure 2.4 without requiring new mechanisms? A hint can be seen in Figure 6.1C, where we see the transient dynamics that were first revealed by CGCAL and then suppressed again to try to match GCAL. In TCAL, these transient dynamics are explored and tuned to generate plausible PSTH profiles in the ON and OFF sheets.

In all the models we have considered, the initial activity in the ON and OFF sheets is instantaneously high due to the high afferent drive from the photoreceptor sheet. Then once the lateral inhibitory projection implementing contrast-gain control becomes active, the ON and OFF activity is reduced. In CGCAL, the additional simulation steps revealed oscillations in the response as activity settled towards a steady state. These dynamics were then suppressed in CGCAL to match GCAL, but in TCAL, it is these exact network dynamics *together* with hysteresis that generate plausible firing-rate profiles.

What TCAL shows is that if the transitory oscillations have the right frequency, they can then be appropriately damped to generate a plausible PSTH profile. There are three parameters of CGCAL that can be tuned to achieve this, without modifying the difference-of- Gaussian weights, the afferent weight profiles, or the size of the lateral inhibitory fields. These three parameters are (1) the lateral inhibitory delay of the contrast-gain control connections, (2) the strength of the contrast-gain control connections, and (3) the hysteresis time constant. Note that the spatial sizes of these lateral projections do not need to be modified from the values set in GCAL.

The necessary changes can be justified fairly directly. First, the oscillations in Figure 6.1C are very rapid and can be slowed down by increasing the delay of the lateral inhibitory contrast-gain control projection to 35 milliseconds, up from 5 milliseconds in CGCAL. Next, as we want to reveal the dynamics of this settling, the lateral in-

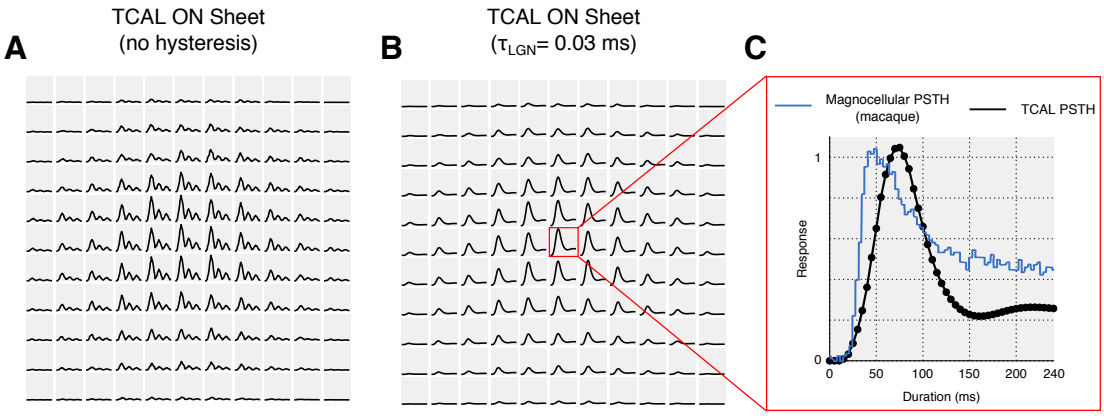


Figure 6.4: **Activity profiles of TCAL ON sheet units, with and without hysteresis in response to a Gaussian spatial pattern presented on the photoreceptor sheet.** (A) Oscillatory behavior after tuning lateral inhibitory parameters in the ON and OFF sheets. (B) Smooth activity profiles after the application of hysteresis. (C) Comparison of activity profile of central unit marked in (B) with average of experimentally recorded PSTHs in macaque. Blue trace shows average PSTH profile of 80 parvocellular LGN cells. Black trace shows TCAL activity profile with circles marking simulated sampling rate every 5 milliseconds. Experimental PSTH reproduced from Maunsell et al. (1999).

hibitory strength is boosted from 0.6 to 8.0. The result of these modifications are shown in Figure 6.4A using the same analysis procedure introduced in Figure 6.1.

What these profiles show are the TCAL ON responses without hysteresis, which now look very different from the step profiles of Figure 6.1A and C. These oscillations do not yet look like PSTH profiles, as their gradient changes discontinuously during the settling process, but the shape can be changed by introducing an appropriate level of hysteresis.

In CGCAL, the hysteresis constant was set to 0.05 $ms^{-1}$ to try to replicate the step shaped profiles of GCAL. In TCAL, this hysteresis constant is slightly reduced to 0.03 $ms^{-1}$, resulting in the profiles shown in Figure 6.4B. These profiles now do appear to be similar in shape to smooth PSTH profiles.

The central PSTH profile of Figure 6.4B is shown in more detail in part C where it is compared to the average magnocellular LGN PSTH in macaque (Maunsell et al., 1999). Although the fit is not perfect, it is certainly plausible and it is far more realistic than the responses of previous developmental models, such as those shown in Figure 6.1.

### Biological origins of the PSTH profile

It is remarkable that PSTHs emerge from the simple application of hysteresis using a mechanism originally introduced to achieve robust orientation map development. There is plenty of evidence for contrast-gain control and similar mechanisms (see Carandini and Heeger 2012 for a review), but is this a reasonable explanation for the temporal pattern of the firing rate response?

The temporal firing rate profile of a cell is a function of its particular internal biophysics and the way the inputs from the surrounding network arrive at the cell over time. It is clear that the interaction with surrounding cells is going to be an important component in determining the PSTH profile for a cell's response.

In TCAL (as for GCAL and CGCAL), these network interactions are modeled as lateral inhibitory settling in the ON and OFF sheets. Each of these sheets represents the combined properties of retinal ganglion cells and lateral geniculate nucleus cells with ON and OFF difference-of-Gaussian receptive-fields. It would be possible to apply one contrast-gain–control projection across both populations, but this would not affect TCAL's ability to generate these PSTH profiles.

What the 35 millisecond delay might represent is some average time constant for action of contrast-gain control across space. What is essential for this explanation to work is that the cell's activity is allowed to rise to reach its maximum firing rate, before the contrast-gain control mechanism has time to settle and bring the response back down to an appropriate, sustained level. The key idea is that the lateral network effects only act to reduce the response of the cell after it has reached a high firing rate due to the afferent drive.

What is clear is that lateral interactions between cells across space must exist in the visual system if the firing rate at one retinotopic position is to affect the firing rate at a different retinotopic position. Where these interactions occur is not specified by the TCAL model as the retinal ganglion cell population and the cells of the lateral geniculate nucleus are collapsed together according to their receptive field types in the ON/OFF sheets. The PSTH profiles are then explained by how fast the network interactions that implement contrast-gain control are able to act.

Whether or not this explanation is correct, two things are clear: (1) lateral inhibitory settling with hysteresis is sufficient to generate plausible PSTH profiles in TCAL, and (2) as long as the input firing-rate profiles are realistic in shape, the behavior in the cortical layer is independent of the underlying mechanisms that determine PSTH shapes

in the ON/OFF sheets.

TCAL is a cortical model that requires realistically shaped input profiles, which could hypothetically be generated by a sufficiently detailed phenomenological model. The means by which TCAL does generate PSTH profiles in the ON/OFF sheets is both extremely simple and sufficient for the purposes of cortical modeling, independently of whether lateral inhibitory settling is the true cause of the PSTH profile shapes.

#### PSTH profiles in the V1 sheet

With the temporal response profiles in the ON and OFF sheets calibrated, absolutely no changes are required within the cortical sheet to obtain the V1 PSTHs shown in Figure 6.5. The only parameter that is modified is the afferent projection strength from the ON/OFF sheets to V1 which is boosted by a factor of five to compensate for the increased subcortical contrast-gain–control strength that reduces the afferent drive to V1.

This result suggests that a major component of the temporal, single-unit activity profiles in V1 is simply inherited from the corresponding response dynamics in the LGN. This property follows naturally in any mechanistic models where the evoked response in V1 is driven by the spiking afferent inputs from the lateral geniculate nucleus.

In all the developmental models we have considered, the properties of the ON/OFF sheets are fixed, with no subcortical plasticity. This means that the profiles shown in Figure 6.4 remain identical at all stages of development. In contrast, the cortical sheet is plastic, which means the profiles shown in Figure 6.5 will change over time as Hebbian learning updates the network structure and as the adaptive homeostatic thresholds vary over time.

In addition to plasticity, there is an initial diversity in the V1 PSTH profiles that does not exist in the ON/OFF sheets. This diversity is present before development, i.e. right after the model is initialized, as highlighted by the two profiles selected in Figure 6.5. The two profiles shown in the red and green boxes have differently shapes, unlike the incoming activity from the ON/OFF sheets that all has the same temporal profile, as shown in Figure 6.4.

This diversity of response profiles is a result of the initial, randomized afferent and lateral inhibitory weights and the competitive nature of the settling process driven by the lateral, Mexican-hat inhibition profile, shown in Figure 2.11C. When a cortical area is stimulated, the individual units compete to represent the stimulus by suppress-

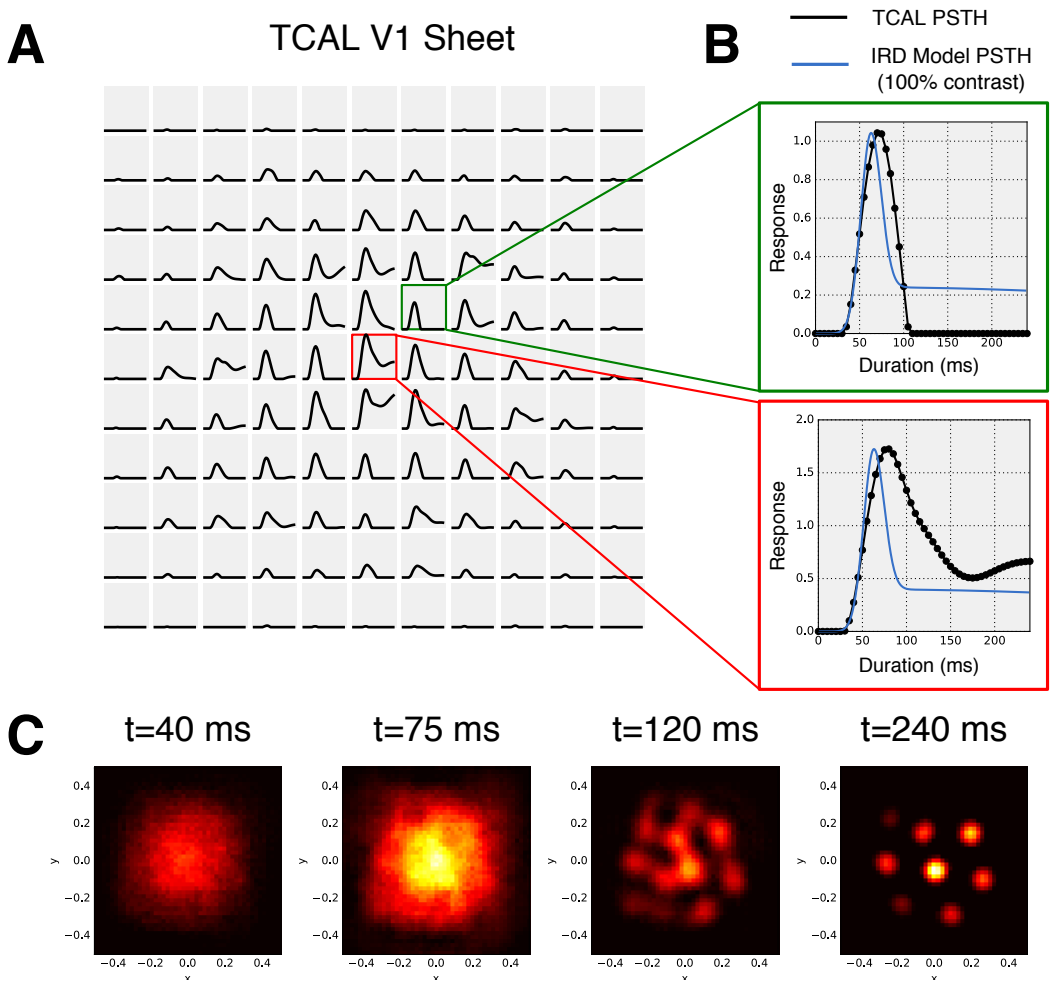


Figure 6.5: **Activity profiles of TCAL V1 sheet units at the start of development in response to an isotropic Gaussian stimulus pattern.** (A) An $11 \times 11$ spatially sampled grid of PSTH profiles in the V1 sheet of the TCAL model after model initializations. (B) Two PSTH profiles visualized in more detail, taken from the red and blue units shown in (A). Blue traces show an summary of the experimental data using the IRD model at $100\%$ contrast (Albrecht et al., 2002). The traces of the descriptive model are normalized to the response level of the TCAL model. The two units have been selected to demonstrate the initial diversity in responses in the TCAL model. (C) Bubble formation timecourse in TCAL, showing the activity over the entire V1 sheet. Between $t = 40$ ms and the maximum mean response at $t = 75$ ms the activity increases to a peak, roughly following the spatial Gaussian profile of the afferent input from the LGN. At $t = 120$ ms, the activity has started to decay and activity bubbles are starting to form. By the end of the simulated fixation at $t = 240$ ms, there are clear, settled activity bubbles.

ing their neighbors via lateral inhibition. Together with the randomized weights, this process breaks symmetry and leads to a corresponding diversity of activity profiles, even for a perfectly symmetrical stimulus such as the Gaussian input used in Figure 6.5.

Another way of viewing this diversity is to note the difference in responses between units *after* the PSTH peak, seen in Figure 6.5C. This subfigure shows the activity of the V1 sheet at four time points within the fixation, at 40, 75, 120, and 240 milliseconds respectively. The initial activity in V1 inherits the Gaussian spatial profile from the ON/OFF sheets, seen in Figure 6.4. This broad Gaussian activity peaks in V1 and then as the afferent input falls, lateral interactions start to dominate and activity bubbles begin to form.

This also explains why the overall profiles of the units across space, shown in Figure 6.5A reflect the shape of the Gaussian input on the retina, while also supporting activity bubble formation. The initial response that includes the peak is similar across the entire cortical population and only later do lateral interactions start to dominate. In other words, all the units that have afferent drive will respond with a similarly shaped initial peak in the firing rate, but only units that form part of an activity bubble will be sustained. Other neurons that are strongly suppressed by lateral inhibition will fall silent, as for the neurons initially responding but now in dark areas between the activity bubbles.

This effect is illustrated by the two selected PSTH curves in Figure 6.5B. Both units are near the center of the response and both units have a clear peak in their firing-rate activity. The firing-rate response of the unit in the green box falls to zero, while the response of the central unit in the red box falls after the initial but then begins to rise again as the bubble in which it is located sharpens.

One helpful way to understand this process is to suppose that the unit marked green is being suppressed laterally by the unit marked in red. This would happen if the green unit is outside the local excitation field of the red unit but within its larger inhibition field. This is simply the Mexican hat interaction that drives bubble formation (but remember, this is only an *effective* profile, not necessarily a fully mechanistic operation). For a description of the Mexican hat profile, see section 2.5.2.

We have now considered the dynamics of the evoked response in terms of the PSTH profile shapes in the V1 sheet and discussed how these PSTHs relate to activity bubble formation. Next we look at how the bulk activity dynamics across the cortical surface may be related to the experimental data.

### 6.2.3 Evoked population response

The spatial sampling of V1 response profiles shown in Figure 6.5A is very similar to the earliest version of the SIRD model shown in Figure 4.6B of Chapter 4. This illustrates exactly how the starting point of the SIRD model connects with TCAL, which can then guide future work. Using the extensive calibration embodied by the SIRD model, it is hoped that TCAL can be made into mechanistic, developmental framework that can also account for the spatiotemporal properties of the VSDI response.

Interestingly, there is an obvious discrepancy between the evolution of the spatial activity profile in TCAL, shown in Figure 6.5C and the spatiotemporal profile of the VSD response shown in Figure 2.6D (Sit et al., 2009). Although the peak TCAL response, shown in Figure 6.5C at $t = 75$ ms has a roughly Gaussian spatial profile just like the VSD response shown in Figure 2.6, TCAL's response then deviates from this initial Gaussian profile as activity bubbles form.

Although only the peak spatial profiles of the response are shown in the Sit et al. (2009) paper, it is possible to infer that the spatial profiles are smooth throughout the response by examining the plots in Figure 2.6D. The raw $\Delta F/F$ signal shown in Figure 2.7B uses a different experimental protocol and there is still no strong evidence that the spatial profile of the response dramatically changes shape as activity bubbles form.

Activity-bubble dynamics are a necessary feature of a developmental map model, and this is the first time they have been considered in the context of the VSDI response. If activity bubbles form in animal V1, why would they not be visible in the VSD signal?

First we will note that activity bubbles form on a similar spatial scale to the orientation hypercolumns, as it is in fact the size of the activity bubbles that determines the spatial scale of the map (Miikkulainen et al., 2005). By comparing the scale bar showing the orientation hypercolumn distance in Figure 2.3B and the scale of the VSDI signal shown in Figure 4.2, it is clear that the activity bubbles shown in Figure 6.5C are not shown on the same scale as the VSD signal.

Figure 6.6A shows the spatiotemporal response analysis of the TCAL model firing rate after the sheet area is tripled, putting the activity bubbles on a more suitable relative spatial scale. In part A, the firing-rate response is shown at the point when the bubbles are not very clearly visible at the peak of the response at 80 ms, clearly visible after 130 ms and at the end of the fixation at 240ms.

The top spatiotemporal plot Figure 6.6C illustrates how activity bubble formation results in a non-smooth spatiotemporal response in terms of the cortical firing rate,

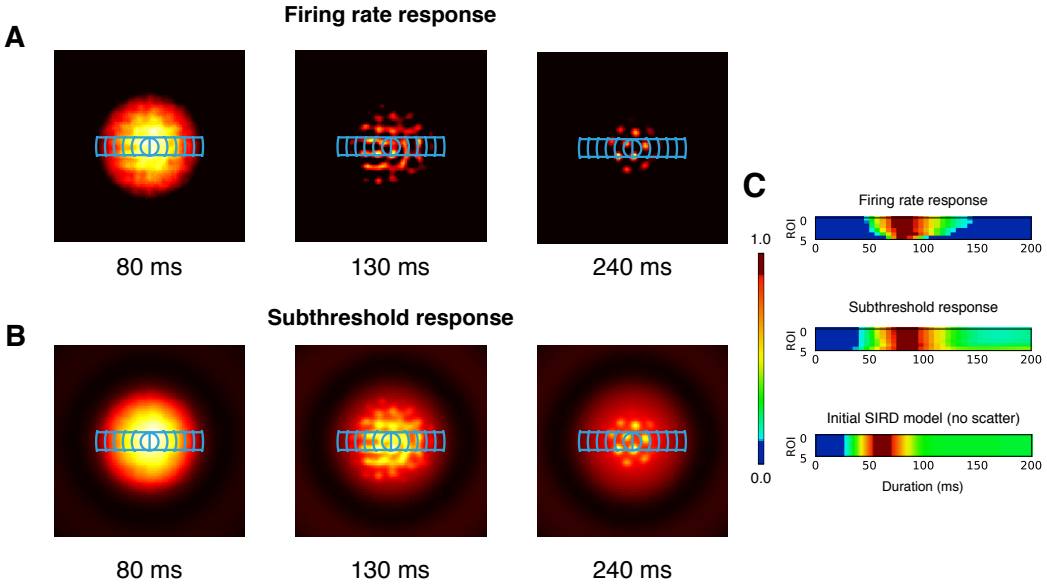


Figure 6.6: **Differences between the firing-rate response and the absolute sum of synaptic contributions per unit.** (A) firing-rate response at 80, 130, and 240 milliseconds after stimulus onset, showing where activity bubbles when they first form and at the end of the response. (B) Sum of absolute synaptic inputs across all projections in V1, weighted by the projection weights, at the same times after stimulus onset. (C) Corresponding spatiotemporal plots. The top plot shows the spatiotemporal analysis of the firing-rate response, the middle plot shows the spatiotemporal analysis of the subthreshold response and the bottom plot shows the spatiotemporal analysis at an early stage of the SIRD model, replicating Figure 4.6D. Note that there is still a response continuing to 240 milliseconds in the firing response as seen in (A) but it is not visible in (C) because of clipping of low values caused by this particular color map.

even after having rescaled the cortical sheet. Both the spatial profile at peak and the spatiotemporal plots are unlike those of the real VSD signal, as shown in Figure 4.2.

I.e., rather than the smooth Gaussian-like activity seen in Figure 4.6 of the SIRD model, TCAL suggests that the underlying activity pattern will be patchy. This result is not incompatible with the SIRD model, because spatial isotropy in that model was imposed directly from the observed spatial profile of the VSD signal, explicitly ignoring any smaller scale spatial modulation. What does demand explanation is why the spatial profile of a developmental model diverges so clearly from the observed properties of the VSD signal.

In TCAL, as in GCAL before it, the firing rate is approximated by applying a firing threshold to a subthreshold contribution, summed over the various types of projection to each unit. If this firing threshold value is ignored, the subthreshold activity can be loosely thought of as the membrane voltage. By considering the subthreshold contribution from each projection to a V1 unit (two afferent projections from the ON/OFF sheets as well as the lateral excitatory and lateral inhibitory projections) it is possible to try to relate the model more closely to the VSD signal than by using firing rates.

As discussed in Section 2.3.2, the VSD signal is generated by a voltage-sensitive–dye that binds to cell membrane and reflects voltage changes, not spiking. This signal reflects both excitatory and inhibitory activity, such that more inhibition will *increase* the VSD signal, even though it reduces the firing rate. This is a very important consideration, because the activity bubbles in GCAL and TCAL develop by inhibition of an initially broader pattern of activity that leads to inactive units between the active bubbles.

The simplest way to model the VSD signal in TCAL is to take the projection activities that correspond to the synaptic contribution, take the absolute value so both excitatory and inhibitory projections contribute positively, and sum these different contributions up, as weighted by the projection strengths in the model. The result of applying this operation is shown in Figure 6.6B.

Now the activity bubbles have a greatly reduced impact on the signal, even though they are still present at the firing rate level as in Figure 6.6A. Unlike the firing response, which can fall to zero as activity falls below the firing rate threshold, the subthreshold response is present across the entire spatial area for the duration of the evoked response. The corresponding spatiotemporal analysis is shown in the middle plot of Figure 6.6C.

There is now a good qualitative match with the bottom plot of Figure 6.6C which shows the top plot of Figure 4.6D. In other words, the responses in TCAL behave like

the earliest stages of the SIRD model, before latency scatter was introduced.

The match is not exact as the TCAL-neuron onsets do not quite align, and TCAL has a slightly greater time constant. Both these issues could be improved with additional parameter tuning and the match is certainly close enough to use the SIRD model as a guide to future work on TCAL. Note that the increased time constant may be partially due to the slightly slower response of TCAL units relative to the IRD model, as shown in Figure 6.5B, but it could also be partly due to averaging the responses in Figure 6.5A which are not all identical.

Improving TCAL's VSD signal model, parameter tuning, and investigating how to introduce the various forms of diversity motivated by the SIRD model are all items for future work, discussed in section 7.5. There is one way that TCAL now make an advance on the SIRD model which is to introduce a real diversity of tuning, taking a step towards explaining the tuning-dependent latency variance term, $\boldsymbol{\tau}_T$. This has not yet been shown in TCAL as we have only examined the initial state of the model. To consider tuning, we need the TCAL units to acquire tuning properties and for that, we now need to consider development.

### 6.2.4 Orientation map development

Having made significant changes to the activity dynamics in order to calibrate single-unit responses, it would not be surprising if TCAL was no longer able to function as a developmental map model like GCAL and CGCAL. The bubble formation seen towards the end of a fixation period, shown in Figure 6.5C, does suggest that the map formation process may still operate but it is impossible to establish this without running a developmental simulation.

In this section we see that TCAL is still a developmental map model, although one particular discontinuity that CGCAL eliminated from GCAL will be reintroduced for expediency, because certain properties of the evoked response interfere with the development process.

#### Learning driven by activity bubbles

Figure 6.7 shows the TCAL orientation preference and selectivity maps after 18000 presentations. The orientation maps are measured in the full model at 240 milliseconds after the onset of the sinusoidal gratings used to measure the orientation response. This approach in the full model is more comparable to what would be measured in

the animal than the analysis in Chapter 5, which was designed to reveal the afferent component of the orientation preferences to allow easier comparison between models. In all other aspects, the measurement protocol remains the same.

It is clear that although the orientation preferences, selectivities, and afferent weights have changed over development, the overall map organization and weight development is poor. This is most clearly illustrated by the lack of organization of the weights shown in Figure 6.7C, which are smooth but appear largely unselective.

Given that TCAL was calibrated only to have realistic PSTH profiles in the ON/OFF and cortical sheets, it is unsurprising that development has failed to proceed in a plausible way given how radically the activity profiles have been modified. The question now is whether plausible development can be recovered without losing the calibrated firing response profiles.

When discussing Figure 6.5C, the presence of the activity bubbles necessary for proper map development was identified but it was also noted that this settling only occurred after an unselective bloom of activity driven by the initial activity to reach the cortical sheet. With continuous learning, the high activity during this bloom will have a large effect on the synaptic weights, as opposed to the activity bubbles that have a much lower activity and form later on in the response.

This suggests a possible way to recover development by introducing one of the mechanisms that was eliminated to make CGCAL into a model without temporal discontinuities. If snapshot learning were reintroduced at the point where activity bubbles have formed, perhaps the developmental process can be recovered without sacrificing the earlier PSTH calibration?

Figure 6.8 shows that this is indeed possible. With snapshot learning triggered at 130 milliseconds into the response of each fixation, the orientation map structure after 18000 iterations looks far more plausible. Even more convincingly, the weight structure shows plausible organization even though these weights were noisy. Note that with snapshot learning reactivated, the learning rates once again match those of the GCAL model exactly, instead being divided by a factor of 20 as they were when continuous learning was applied.

As argued in Chapter 5, presenting a few orientation maps as in Figures 6.7 and 6.8 does not make for a convincing analysis. In that chapter, a quantitative approach was presented to assess developmental models of orientation map formation. It now makes sense to use that approach for comparing development in the TCAL model with and without snapshot learning.

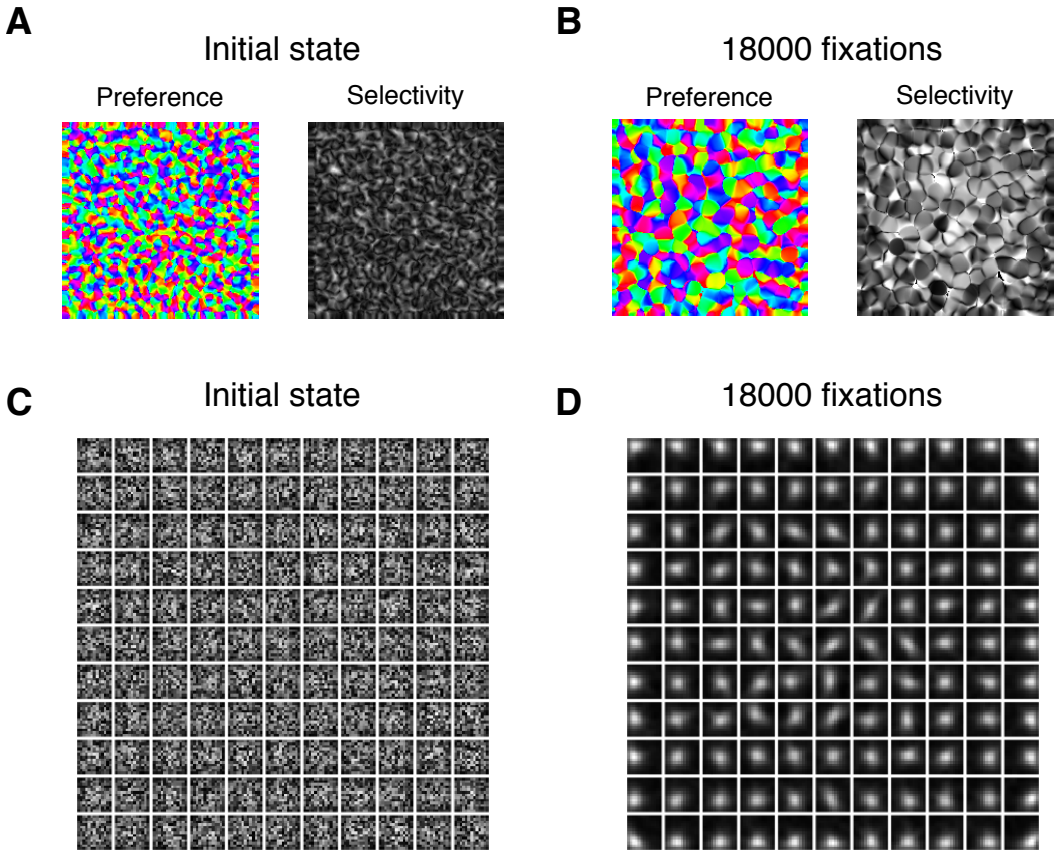


Figure 6.7: **Development of orientation maps and afferent weights in TCAL with continuous learning** (A) Orientation selectivity and preference map of the full model after initialization. The apparent structure in the preference map is due to the Mexican hat interactions in an unorganized network, as illustrated in Figure 2.11C, causing bubbles to form but not in any reliable way. (B) Corresponding map measurements after simulating 18000 fixations. The map structure is poor overall, with sharp discontinuities, but some neurons are highly selective. (C) Afferent weights from the ON sheet to the V1 sheet for an $11 \times 11$ regular sampling of units after initialization. The random initial state of the network is visible. (D) The same afferent weights from the ON sheet after 18000 fixations. The weights are now smooth but only a few of the neurons have become selective.

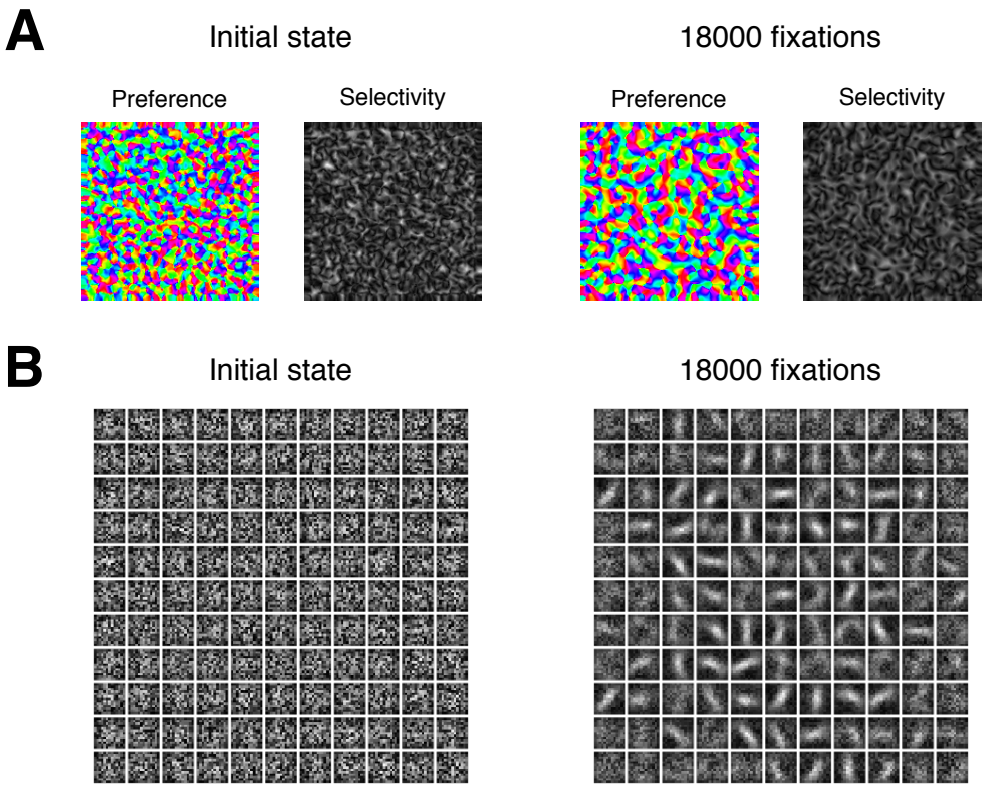


Figure 6.8: **Development of orientation maps and afferent weights in the TCAL model with snapshot learning after** 130 **milliseconds** (A) Orientation selectivity and preference map of the full model after initialization. As the same initial random seed was used as in the previous figure, these maps are identical to those in Figure 6.7A. (B) Corresponding map measurements after simulating 18000 fixations. The map structure is now much more realistic than before although selectivity is poor. (C) Afferent weights from the ON sheet to the V1 sheet upon model initialization that matches those in Figure 6.7C due to the use of an identical random seed. (D) The afferent weights from the ON sheet after 18000 fixations. The weights are still noisy, suggesting the learning rate could be increased, but now there is evidence of proper organization of the afferent weight structure.

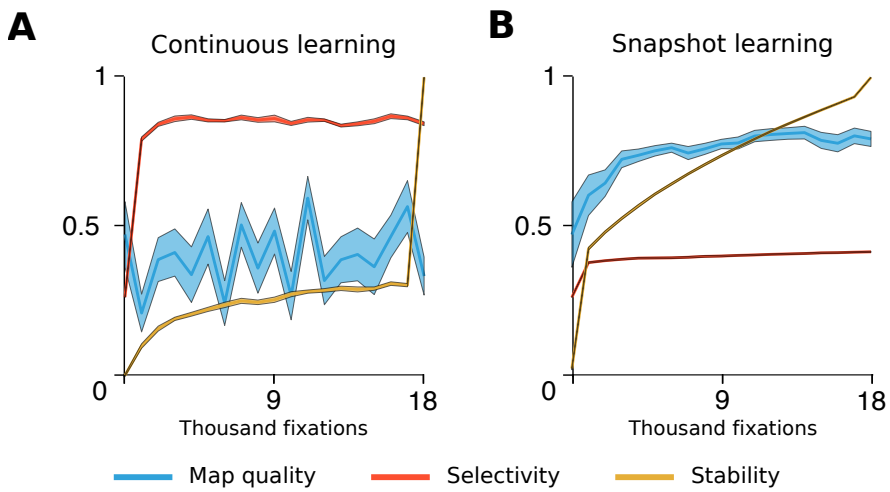


Figure 6.9: **Analysis of ten different randomized TCAL simulation runs over 18 thousand training presentations with and without continuous learning. Metrics shown are map quality, selectivity, and stability.** Maps were simulated with a $2.0 \times 2.0$ area that was cropped to a $1.75 \times 1.75$ area to reduce border effects. Solid lines indicate the mean values and the surrounding area indicates one standard error deviation from the mean. (A) Map quality, selectivity, and stability using continuous learning. Selectivity grows quickly but the map quality metric indicates no proper map organization and stability is very poor. (B) Map quality, selectivity, and stability using snapshot learning after 130 milliseconds. Selectivity is now lower but both the stability and the map metric values have improved. Note that the average Hebbian learning rates match those of the GCAL model, with the continuous learning rates rescaled appropriately.

Figure 6.9 shows the results of this analysis using ten simulations for 18000 training presentations using ten different random seeds to control weight initialization and training sequence. The mean map quality, selectivity, and stability metrics described in Chapter 5 are shown, including the standard error from the mean. In A, the results are shown using continuous learning while in B the results are shown for snapshot learning after 130 milliseconds.

The results confirm the analysis shown in Figures 6.7 and 6.8, showing that snapshot learning helps development proceed more quickly and reliably, reaching a greater level of organization within the simulated time period. With snapshot learning, the stability and the map quality metric are greatly improved. Although the map metric is still not as good as GCAL's perfect $\pi$ pinwheel density, the way the metric increases over development demonstrates that the same self-organization process is at work in

TCAL.

The results shown in Figures 6.8 and 6.9 are remarkable when considering the ways TCAL is both different from and similar to GCAL: (1) Unlike GCAL, TCAL has calibrated firing rate profiles in the LGN and V1 sheets, which now have a very different shape. (2) Only one additional parameter is needed, namely the hysteresis time constant in the ON/OFF sheets. (3) The only modified parameters in the ON/OFF sheet are the strength of the contrast-gain control mechanism and its delay. (4) All parameter values are equivalent to those in GCAL except for the afferent projection strength to V1, which has been boosted by a factor of five to compensate for reduced activity in the LGN.

Additionally, it is likely that the map quality could be further improved by either increasing the number of training steps or boosting the learning rate, in order to get rid of the noise in the weights shown in Figure 6.8B after development. It is clear that with additional parameter tuning and training iterations, it will be possible to improve both the PSTH calibration and orientation map development in TCAL simultaneously.

What is not clear is whether the re-introduction of snapshot learning has some biological justification, or whether it is indeed necessary. It is possible that a constant, continuous learning rate simply means map development occurs much more slowly, due to the way the activity bubble response is dwarfed by the initial unspecific response. Alternatively, the snapshot learning mechanism could be replaced by a continuous mechanism that modifies the learning process. For instance, perhaps learning is quicker when the network is quiescent or approaches a stable steady state, or there are some other mechanisms that result in reduced learning of the initial unspecific response. Investigating the ways snapshot learning can be eliminated in favor of a continuous mechanism and finding the corresponding biological justification (if any) is a topic for future work, discussed in section 7.5.5.

## 6.3 Bridging timescales

Figure 6.10 shows how TCAL bridges the timescale of development with the timescale of the evoked response within a single, unified model. All the results shown in this figure are drawn from the same map simulation shown in the previous figure. As a result, A, B, and C match the results shown in Figure 6.8A after zero and 18000 training iterations. In addition, Figure 6.10 shows the same results at two intermediate stages of the simulation run, after 6000 and 12000 iterations respectively.

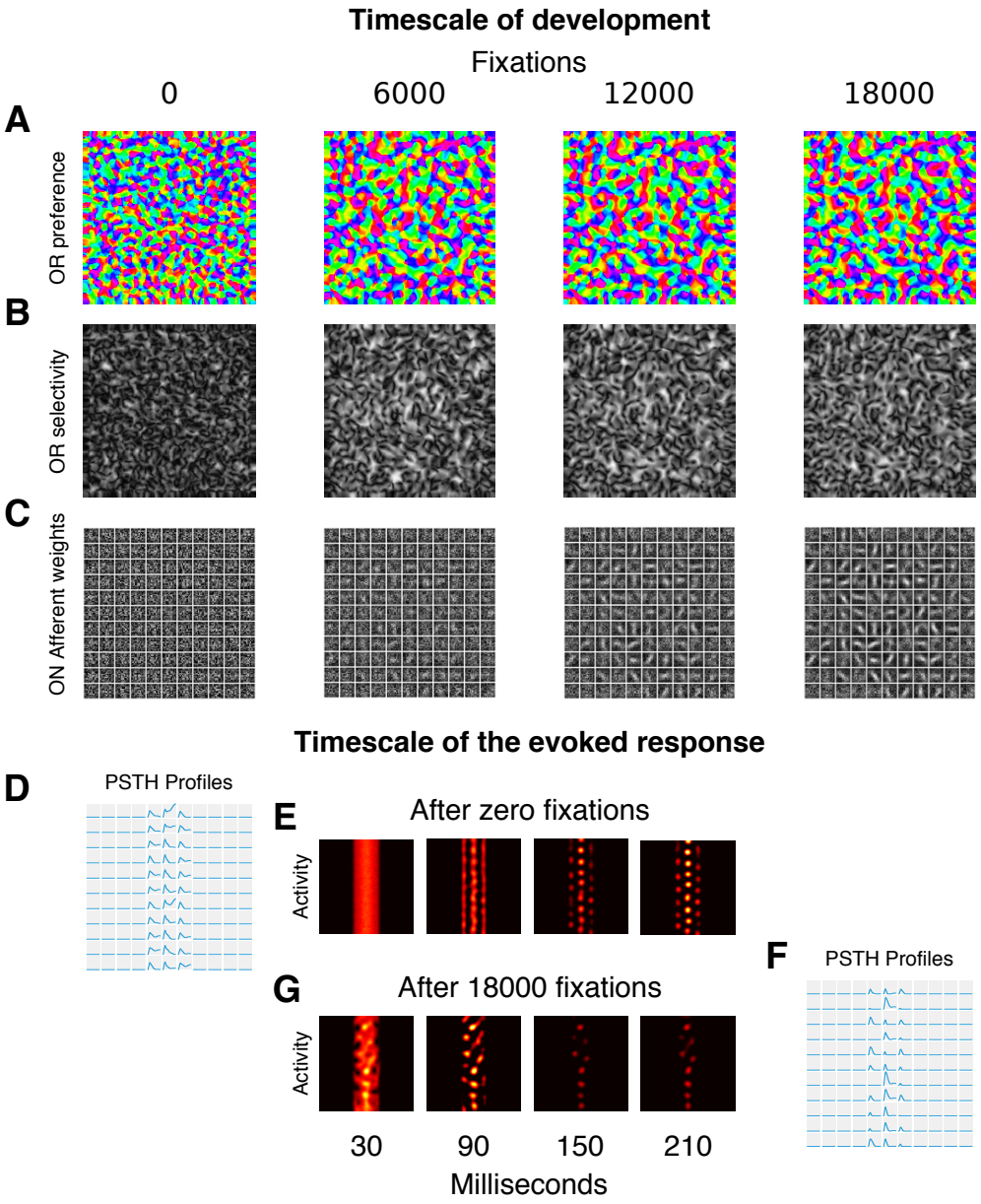


Figure 6.10: **Illustration of how the TCAL models bridges across temporal scales** (A) Orientation preference maps recorded after zero, $6000$, $12000$, and $18000$ fixations. (B) Corresponding orientation selectivity maps. (C) Afferent weights from the ON sheet for a regularly sampled $11 \times 11$ selection of neurons in the V1 sheet. After $18000$ fixations, there is self-organization from the random initial weights shown at fixation zero. (D) Corresponding PSTH profiles for the same $11 \times 11$ sampled units in response to a vertical line stimulus after zero training fixations. Each PSTH response is measured for a $240$ millisecond duration, corresponding to the duration of a fixation (E) Activity response of the entire V1 sheet at $30$, $90$, $150$, and $210$ milliseconds in response to the vertical line stimulus. (F) Corresponding PSTH profiles for the same $11 \times 11$ units after $18000$ training fixations. Note that the PSTH profiles have changed due to the developmental process and that the responses are now more diverse. (G) Activity of the V1 sheet to the same vertical line stimulus after $18000$ training iterations, shown at the same time points in the evoked response. The activity bubbles at the end of the response now reflect areas that are selective to the vertical orientation, and the dark areas in between represent areas selective for other orientations.

In Figure 6.10D, we observe the evoked response before training in response to a vertical line stimulus, for exactly the same regular sampling of units used throughout this figure and shown in Figure 6.9B. The response is initially unselective as the network has not been trained. Figure 6.10E shows the activity of the entire V1 sheet at four different times, illustrating clearly the initial unselective bloom after 30 milliseconds and the subsequent activity bubble formation. By 210 milliseconds, a fairly regular array of activity bubbles has formed.

In Figure 6.10F and G shows the corresponding measurements of the evoked activity at the end of the developmental run, after 18000 training pattern presentation or fixations. The evoked response properties as sampled over 240 milliseconds and shown in F are now different from the responses shown in D. This is because the weights have now self-organized through Hebbian learning and the units have now become orientation selective, and have structured receptive fields as evidenced by the afferent weight structure at the corresponding developmental point shown in C. This difference in response also reflects the tuning of the units, as vertically selective units will be most strongly driven by this vertical line stimulus.

The effect of development on the evoked response can also be seen in G, which shows the activity of the V1 sheet in response to the same stimulus and at the same time points shown in E. The regular bubble position and overall uniformity of the response present before training no longer applies. Once again, this is due to the diverse tuning properties of the V1 units that have emerged over the course of development.

The difference between Figures 6.10D and F illustrates the core idea of this thesis. Both are examples of evoked responses to a visual stimulus, of the sort that might be recorded experimentally with an electrode array. As such experimental recordings are typically performed in the adult animal, an experimenter is likely to observe a diversity of responses and tuning properties, as shown in Figures 6.10F. These models predict that this diversity reflects the long-term relationship between these neurons and their visual environment, encoded in a pattern of weights and network structure that determines how the neurons will respond to a new stimulus.

## 6.4 Discussion

What is unique about the TCAL model is in the particular way it accounts for this diversity of evoked responses and tuning properties in the evoked response. Activity is computed per unit by integrating neural activity projected through weight fields such

as those shown in Figures 6.10C. These weight fields are not imposed by attempting a direct calibration with experimentally determined connectivity profiles recorded from adult animals. Instead, they are self-organized through a developmental process, driven by tens of thousands of visually evoked responses during training.

Other than the re-introduction of the snapshot learning process, TCAL is a continuous model. This means every simulation timestep follows the same rules, computing activity responses and very gradually shaping the network structure through Hebbian learning. In this way, the difference in the evoked response shown in Figure 6.10D and Figure 6.10F is a reflection of the 18000 evoked responses that occurred during training that separates these two states of the model. As the statistics of the training patterns used will affect the development of the network, we can now see how TCAL meets the stated aims of this thesis, listed at the start of Section 1.1.

To recapitulate these aims, expressed in terms of what the TCAL model achieves: (1) Each V1 unit responds in the context of a surrounding population to which it is connected. (2) The history of evoked responses, starting from the initial conditions explains how this specific pattern of connectivity arises. (3) This history of evoked activity is driven by visual input to the model, and so the connectivity of the model reflects the long-term statistics of the visual environment.

In addition to showing a concrete example of how the stated goals of this thesis were achieved, Figure 6.10 shows how TCAL is connected to the work in two of the previous chapters. Only the analysis developed in Chapter 5 with respect to the pinwheel density map metric is not shown, as this was demonstrated in Figure 6.9.

The first link is with Chapter 3, as the results shown in this figure were taken from one of ten different simulation runs analyzed in Figure 6.9, launched with Lancet and visualized with HoloViews. HoloViews generated these plots and allowed interactive exploration of all the results across all the simulation runs, before they had even completed execution on the compute cluster.

This figure then connects with the work in Chapter 4, as subfigures D and F show regularly sampled PSTH profiles across space in a very similar way to those presented at the start of the SIRD model, shown in Figure 4.6B. The VSD signal model shown in 6.6 could also have been applied to the responses shown in subfigures E and G. This would have allowed the same spatiotemporal analysis and plots to be generated as used throughout Chapter 4. These plots would show the response to a vertical line stimulus and not a Gabor path and the responses shown would have been computed by a mechanistic, developmental model.

## 6.5 Conclusion

TCAL is a mechanistic, developmental model that connects developmental processes to the evoked response. It features calibrated firing rate profiles in both the subcortical and cortical sheets. In the subcortical sheets, calibration is done with respect to experimentally derived PSTH profiles and in the V1 sheets, calibration is done using by the IRD model which is also the basis of the SIRD model presented in Chapter 4. TCAL also feature developmental self-organization including the formation of smooth orientation maps and the emergence of varied tuning properties across the cortical population.

All this is achieved by the introducing a single hysteresis time constant and very simple damping mechanism to the basic structure of the GCAL model that precedes it. This addition is counteracted by the elimination of various discontinuities in temporal processing of the GCAL model, making the final definition of TCAL conceptually and mathe atically simpler than that of GCAL, though it does take much longer to simulate.

The result is a unified, mechanistic model that computes cortical activity in response to a stimulus using a firing rate representation. These evoked properties make contact with firing rate data and all the experimental calibration embodied by the SIRD model, presented in Chapter 4.

Using Hebbian learning, this evoked activity gradually shapes the connectivity of the network, starting from a random initial condition. This results in a model of development that can be analyzed using the techniques presented in Chapter 5.

These results shows how TCAL allows two different experimental literatures to be unified into a single model. In particular, all the evoked response literature described in Section 2.3 can be used to calibrate TCAL, and so can all the developmental literature described in 2.4. The result is a unified model across spatial and temporal scales that can serve as a new platform for future research.

# Chapter 7

# Discussion

The goal of this thesis is to build a single, unified model that can account for the evoked response of an extended population of cortical neurons in the primary visual cortex, over developmental timescales and with millisecond resolution. In this framework, the evoked response is computed in a network that is shaped by the entire history of visual input. This unified approach is designed to preserve the reciprocal link between activity dynamics and the gradual, developmental changes to the network structure over time.

The contributions and implications of the work presented in each results chapter will be now discussed, followed by suggestions for future work. A wide breadth of topics and models has been covered in this thesis and the aim of this chapter is to highlight the links between them, showing that the total contribution of this work is more than the sum of its parts.

Each chapter has tackled a different scientific problem that is independently meaningful, and indeed each chapter has been published or will be published independently. Yet when taken together, this work points to a new modeling approach that allows activity dynamics and self-organized neural structure to be unified within a single, consistent framework.

## 7.1 Firing-rate and spiking models

Throughout this project, the activity of cortical neurons has been modeled at the firing-rate level, ignoring the more detailed biophysics that can be achieved by spiking, compartmental approaches. The trade-offs between these two representations of neural activity will be discussed in the context of the SIRD and TCAL models.

First the SIRD model was introduced, using firing-rate units to link the single-unit responses to the population response. In this instance, representing activity with firing rates was appropriate because (1) the great majority of the available experimental data that can be used for calibration has been reported at the firing-rate level, such as the PSTH profiles captured by the IRD model, (2) detailed cellular biophysics would not have been a sensible match to the other simplifying assumptions in the model regarding the form of population level dynamics and applicable diversity of temporal responses, and (3) a firing-rate simplification makes it far less computationally expensive to simulate a large population of units to correspond to the V1 area observed in VSDI studies.

This is not to say that spiking models do not have a useful role to play. The latency scatter mechanism in the SIRD model was partly inspired by previous work on the VSDI signal that used spiking compartmental model (Chemla and Chavane, 2010a). It is worth considering what benefits there would be if this kind of spiking, compartmental model were expanded spatially in order to simulate several square millimeters of cortical area.

Firstly, it would be possible to identify and quantify the detailed, mechanistic contribution made by various processes to each form of temporal diversity identified by the SIRD model. For instance, latency scatter could be explained for in terms of varying axonal lengths, differences in the propagation speed of action potentials, laminar differences and delays, as well as the biophysical processes within the individual cells. In addition, it would be possible to introduce a realistic diversity of cell types with different temporal profiles such as regular spiking, fast-spiking, or bursting behaviors.

If properly calibrated and constrained, such a model would directly account for all the forms of temporal response diversity introduced by the SIRD model, expressed in terms of biophysical mechanisms. Unfortunately, it is not feasible to run such a large model without consuming significant supercomputing resources. Even if the necessary computational power were available, there is currently far too little experimental data suitable for constraining all the necessary biophysical parameters.

By adopting a firing-rate representation, the SIRD model loses biophysical detail, but becomes far more convenient to simulate on available computers and using available experimental results as constraints. In addition, SIRD is expressed in a very simple way that makes it easier to understand and systematically analyze each hypothesized contribution to the VSDI signal. A detailed spiking equivalent to this model could explain the origin of the experimental data that was used to calibrate the SIRD model,

but only in terms of a greater number of more fundamental biophysical constants.

Using firing-rate units also allows the SIRD model to connect to other models using a similarly simplified representation of neural activity. As the activity state of each unit is expressed by a floating point number, a firing rate model like the SIRD model can be easily be related to other firing-rate models such as TCAL. If a detailed compartmental model were used, it would be more difficult to bridge across models, because each cell would have a complex state, defined by dozens of biophysical variables. Ensuring two detailed models mesh together in a consistent way is likely to pose a greater challenge than when the firing rate representation is used.

The immediate reason that TCAL uses firing-rate units is historical, because TCAL inherits nearly all of its structure from the GCAL model. Like nearly all such developmental models, GCAL is based on firing rates, so that it can simulate long timescales. Not only is an individual spike unlikely to be an important event on long developmental timescales, the computational cost of running a detailed, developmental model at the spiking level would be prohibitive.

What the TCAL model shows is that it is possible to express additional, biologically relevant information using the firing-rate units of a developmental model without requiring a huge increase in complexity or sacrificing the process of gradual self organization. The purpose of TCAL is to increase the temporal resolution to the limit where the firing-rate representation begins to break down.

By definition, TCAL does not represent the individual spikes that would be recorded in an experimental setup, but it still can make direct contact with experimental data every time experimentalists choose to use a firing-rate approximation. This simplification happens frequently, as it is difficult to express a large number of precise spike timings in a publication, due to the volume of data involved. Whenever a experimenter needs to summarize average spiking activity, for instance by computing a PSTH and selecting an appropriate temporal bin size for the spikes, it should be possible use this data to calibrate a firing-rate model like TCAL.

There are a few downsides with how the firing-rate units of TCAL are implemented, and in particular when it comes to expressing the various types of temporal diversity identified by the SIRD model. TCAL inherits its structure from GCAL and is also implemented in the Topographica simulator, used to simulate the development of neural sheets of firing rate units. Each sheet defines a population of neurons and connections, known as projections, propagate activity between them. This architecture is well optimized for simulations such as GCAL, because it exploits similarities be-

tween all the units in a projection, but it does make it more difficult to add diversity, such as distance-dependent lateral connection delays within a single projection. Modeling these effects is of course not impossible in Topographica, but it is less efficient and more difficult to set up. Some workarounds for these difficulties are suggested as future work items in sections 7.5.4 and 7.5.6.

## 7.2 Building a unified model

In any unification, connections need to be established between different threads that initially appear unrelated. On the surface, each of the following topics tackle different problems in neuroscience: (1) establishing a reproducible workflow that is flexible and allows easy exploration, (2) relating single-unit electrode recordings to the voltage-sensitive–dye response, (3) automatically assessing the plausibility of developmental simulations of orientation map formation, and (4) achieving plausible PSTH profiles in a developmental model. This section shows how these threads connect together.

Each of these research threads address separate aspects of one underlying problem, which is to construct a cortical model that is unified across spatial and temporal scales. In this section, the relationship between these research topics will be discussed and summarized to show how these different ideas fit together.

Taken as a whole, this thesis is not intended to be an endpoint to any particular line of research, but is instead intended to open up new avenues for exploration. A model that can operate on developmental timescales with plausible activity dynamics will enable more effective exploration of an entire class of research questions. Previously, such questions could not be addressed by any single modeling framework and had to be tackled in a patchwork manner. Given how sparse the experimental data is, being able to draw constraints from a diverse set of experiments is a major potential advantage. TCAL shows that this gap can be bridged, using data obtained from developmental studies as well as from evoked activity recordings in the adult animal to calibrate the same model. It is the goal of this thesis to ensure this process can continue, by building a solid foundation for many exciting new research projects.

To allow future researchers to build on this platform, it is essential the work is open, extensible, and reproducible. The nature of the research workflow developed in Chapter 3 will be discussed in the next section. The scientific workflow is important, but it is not the only factor that is important for enabling new research. A solid foundation for future models must be as simple as possible, as a model that can be easily

understood will be easier to work with and will encourage other researchers to engage with the work.

The GCAL model analyzed in Chapter 5 is simpler, more robust and uses more biologically plausible mechanisms than previous self-organizing models of orientation map formation. This model is the basis of the TCAL model presented in the final chapter. Although TCAL introduces one new hysteresis mechanism, it also eliminates various optimizations used in developmental modeling that disrupt continuous temporal processing, thereby simplifying things further. TCAL introduces one very simple equation for hysteresis but otherwise the overall definition is simpler than that of GCAL, which in turn is simpler than the models than came before it. The goal is always to build a model that can explain more with less, just as the goal with reproducible notebooks is to achieve more results with less code.

Of course, simple models are good, but simple models that can be well validated are even better. The map quality metric in Chapter 5 served not only to validate the GCAL model, leading to its publication, but it also serves as a way to validate any developmental map simulation involving orientation maps. Without this metric, the analysis of orientation map development relies only on subjective judgment. By introduced a clearly defined, automated map-quality metric, it becomes possible to validate developmental map models in a more quantitative way. This proces was demonstrated at the end of Chapter 6 when the developmental properties of the new TCAL model were investigated.

A map-quality metric is also important for a unified model, as it establishes an objective upper bound on map quality, namely $\pi$ pinwheel density. This metric makes it easier to determine the trade offs that must be made within the model. For instance, if the pinwheel density of the orientation maps is already close to $\pi$, as it is for GCAL, there is no way to further increase the map quality metric. This information allows a researcher to focus on optimizing other features of the model, such as improving the realism of the evoked response dynamics instead.

Finally, a link has been established between TCAL and the SIRD model, which offers a guide for future research. The SIRD model was introduced as a way to relate the evoked response of single units to the population response as measured with voltage-sensitive–dye imaging. In doing so, the SIRD model highlights the forms of diversity necessary to transform a homogeneous population of firing rate profiles into a spatiotemporal distribution that accounts for the VSDI response.

The initial assumption of the SIRD model connects to the PSTH profiles of the

TCAL model as shown in Figure 6.6. Whereas the SIRD model made use of the phenomenological IRD model for describing a population of firing rate profiles, TCAL is a mechanistic, developmental model with a subcortical pathway and units that develop their own unique tuning properties. Guided by the SIRD model, it should be possible to account for the VSDI signal using TCAL in future work. Such a model would start with the stimulus on the photoreceptor sheet, propagate the activity through the LGN and to V1 units using specific afferent and lateral weight profiles learned from the statistics of the environment, and then compute the firing rates and VSD signal contribution in the cortex. Such a model would offer a new, unified understanding of V1 that can make contact with a huge body of experimental literature, allowing the model to become increasingly well constrained and calibrated, able to account for an ever larger body of observations with fewer assumptions and mechanisms.

## 7.3 Research tools and reproducibility

As argued in Chapter 3, it is essential to improve research productivity as well as reproducibility, if robust work practices are to be widely adopted. The original motivation for creating Lancet and HoloViews was to facilitate the work presented in this thesis. Due to their carefully general and modular design, these tools are not specific to this particular project, and are now being used around the world by different researchers in different domains. These researchers can now carry out their work more efficiently using the Jupyter notebook, a literate programming environment that can be used to improve scientific reproducibility.

Productive open-source research tools are extremely valuable to science. In general, the impact of software tools scales with the number of corresponding users: good tools can magnify productivity gains and bad tools can cause innumerable difficulties. Using an open source (BSD) license, both Lancet and HoloViews have an unlimited potential number of users. As their popularity grows, these tools may potentially make a significant impact on the scientific Python ecosystem.

Both Lancet and HoloViews help improve reproducible data exploration, but there is a natural reluctance of researchers when considering new research tools. Researchers are focused on exploring and investigating scientific questions and may not be programmers who have the necessary skill to make improvements to the software they use daily. As a result, people often reach for the software tools that are available and familiar.

Tools are often used not based on their own merits but because they are already in use with a research group, because migrating to better tools is difficult. Keeping a workflow that is suboptimal but familiar often appears to be a rational decision in the short term. Yet in the long term, the difficulties associated with using inferior software environments can be devastating to scientific productivity and reproducibility, especially now that there are much better alternatives.

Poorly designed research software demands that researchers write large amounts of code without succinctly expressing intent. This wastes time, reduces research productivity, and impedes clear scientific communication. If the software is proprietary, researchers may face an additional financial cost as well as licensing issues which prevent their work from being freely used by an interested party, and the work may be unable to be used should the license owner cease to support it.

Every line of unnecessary code is a liability that ends up limiting the space of scientific possibilities that a researcher can contemplate. Unnecessary, untested code is typically very fragile and breaks often. When such code does run successfully, it can contain errors that are difficult to catch, increasing the probability of incorrect results winding up in a final publication. In the worse case, incorrect findings are disseminated throughout the research community for a long time before they are identified and fixed.

Both Lancet and HoloViews are designed to express scientific intent with less code. Using these tools, more research can be done quicker and captured in the context of a reproducible notebook. In my own experience, HoloViews enables the rapid creation of complex, interactive visualizations that would otherwise take far too long to implement. With complex visualization tasks expressed by a minimal amount of readable code, there are completely new possibilities for visualization and analysis that would otherwise be too impractical to consider.

The availability of these tools has had a major impact on the scientific content of this thesis. In numerous cases, new visualizations for interactively exploring the SIRD and TCAL models could be expressed very rapidly, allowing new ideas to be explored. The cost involved in developing these tools has paid off scientifically, even in the context of a single PhD thesis. Launching a large batch of TCAL simulation jobs with Lancet is easy, and being able to interactivity visualize the entire dataset with HoloViews, while it is still being generated on a compute cluster, is an invaluable boost to productivity.

Improved, reproducible workflows such as these are not ancillary goals, when considering large scale models of cortical function. As models become larger, and are

calibrated against more and more experimental data in order to account for more biological details, effective tools for the generating, collecting and visualizing data are absolutely essential. As the number of parameters and different dimensions of the data expand, the easier it needs to be to slice and visualize this data across those dimensions. Suitable tools are required to help make scientific research easier, as without improved tools, making advances in research will only become more difficult over time.

Despite the payoff in power, there was of course a cost of a significant investment of time and energy to build and maintain these open source tools. It would be fair to say that the work presented in this thesis *could* have been achieved by done without them, although this would have meant working in a far less efficient and reproducible manner, leading to a result that would be far more difficult to build on. What has made this investment truly worthwhile is that Lancet and HoloViews are entirely general tools that are not explicitly tied to this research project in any way. This means they will continue to be used in the future, allowing this cost to be repaid many times over. They will continue to be invaluable in my future research and in addition, they will save the time and effort of all the people who recognize what they have to offer and use them to carry out more productive and more reproducible research.

## 7.4 Three different models

In this section, the implications of the three models described in this thesis are discussed, both in terms of what they have achieved as well as their main limitations.

### 7.4.1 SIRD model

The SIRD model expresses the VSDI signal in terms of a population of single unit responses that include different forms of temporal diversity. At each step, starting with the well-established IRD model, a mechanism is carefully introduced and calibrated against the experimental data, using the simplest justifiable mathematical formulation.

The contribution of this model is twofold. Firstly, the effect of each mechanism on the signal is analyzed and understood in isolation with reference to available experimental data. This makes it possible to dissect the various properties of the VSD signal in terms of individual mechanisms. Secondly, the SIRD model points to particular gaps in the experimental literature where insufficient data is available. That is to say, each time the SIRD model is forced to make an assumption, there is an opportunity for cali-

bration using new experimental data. If the necessary data is not available, this may be a motivation for new experiments. In this way, the SIRD model offers a framework for future experimental work to calibrate the model further and fill in the gaps, reducing the number of assumptions that need to be made and more directly relating single-unit and population measurements.

Although the SIRD model successfully accounts for various features of the VSDI signal, it does have limitations. In particular, SIRD model units do not have explicit receptive fields or feature selectivity. The various calibration steps had to be imposed on the model in order to account for the VSDI signal, without being able to trace them back to their detailed mechanistic origins. In other words, the SIRD model offers a mechanistic link from various experimental observations to the VSDI signal but does not attempt to explain anything further to show how these mechanisms themselves may be implemented. It is hoped that by linking the SIRD model with TCAL, these limitations can be addressed, as discussed in sections 7.5.4 and 7.5.6.

### 7.4.2 GCAL model

The analysis presented in Chapter 5 is what led to the publication of the GCAL model. This simple, robust model existed for several years before this research project began, but there had been no objective way of validating it. With the development and use of the $\pi$ pinwheel density map quality metric to assess simulated maps, the consistent high quality of GCAL orientation maps was convincingly demonstrated, in a way that has yet to be matched by any other developmental model.

This result illustrates why objective, automated metrics are essential. When relying on subjective judgment alone, it is difficult to tell a good orientation map model from a poor one, especially when selection biases come into play. Every researcher will attempt to present their model in the best possible light, and without a map quality metric, it is natural to pick the best subjective result for publication. This is a highly problematic practice, as even a poor model may occasionally generate maps that appear to be high quality, making it difficult to make a compare to a model that does consistently generate high quality orientation maps. The development process in animals is clearly very robust, and so a proposed mechanism that is not robust is clearly a poor explanation, which can be revealed through automated metrics.

As the GCAL model existed before the $\pi$ pinwheel analysis was developed, discovering $\pi$ pinwheel density in GCAL was a rigorous tests of both the model and the

analysis approach. No model parameters were changed in GCAL before it was tested and found to exhibit $\pi$ pinwheel density. Although the fact that GCAL has $\pi$ pinwheel density is a result in itself, this did not necessarily mean that the raw pinwheel density would be a suitable basis for a general, normalized map metric.

The analysis in section 5.3.3 shows that there is a high likelihood of elevated pinwheel density in low-quality maps which is what makes a general metric for simulated maps possible. As explained in Chapter 5, the stability and selectivity metrics alone are not sufficient for evaluating the developmental process of a model simulation and unlike experimental maps, simulated maps can potentially be anything. With all three metrics working together, there is now an objective way to assess any model of developmental model that develops orientation preferences and selectivities across cortical space.

The publication of the GCAL model itself has important implications. The model has superseded all earlier work based on the LISSOM model (Miikkulainen et al., 2005) as it is simpler, more robust yet develops according to the same basic principles of self organization. Even before it was published, several new models based on GCAL were already being constructed. Now that it has been published, these GCAL-based models, including TCAL, now have a clear reference to build on.

### 7.4.3 TCAL model

The TCAL model both extends and further simplifies GCAL. It extends GCAL by showing how a developmental model can simulate weeks of self-organization as the cortex matures while incorporating plausible PSTH profiles within each individual fixation. It simplifies GCAL by eliminating certain scientifically unnecessary optimizations, namely instantaneous contrast-gain control in the ON/OFF sheets and activity resets.

The fact that TCAL is able to make these simplifications is further validation of GCAL, by demonstrating that these optimizations were purely about improving computational performance and do not affect the fundamental results. This is true of all the mechanisms that were discontinuous with time except for snapshot learning, which can now be investigated properly as described in section 7.5.5.

In addition, TCAL shows how an existing component of GCAL, namely the contrast-gain control mechanism that was originally introduced to achieve robust orientation map development, is also able to generate plausible PSTH profiles. This means that

not only does TCAL show how to bridge timescales, it does so in a very parsimonious way. It also shows that an individual mechanism can have an impact on different dynamic processes operating on very different timescales.

TCAL is designed to include the minimum set of mechanisms necessary to achieve its goals. The process of activity integration coupled with the Hebbian learning rule are absolutely essential for simulating activity driven plasticity. The lateral interactions implementing the Mexican-hat profile shown in Figure 2.11 are necessary for smooth self-organization and orientation map formation. In addition, the subcortical pathway is very basic, constituting a photoreceptor sheet to define the visual stimulus as well as units with ON/OFF center-surround receptive fields, which could be implemented in a one step convolution. The contrast-gain control mechanism could be omitted for the purposes of development, resulting in a less robust model (Stevens et al., 2013b), but it is central to shaping the calibrated firing rate profiles in TCAL (along with hysteresis). Only the homeostatic adaptation mechanism in the V1 sheet that regulates cortical activity could be potentially be eliminated by manually finding a suitable threshold, although the resulting development would be far less robust.

TCAL thus defines a combination of simple, core mechanisms required to build a model that can account for receptive field formation, robust and smooth orientation map development, contrast invariant orientation tuning, and plausible single-unit PSTH profiles. Each mechanism has a well-established reason for being in the model. This ensures that not only does TCAL bridge development to the evoked response profiles, but it does so in as simple a way as is feasible.

Simplicity is essential, as the main goal of TCAL to serve as a proof of concept as well as a foundation for future work. There is now an entire literature that can be used to calibrate this developmental model that could not be used to calibrate any previous developmental map model. TCAL is therefore designed as an extensible framework that can incorporate new experimental calibration. In the next section, some possible directions for future work are considered along these lines.

## 7.5 Future work

The work in this thesis is designed as a platform for exploring new research questions that can only be tackled within a spatiotemporally unified cortical model. Using HoloViews and Lancet to extend the models developed here, it is hoped that the following scientific questions will now be feasible address. Each of the following proposals

may be considered as a self-contained research project.

### 7.5.1 Modeling both the VSDI onsets and offsets

The SIRD model intentionally focuses on the VSD signal in relation to stimulus onset. The basic reason for this is that the IRD model that the SIRD model is built on only describes the firing-rate profiles in relation to the appearance of a stimulus. Secondly, there is less experimental data available for calibrating offsets in general.

In TCAL, there is no reliance on the IRD model, and both stimulus onsets and offsets are processed in exactly the same way. In other words, there is an ongoing process of simulated activity that is continually being driven by whatever input is supplied by photoreceptor layer. Unlike the SIRD model, TCAL also features ongoing, recurrent lateral dynamics, and it would be worth investigating how these lateral dynamics impact the offset response, in order to relate it to the available experimental data.

### 7.5.2 Calibrated tuning dependent latency spread

One of the forms of temporal diversity that is introduced by the SIRD model corresponds to the latency differences between neurons that corresponds to variable feature tuning. This form of diversity is the least-constrained term of the SIRD model and it would be interesting to investigate how it could be quantified and constrained either by experiment or by improving the model.

Quantifying this unknown distribution may present an opportunity to guide the collection of more experimental data. All the parameters of the SIRD model could be better calibrated, but this one in particular poses a challenge. A starting point would be to record the latency of cells as orientation, spatial frequency, and phase are individually varied from the optimal tuning for the recorded cell. Then it would be interesting to vary all these tuning parameters together to try to understand the relationship of the general tuning latency term $\boldsymbol{\tau}_T$ to the latency variation defined by varying only a single visual feature at once.

From a modeling perspective, it is currently not known how this distribution should be accounted for. One possibility is to use a mechanistic approach by assuming the latency of a unit depends only on its synaptic input. Then, perhaps, all tuning dependent latency effects could be explained at once if the cells with more overall input (due to their applicable receptive field structure) respond quicker than those with weaker overall input.

One feature of TCAL that will help address this problem is the wide diversity of receptive fields that emerge during development. Considering orientation tuning as an example, it is clear that the orientation tuning of most TCAL units will not match any given oriented stimulus. The challenge then is to understand how the orientation of the unit relates to onset latency, and how this effect can be calibrated with respect to experimental data. If this issue can be addressed along with the suggestions in sections 7.5.3 and 7.5.4, it is hoped this calibration could be done with respect to VSDI data and thereby help us understand how the VSDI signal relates to the underlying neural activity.

### 7.5.3 Imposing latency scatter in TCAL

The first type of latency diversity that can be incorporated in the TCAL model as motivated by the findings of the SIRD model is afferent latency scatter. This mechanism has already been implemented in the SIRD model in a way that can be directly used by the TCAL implementation, but its effect on TCAL has not yet been quantified properly.

This implementation works by sampling a random latency distribution, binning these latencies into multiples of some timestep, and then delaying the input-to-output relation as appropriate. In the SIRD model, this delay is applied directly to the PSTH profiles, which is an approach that can also be transferred to TCAL. A more interesting and plausible approach is to apply the scatter to the activities projected between the various neural sheets, corresponding to scatter at the synaptic level.

Preliminary work on introducing latency scatter to the TCAL units in this way suggests that some modification will be required. The V1 units in the TCAL model integrate over the entire afferent weights field, modeled using dense numeric arrays. As a result, the individual V1 units of TCAL have far more synaptic inputs from the LGN than is realistic. Temporal scatter across all these inputs averages out in the same way it does across cortical space in the SIRD model, resulting in a low temporal scatter of the overall PSTH profile relative to the synaptic input scatter.

To reduce this temporal blurring effect, it should be possible to make the afferent weights sparse, reflecting the average number of synaptic inputs from the LGN that synapses to individual cortical neurons. With calibrated sparsity and appropriate levels of afferent synaptic scatter, it is expected that the model units could have appropriate scatter at the level of their firing rate profiles, as well as appropriate level of synaptic latency scatter. Such a research project could potentially combine with those described

in sections 7.5.4 and 7.5.6 to model the VSDI signal using TCAL.

Alternatively, there could be some grouping by afferent delay, such that individual V1 units would receive only a narrow range of latencies, allowing the population to maintain diversity. How this grouping would be achieved is an open question.

### 7.5.4 Calibrated lateral propagation speeds

The distance-dependent latency shift, also motivated by the SIRD model, could be introduced to TCAL by using calibrated lateral propagation speeds for the lateral connectivity. This can be achieved using sparse, concentric weight rings within the implementation of the lateral projection. Using this approach, the propagation speed is defined by the width of the rings in cortical space divided by the simulation timestep, reflecting how far activity is propagated spatially per timestep. This approach is feasible and has already been tested in an early prototype of TCAL, but only a minor change to the model response was observed. This may be because the impact of spatially dependent diversity on the VSDI signal as predicted by the SIRD model had not yet been established, and so it would be good to revisit this issue now.

Another form of diversity could be introduced by adding stochastic variability to the concentric projection rings used to implement the calibrated lateral propagation speed. The variability added to these concentric weight profiles would correspond to a spread in the lateral propagation speed. This would be an additional type of diversity that is spatially related that is not explicitly included in the current formulation of the SIRD model.

It is hoped that TCAL can eventually be made to incorporate all the features of the SIRD model by combining this suggestion with the projects described in 7.5.2 and 7.5.3. The result would be a developmental map model with calibrated evoked response dynamics that can be directly related to VSDI data. Examining how a diversity of propagation speeds would affect the signal would be an interesting project to try with such a model.

### 7.5.5 Snapshot learning and development

In section 6.2.4, it was found that one of the discontinuous mechanisms from GCAL, namely snapshot learning, had to be re-introduced in order to achieve robust self-organization. It would be trivial to replace this with a continuous equivalent that assigns additional weight to the learning process once the network has settled to form

activity bubbles.

What possible biological justification would such a mechanism have? Activity bubbles are essential for the self-organization process in the developmental models we have considered, and primates do have smooth orientation maps. It is also clear that the initial afferent activity peaks rapidly and that this peak may occur before activity bubbles have had a chance to form, resulting in a high, non-selective initial response that would be expected to have a strong impact on learning. Understanding how these different constraints relate to each other and investigating the possibility of modified learning rules (or some other mechanism) to favor learning when the network activity is settled, would be another interesting research project. The essential component of such a project would be to understand how this conceptual proposal relates to known, biologically plausible mechanisms.

### 7.5.6 Latency diversity and development

There are two closely related questions with respect to the relationship between latency diversity in the TCAL model and the developmental process. Firstly, what effect does imposing such temporal diversity on the model have on the map development process? Secondly, is it possible to explain the emergence of the temporal diversity as a *consequence* development?

Some preliminary work in early TCAL prototypes suggests that latency scatter will have an effect on development. It was noticed that activity bubble formation occurs more slowly after the introduction of latency scatter, with the network taking longer to reach a steady state. This suggests that a shorter timestep may be needed to simulate development once latency diversity has been introduced. This will increase the number of recurrent steps that simulate lateral interactions and might help the activity bubbles form in order to achieve smooth map development.

The second question is an entirely open one. Understanding the developmental origin of the latency diversity would be more satisfying approach than having to directly impose the same latency diversity observed in adult. At this time, it is not known whether such latency diversity is the result of a static random process or whether the latency distribution itself changes over development. If the necessary forms of latency diversity are included in TCAL based on the findings of the SIRD model, this question could generate predictions as to whether the properties of the VSDI signal will change over development.

### 7.5.7 Improved spatial calibration

TCAL is a spatiotemporal model focused on the temporal calibration that deliberately retains the spatial calibration of the GCAL model in order to keep the model definition simple. Spatial calibration in the GCAL model is very approximate, and can be derived by estimating the orientation hypercolumn distance after development and relating it to the known hypercolumn sizes in different species.

There is plenty of scope for a more thorough spatial calibration with respect to a chosen species, namely macaque monkey. This calibration can be achieved in terms of receptive field sizes, magnification factors, and spatial integration fields to construct a Spatially CALibrated GCAL model (SCAL) (Rudiger, 2016). This spatial calibration can then be unified with TCAL, to create a SpatioTemporally CALibrated model (STCAL) where both the spatial and temporal calibration is based on experimental values derived from macaque monkey.

### 7.5.8 Modeling propagating waves in the VSDI signal

The VSDI signal is more dynamic than suggested by the results shown in Figure 4.2 and in particular, propagating waves have been detected using VSDI (Muller et al., 2014; Sato et al., 2012). Investigating these waves in the context of the SIRD model may be possible but it is likely that TCAL would be a better platform for such a project once distance-dependent lateral propagation delays are introduced.

In general, the self-organization of specific lateral connectivity in TCAL, coupled with temporal calibration, including calibrated lateral propagations speeds as detailed in section 7.5.4, offers a way to probe the temporal evolution of cortical dynamics across space. Propagating waves in the VSDI signal is only one example of such a phenomenon.

### 7.5.9 Relating the evoked activity to image statistics

Once a unified framework is established that can integrate developmental and both single-unit and population data, fundamental new research questions can be tackled. Using this modeling platform, it becomes possible to connect the properties of the evoked response, back through the developmental process to the properties of the visual world. This will enable a new, more holistic perspective into the computation involved in cortical vision. The following list illustrates the types of question that

could be addressed within such a framework:

**How do PSTH profiles and the VSDI responses vary over development?**

This is the first, most obvious line of investigation as soon as single unit responses are incorporated into the development process and a VSDI signal model is available. This question is clearly illustrated by the two PSTH grids shown in last figure of the thesis, Figure 6.10. Such a model would be a way to integrate experimental data obtained across different chronic, developmental studies and combine it with studies measuring PSTH profiles at different stages of development.

**How are spatial statistics encoded and how is the evoked response affected?**

The GCAL model is known to encode first-order statistics in terms of the distribution of orientation preferences across the developed orientation map (Stevens et al., 2013b). Whether second-order visual statistics are also encoded in the lateral connectivity is an open question. What TCAL allows is for these types of questions to also be related to the properties of the evoked response. For instance, skewed visual statistics may change the lateral connectivity of the model, and if these lateral connections act causally, this may also affect the temporal properties of the response. Alternatively, it would be possible to investigate how temporal properties vary when the stimulus that is used to evoke a response has either very similar, or very different visual statistics from the training patterns.

**How are spatiotemporal statistics encoded and how is the evoked response affected?**

Introducing temporal properties to both the training stimuli as well as the stimulus raises a whole set of new questions. Instead of using static patterns to represent fixations on a visual scene, short video clips could be sampled to try an approximate naturalistic vision in an environment with motion. Using spatiotemporal training patterns would then affect the temporal activity profiles during development which could then be encoded by the afferent and lateral projections. Would motion and direction selectivity develop? There have been developmental models of direction map formation (Miikkulainen et al., 2005), but these models did not have continuous, consistent models of time and resorted to using multiple lagged populations of neurons instead of true spatiotemporal stimuli. It would also be interesting to investigate how latencies in the visual system, such as those involved in latency scatter, impact visual motion processing.

**What does lateral connectivity encode and how does it modulate the response?**

Specific lateral connectivity is a feature of self organizing developmental map models (Miikkulainen et al., 2005; Stevens et al., 2013b) that is observed anatomically (Buzás et al., 2006). The details of how this connectivity encodes natural image statistics and how it modulates the afferent response remains unknown. In a developmental model with specific lateral connectivity and calibrated, evoked responses, it becomes possible to investigate how this lateral connectivity modulates activity and to consider what types of computation might be performed.

**How do mechanistic and normative models relate to each other?**

As the relationship between the structure of TCAL after development and image statistics used to train it are explored in more detail, it is natural to try to understand what the visual cortex is computing in more general terms. For instance, maybe simple cells are attempting to encode images to minimize reconstruction error under a sparsity constraint (Olshausen and Field, 1996). In a spatiotemporal, mechanistic, developmental model it becomes possible to explore such questions in more detail. Normative models are often idealized, by optimizing some objective function concerning what a biological system *should* do. An appropriate mechanistic model could try to evaluate such claims and establish how closely the biological mechanism can satisfy the various constraints predicted by normative criteria.

### 7.5.10 Realistic anatomical connectivity

The TCAL model simulates the primary visual cortex using a single sheet of units in the same way as GCAL, with incoming afferent connectivity from the ON/OFF sheets and lateral excitatory and inhibitory connectivity within the cortical sheet. This simple approach is sufficient for directly modeling the Mexican-hat interactions required for map development described in section 2.5.2, but it also ignores several key points regarding cortical anatomy.

Anatomically, a single population of neurons cannot have make both excitatory and inhibitory connections, and the neuroanatomy suggests short-range *inhibition* and longer-range *excitation* where inhibition is mediated by smaller interneurons and where excitation is mediated by long-range lateral connectivity. There has been work to resolve this apparent discrepancy in the context of developmental models based on GCAL (Law, 2009; Rudiger, 2016) and some of the relevant issues are also discussed

in section 2.5.2.

Modeling separate populations of cells with more plausible anatomical connectivity would allow TCAL to analyze the overall response of the population in terms of the different temporal properties of different cell types. Such a breakdown could allow different contributions to the latency diversity to be attributed to different populations of cells in more detail. For instance, there may be different temporal properties in the way excitatory activity propagate across the cortex relative to inhibitory activity.

In addition, realistic connectivity profiles would allow the process of development to be related to other spatiotemporal dynamics of other phenomena such pop-out, orientation contrast suppression as well as contour completion or iso-orientation facilitation.

### 7.5.11 Laminar organization

Another way to improve the anatomical detail of the model is to introduce a laminar organization. This has also been previously addressed within the framework of self-organizing map models in order to understand the emergence of both simple and complex cells (Antolik and Bednar, 2011). Introducing a laminar organization could be relevant for improving the VSD signal model as (1) the distinction between layer 4 and layer 2/3 may be relevant to understanding the distribution of spatially independent and spatially dependent latencies in the SIRD model and (2) complex cells are found in the more superficial layers and are therefore likely to be more heavily represented in the VSDI signal.

This suggests that one clear way to improve the SIRD model is to understand the VSD signal better as a function of cortical depth. For instance, it would be interesting to use the detailed biophysical model of the VSD signal (Chemla and Chavane, 2010a) to understand whether the simple parameters of the SIRD model are justified in terms of more detailed biophysical contributions. This could help weight the two types of responses in the SIRD model which would then offer a suitable way to calibrate a laminar version of TCAL.

Combining TCAL with these more anatomically detailed, self-organizing map models show that there is plenty of scope for more biophysical detail, such as laminar organization, without having to abandon firing-rate units.

### 7.5.12 Improvements to the map metric

Tighter model calibration and an improved suite of metrics is always desirable. In particular, the map quality metric presented in this thesis is based only the orientation preference map and does not take into account orientation selectivity. Unfortunately, it is not yet known whether a metric can be constructed based on orientation selectivity that offers an absolute reference for comparison that is applicable across species.

One possibility may be to examine the ratio of selectivity at pinwheel positions in relation to the mean selectivity across a map. This measure would tend to fall in the unit range: good maps where pinwheels have a far lower selectivity than the surrounding area would have a ratio close to zero. Poor maps that have an average selectivity at the pinwheel locations that is no different from the rest of the map would have a ratio close to unity. Subtracting this ratio from unity may then offer a dimensionless metric where a value of zero corresponds to unrealistic selectivity maps and values close to unity corresponds to more realistic selectivity maps.

This suggestion would depend on the average orientation selectivity in the vicinity of a pinwheel and would not depend on the selectivity at the level of individual cells near the pinwheel center. Using this selectivity based metric together with the pinwheel density metric in some way may lead to a more comprehensive map quality metric. Improving the map metric by using selectivity information is one promising direction for future work that would require rigorous evaluation against experimentally recorded orientation selectivity maps.

Another possibility is to quantify the repulsion and attraction effects between pinwheels of similar and opposite polarities respectively (Müller et al., 2000). Although this property has been observed in the GCAL model, it is not apparent how the strength of this effect may be assessed and related to experimental maps, especially in a cross-species manner.

### 7.5.13 Automated map metric optimization

Developmental models can be difficult to work with and having to additionally calibrate the evoked dynamics properties would make them even more difficult to tune. Only the initial conditions and training pattern statistics are specified at the start of the simulation, and it is difficult to predict how these settings will affect the developmental process. This means it is necessary to wait for a simulation run to complete before the effect of the parameter change can be observed.

The combination of Lancet, Topographica, and the map quality metric may allow the developmental parameters of such a model to be tuned automatically. Without a map metric, the task of assessing orientation map quality requires human intervention and judgment. With an automated metric to assess whether or not the map development process is realistic, it should now be possible to use automated parameter optimization techniques.

Lancet was designed to enable this type of optimization as it supports guided parameter exploration. It is possible for Lancet to launch a collection of simulation runs in parallel, collect some statistics from those runs once they complete, and then use these results to determine appropriate parameters for a new set of parallel simulations. This would make it possible to use Lancet together with optimization techniques, such as gradient descent or hill climbing to automatically find appropriate parameter values.

This approach would be a powerful research tool that could greatly enhance research productivity and the speed with which researchers can tune developmental map models. As Topographica simulations are launched with Lancet and have their output serialized as HoloViews objects, it is easy to collect all the data as it is being generated on a cluster and interactively visualize it in a Jupyter notebook while the jobs are still running. This would allow the automated exploration process to be supervised, allowing exploration to be terminated early on if necessary or perhaps allowing additional human guidance to guide the automated procedure.

### 7.5.14 Speeding up simulations

The faster a simulation run can be completed, the easier it becomes to explore and tune the model. If the core computations can be accelerated, it is possible to either run more simulations in a set time or the same number of simulations may be executed in more detail. Some of the ways of improving simulation detail in TCAL include using a smaller timestep, simulating a larger cortical area or using a higher spatial resolution. Even though TCAL only has a single cortical population of firing-rate units, simulations still take several hours to complete.

In recent years, graphics processing units (GPUs) have become a core component of high performance, scientific computing. When used properly, GPUs can achieve a speed up of several orders of magnitude. As computation using GPUs is so much more efficient for highly parallel processes, it is worth investigating whether they can be used to speed up simulations in models such as GCAL or TCAL.

The most computationally expensive steps are the activity integration step, given by Equation 2.4 and the learning step, specified by Equation 2.10. At first glance, these equations look ideal for GPU execution as these computations appears highly parallelizable. Unfortunately, early attempts to optimize the Topographica implementation with GPU support proved unsuccessful.

Recently, a far more promising approach has been found, which is to use sparse matrices on the GPU, allowing Equations 2.4 and 2.10 to be fully vectorized within the available GPU memory. As sparse projections have already been mentioned in sections 7.5.4 and 7.5.3, it seems that future work with TCAL would greatly benefit from a sparse GPU implementation.

## 7.6 Overview

The work presented in this thesis addresses a number of specific, scientific questions, including: How is the population response related to the single-unit response?, Is it possible to formulate an automated orientation map quality metric? Does the GCAL model have $\pi$ pinwheel density? Can a developmental model be made to simulate the evoked response with plausible PSTH profiles? What additional mechanisms are necessary to achieve plausible PSTHs? Answering these specific questions was one of the goals of this thesis, but the real motivation has been to enable new research opportunities, and not to answer any one particular question.

There is no shortage of specific, targeted models in computational neuroscience, but there is a real lack of unifying theory, a point that has been made by Stevens (2000). Without an overarching theory, it can difficult to make progress as different results are established without being integrated together into a bigger picture. This is point is well illustrated in a number of papers that discuss whether the techniques of neuroscience would assist in the investigation of well understood computational artefact, namely a computer processor (Brown, 2014; Jonas and Kording, 2016). The results so far are not encouraging.

The work presented has been geared towards making a more unified model and a more unified theory of cortical function possible. The work on building new, improved research tools in Chapter 3 is aimed to make it easier to deal with larger, integrative models in a reproducible way. In Chapter 4, the goal was to all the integration of experimental data across spatial scales, helping to build a unified picture of the evoked response at the single-unit an population levels.

Next, the emphasis switched to developmental models. Chapter 5 was focused on the analysis of one particular development model, GCAL, but the map metric is a general tool. With a map metric, a researcher can decided that an orientation map model is "good enough" with respect to map development, allowing them to expand the model into new areas. Lastly, the TCAL model in Chapter 5 shows how GCAL can be extended into an entirely new temporal domain. This shows that developmental models can be designed in a way that makes contact with the large experimental literature regarding the evoked response. In particular, TCAL makes contact with the SIRD model, hinting at new directions for future research.

Taken together, this is intended to be a platform for new research that acknowledges the importance of development when it comes to understanding vision. The breadth of the future work section above indicates only some of the ways developmental models may be extended in order to build a unified, consistent theory of cortical function.

# Chapter 8

# Conclusion

Large-scale cortical structures are composed of millions of neurons, each making thousands of connections. The neural tissue is composed of billions of synapses integrating signals across countless different types of cells with different morphological properties, different connectivity profiles, and using different neurotransmitters. Understanding the components of the nervous system, how they interact, and how they respond is at the core of neuroscience. And yet all this complexity emerges from a developmental process, starting with a single cell.

In this thesis, a link is established between the literature concerning the evoked response to a visual stimulus at the level of individual neurons to the literature concerned with understanding how entire population of neurons slowly develop over time. The developmental process is key to understanding how neural responses become diverse, where their connectivity comes from, and how a neural response relates to a stimulus according to the long-term environmental statistics. Trying to understanding how the cortex got to be the way it is can simultaneously simplify and deepen our understanding of the visual process.

Chapter 3 lays the methodological foundation for the work presented in this thesis, by recognizing that in order to make progress, research needs to be both productive and reproducible. Two new tools, Lancet and HoloViews, are presented in this chapter, showing how they can help achieve this goal in a literate programming environment. These tools not only enabled the simulations and analyses presented of subsequent chapters but are designed to make it easier for future researchers to build on this work.

Chapter 4 showed that various different forms of onset latency diversity across the cells in the primary visual cortex are necessary for explaining the bulk response of a population, starting with a descriptive model of single-cell responses. This model

has a very simple mathematical formulation but incorporates many different sources of experimental data in its calibration. The result is the first large-scale model that can relate the spatiotemporal dynamics of the population response, as revealed by voltage-sensitive–dye imaging, to the response of single cells.

Chapter 5 developed an approach for quantitatively analyzing developmental models of orientation map formation, leading to the validation and subsequent publication of the GCAL model. This analysis demonstrated that this particular developmental model is simpler than its predecessors and yet more robust and stable. From easily specified, random initial conditions, this model develops a diverse population of cells with a variety of afferent receptive fields, different tuning properties and specific lateral connectivity.

Chapter 6 showed that GCAL could be simplified further by removing the mechanisms that made it specific to developmental modeling, resulting in a new model that operates uniformly across both short and long timescales. Every simulation step in TCAL operates according to the same rules as every other simulation step, ensuring that development emerges from the accumulation of tiny changes that occur on the timescale of the evoked response. Furthermore, TCAL achieves plausible PSTH profiles in the cortical layer by revealing the dynamic response of a mechanism already introduced to the GCAL model for the purposes of improving the robustness of map development. The result is a developmental model that can be connected directly to the initial assumptions of the SIRD model, pointing the way forward for future work.

In conclusion, the work in this thesis shows that the problem of understanding vision can be tackled within a unified modeling framework spanning the temporal and spatial scales relevant to the visual process. The result is a new type of model that makes no distinction between the processes that act on short and long timescales, which will allow new scientific questions to be posed within a single, unified computational framework. The purpose of this model is to open up a novel set of possibilities for theoretical investigation and suggest new avenues for experimental research.

# Appendix A

# Publications

All the code and material needed to run the simulations in this thesis, as well as the materials used to build this thesis itself, may be found at jlstevens.github.io/thesis. In addition, this appendix contains three papers containing material relevant to this thesis that were published over the course of the research project:

*Mechanisms for Stable, Robust, and Adaptive Development of Orientation Maps in the Primary Visual Cortex* by **Jean-Luc R. Stevens, Judith S. Law, Ján Antolík, and James A. Bednar.** Journal of Neuroscience, 33:1574715766.

*An automated and reproducible workflow for running and analyzing neural simulations using Lancet and IPython Notebook* by **Jean-Luc R. Stevens, Marco Elver, and James A. Bednar.** Frontiers in Neuroinformatics, 7:44.

*HoloViews: Building Complex Visualizations Easily for Reproducible Science* by **Jean-Luc R. Stevens, Philipp Rudiger, and James A. Bednar.** In *Proceedings of the 14th Python in Science Conference*.

Systems/Circuits

# Mechanisms for Stable, Robust, and Adaptive Development of Orientation Maps in the Primary Visual Cortex

**Jean-Luc R. Stevens,[1] Judith S. Law,[1] Ján Antolík,[1,2] and James A. Bednar[1]**

[1]Institute for Adaptive and Neural Computation, University of Edinburgh, Edinburgh EH8 9AB, United Kingdom and [2]Unité de Neuroscience, Information et Complexité, Centre National de la Recherche Scientifique, 91198 Gif sur Yvette, France

**Development of orientation maps in ferret and cat primary visual cortex (V1) has been shown to be stable, in that the earliest measurable maps are similar in form to the eventual adult map, robust, in that similar maps develop in both dark rearing and in a variety of normal visual environments, and yet adaptive, in that the final map pattern reflects the statistics of the specific visual environment. How can these three properties be reconciled? Using mechanistic models of the development of neural connectivity in V1, we show for the first time that realistic stable, robust, and adaptive map development can be achieved by including two low-level mechanisms originally motivated from single-neuron results. Specifically, contrast-gain control in the retinal ganglion cells and the lateral geniculate nucleus reduces variation in the presynaptic drive due to differences in input patterns, while homeostatic plasticity of V1 neuron excitability reduces the postsynaptic variability in firing rates. Together these two mechanisms, thought to be applicable across sensory systems in general, lead to biological maps that develop stably and robustly, yet adapt to the visual environment. The modeling results suggest that topographic map stability is a natural outcome of low-level processes of adaptation and normalization. The resulting model is more realistic, simpler, and far more robust, and is thus a good starting point for future studies of cortical map development.**

## Introduction

Orientation-selective neurons in carnivoran and primate primary visual cortex (V1) form systematic topographic maps organized by preferred retinal location and orientation (Blasdel, 1992a,b; Kaschube et al., 2010). Optical imaging studies in ferrets indicate that these orientation maps develop in a stable way, with a high similarity in the spatial layout between the weakly selective initial maps and the final highly selective configuration (Chapman et al., 1996; Chapman and Bonhoeffer, 1998; Gödecke et al., 1997). This stability may be important for allowing regions downstream from V1 to develop, and is particularly remarkable given that V1 neurons undergo massive morphological changes over this time (for review, see White and Fitzpatrick, 2007; Huberman et al., 2008). V1 map development is also quite robust against differences in inputs, with similar final map patterns developing until nearly 3 weeks of age in cats, regardless of whether the eyes are open (Crair et al., 1998). Yet map development is not simply decoupled from vision, because the development of selectivity corresponds to a critical period during which visual experience is required for neurons to achieve fully mature levels of orientation tuning (Crair et al., 1998; Chapman and Gödecke, 2000; White et al., 2001; White and Fitzpatrick, 2007). Moreover, neurons and maps in animals reared in abnormal visual conditions during the critical period come to reflect the statistics of the abnormal visual input [e.g., lid suture, (Wiesel and Hubel, 1963; Blakemore and Van Sluyters, 1975; White et al., 2001), striped environments (Blakemore and Cooper, 1970; Sengpiel et al., 1999), or oriented blurring via goggle rearing (Tanaka et al., 2006, 2009)].

Together, these results indicate that orientation map and selectivity development is remarkably stable and robust against variability in inputs over time or across animals, yet adaptive to the statistics of available visual input. Although numerous computational models of activity-driven map development have been proposed (for review, see Swindale, 1996; Goodhill, 2007), no map model has yet been shown to develop stably in response to patterned inputs of different types and strengths. Understanding how maps can develop stably is both of direct scientific interest, and may provide a criterion that allows inherently unstable models to be rejected.

In this paper, we begin with a simple mechanistic model including lateral connectivity that captures the essential features of previous incremental Hebbian self-organizing network models for map development (von der Malsburg, 1973; Kohonen, 1982; Obermayer et al., 1990; Sirosh and Miikkulainen, 1994; Barrow et al., 1996; Burger and Lang, 1999, 2001; Bednar and Miikkulainen, 2004; Miikkulainen et al., 2005), while omitting mechanisms that destroy stability. Although this model develops maps, we show that it is still relatively unstable, and is not robust to changing inputs, e.g., varying input contrast. We then demonstrate that

Received March 7, 2013; revised Aug. 1, 2013; accepted Aug. 13, 2013.

Author contributions: J.-L.R.S., J.S.L., J.A., and J.A.B. designed research; J.-L.R.S., J.S.L., and J.A. performed research; J.-L.R.S., J.S.L., and J.A.B. contributed unpublished reagents/analytic tools; J.-L.R.S., J.S.L., and J.A. analyzed data; J.-L.R.S., J.S.L., and J.A.B. wrote the paper.

This work was supported in part by Grants EP/F500385/1 and BB/F529254/1 to the University of Edinburgh Doctoral Training Centre in Neuroinformatics and Computational Neuroscience (www.anc.ed.ac.uk/dtc) from the UK Engineering and Physical Sciences Research Council, Biotechnology and Biological Sciences Research Council, and Medical Research Council. The work made use of resources provided by the Edinburgh Compute and Data Facility (www.ecdf.ed.ac.uk). Thanks to Philipp Rudiger, Chris Ball, Stuart Wilson, and other members of the Institute for Adaptive and Neural Computation for helpful comments and discussion. This work is dedicated to the memory of Barbara Chapman, 1963–2013.

Correspondence should be addressed to James A. Bednar, School of Informatics,The University of Edinburgh Edinburgh EH8 9AB, UK. E-mail: jbednar@inf.ed.ac.uk.

DOI:10.1523/JNEUROSCI.1037-13.2013

 0270-6474/13/3315747-20$15.00/0

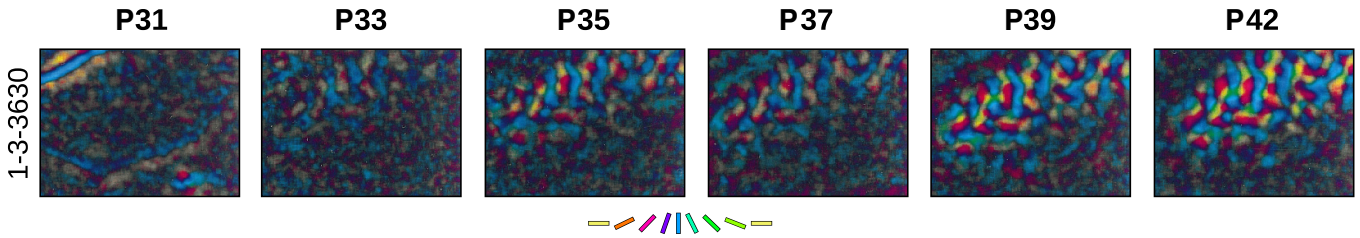


**Figure 1.** Stable development of orientation maps in ferret striate cortex. Orientation maps recorded in the primary visual cortex of one ferret (animal 1-3-3630) at the postnatal ages indicated are shown. In these polar maps, pixel color indicates orientation preference, and pixel brightness indicates the strength of orientation tuning. The selectivity of each map is normalized independently, making blood vessels visible at P31 (e.g., the blue streak in the top left corner), but not at P42, once orientation selectivity has developed. As the maps mature, iso-orientation domains become visible as colored regions that become more strongly responsive over time, without changing the overall map pattern. Data are reproduced from the study by Chapman et al. (1996).

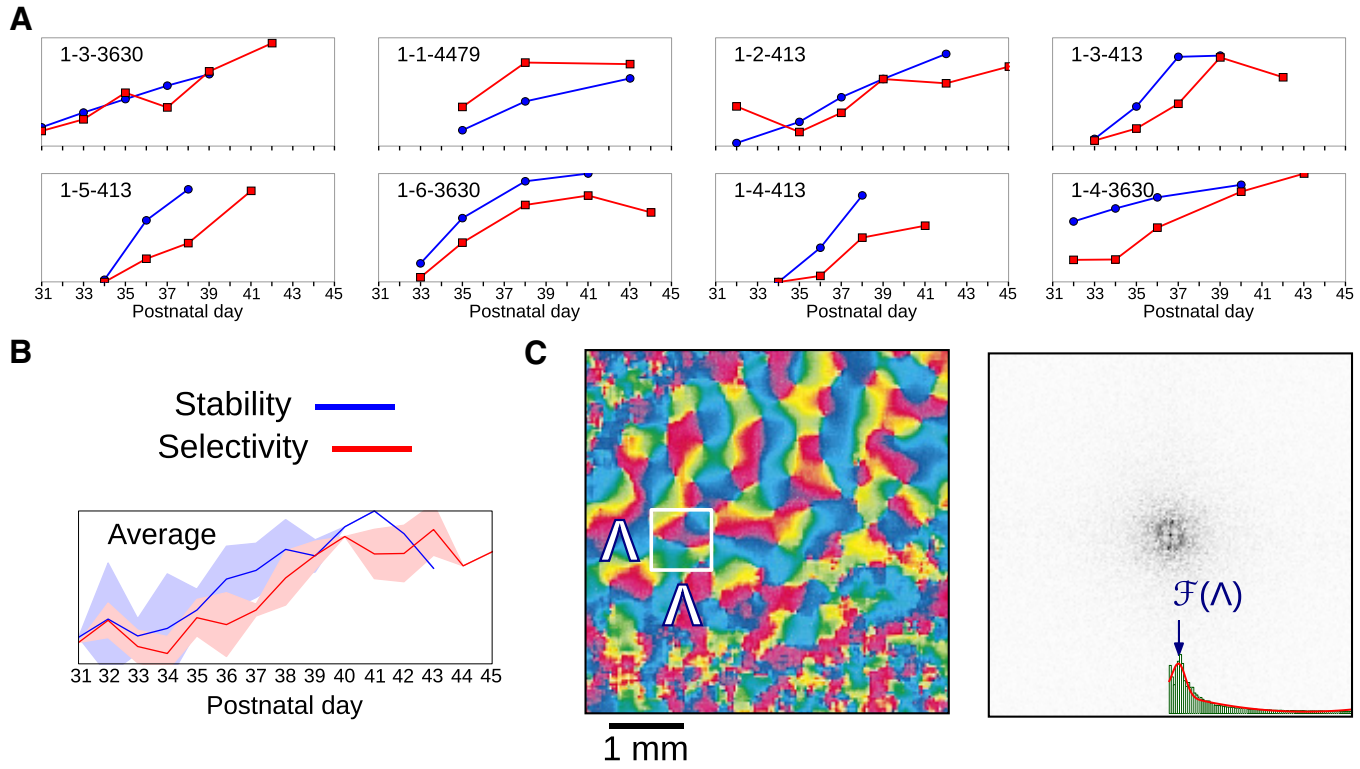


**Figure 2.** Development of orientation maps measured by chronic optical imaging in ferrets. ***A***, Recorded selectivity (red square markers) and stability values (blue round markers) for all eight ferrets, as a function of postnatal day [replotted data from the study by Chapman et al. (1996), their Figs. 4, 7]. Stability is quantified by the average difference in preference of each orientation map with the final map, as defined by Equation 1. A common selectivity and stability scale is used to allow comparison between ferrets. All values are in arbitrary normalized units, using the lowest recorded value as the zero reference point and the highest recorded value as the maximum (see Materials and Methods). ***B***, The mean selectivity and stability across ferrets as a function of postnatal day, with the $\pm$95% confidence intervals for each day indicated by the shaded area around the mean line. Selectivity and stability increase steadily and simultaneously over development so that once neurons are selective, they are organized into a map with the same form as the final measured map. These results show that selectivity does not precede increasing stability, indicating stable map development. ***C***, As a reference for later modeling work, the final orientation preference map (without selectivity) for ferret 1-3-360 on day P42 (Fig. 1, rightmost plot) is shown. The map is organized into regularly repeating hypercolumns in all directions, as seen by the ringness in the Fourier power spectrum, plotted with the highest amplitude component in black. The center value is the DC component, and the midpoint of each edge represents half of the highest possible spatial frequency in the cardinal directions (i.e., the Nyquist frequency). This Fourier spectrum is used to calculate the average periodicity of the map, which is then plotted as a hypercolumn area $\Lambda^2$, covering one period in the cardinal directions (white boxed area). Figure 3 illustrates how to use these calculations to determine whether a model map resembles this animal data.

extending the model with two well-established low-level properties, neuron threshold adaptation and feedforward gain control, independently improves robustness and stability, and that when used together, the final GCAL (gain control, adaptation, laterally connected) model achieves biological maps with robust and stable development, while reflecting the statistics of visual input. The results suggest that the well-established mechanisms of gain control and single-neuron homeostatic adaptation are important basic principles underlying map development, not just for individual adult or developing neurons.

## Materials and Methods

The following sections first present methods used for quantifying the emergence of selectivity in biological maps, the stability of these maps over time, and the degree to which model maps resemble biological maps. We then describe the set of models used in this paper.

*Measuring stability and selectivity.* The stability of orientation map development is illustrated in Figure 1 with data from chronic optical imaging in ferrets, reprinted from Chapman et al. (1996). Stability and selectivity measures for maps in eight ferrets (of either sex; T. Bonhoeffer, personal communication) are shown in Figure 2 and described below.

Qualitatively, map development is considered stable if the map pattern remains constant once neurons have become highly selective; unstable development would be characterized by maps that reorganize substantially after orientation-selective patches first emerge (Chapman et al., 1996). In electrophysiological measurements from ferret V1, average selectivity (orientation tuning) is low from about postnatal day 23 (P23; when visual responses can first be measured) until P28–P34, reaching adult levels by P42–P49 (Chapman and Stryker, 1993). To examine stability of maps using optical imaging, Chapman et al. (1996) thus focused on the period from P31 to P45. This period also corresponds to the critical period for visual experience—disrupting activity before P50 eliminates selectivity even with normal visual experience after this age (Chapman and Gödecke, 2000). For this modeling study, we will also focus on this period of normal orientation selectivity emergence; stability after the critical period can be modeled trivially by disabling or greatly reducing plasticity, and will not be investigated further here.

The red lines with square markers in Figure 2*A* show the average selectivity of the optical imaging signal from P31–P45 for all eight ferrets (Chapman et al., 1996); one additional data point from ferret 1-1-4479 at P55 is not shown. Although the selectivity values are on an arbitrary scale due to the measurement technique, the values for each postnatal age correspond closely to those found electrophysiologically (Chapman et al., 1996), and thus cover the range from highly unselective to adult-like selectivity.

To assess stability quantitatively over this same time period, Chapman et al. (1996) computed the correlation of orientation preference at each developmental age with the organized preference map observed in the final recording for that animal. For convenience, we have linearly rescaled the "orientation similarity index" values from that study to vary from 0.0 (uncorrelated preference) to 1.0 (identical preference), which we compute as a stability index (SI):

$$\mathrm{SI} = 1 - \frac{4}{n\pi}\sum_i \left| (F_i - O_i) \bmod \left(\frac{\pi}{2}\right) \right|, \qquad (1)$$

where $i$ iterates over all $n$ data points (pixels in the imaging frame), $F_i$ is the final orientation preference value for the neural unit corresponding to pixel $i$, and $O_i$ is the corresponding preference value at an earlier age. The blue lines with round markers in Figure 2*A* show the results of this calculation for eight ferrets from Chapman et al. (1996) measured at different postnatal days. Note that even identical underlying maps would give values below 1.0 on this measure, if there is any noise or variation in measuring the map, and so the values shown have been normalized by the highest and lowest stability values across all ferrets studied.

Given this data, stable development is defined as a map having high SI values whenever the average selectivity value is high. When the selectivity is very low, presumably before the neurons have developed at all, the map is arbitrary and dominated by measurement noise, but if development is stable, the SI value should increase as soon as the selectivity value increases. Unstable development would be visible as a selectivity value that increases well before stability does, representing neurons that have achieved selectivity before assuming the final preference value observed in the last map measured. As can be seen in Figure 2*A*, the SI value and the average selectivity are highly correlated in every ferret shown, with both values typically increasing over time rather than selectivity predating stability (though ferret 1-1-4479 could be seen as relatively unstable). An overall correlation between selectivity and stability is a hallmark of stable map development, and is visible as a general trend in the average data from all ferrets, plotted in Figure 2*B*.

*Assessing the organization of orientation maps.* Recording the selectivity and stability index is sufficient for evaluating development in experimental recordings, where all maps are necessarily biologically realistic. In simulated models of map development, there is no guarantee that simulated maps have an orientation structure matching those observed in real animals. An unrealistic map may be both stable and selective, and so we require that a good model of map formation also results in realistic maps, similar to those found in animals. Ideally, map quality would be assessed with an automated metric that reports how close the maps are to animal data, e.g., on a scale from 0 to 1.0 as for stability.

There have been several previously published measures that could be used to quantify the degree of map organization. Biological maps tend to be smooth, which can be measured by the local homogeneity index (LHI; Nauhaus et al., 2008) or the earlier "local input orientation selectivity index" (Schummers et al., 2004), which both convey the local homogeneity in orientation preference at a particular cortical location. Biological maps typically have non-uniform orientation histograms (Coppola et al., 1998; Müller et al., 2000; Tanaka et al., 2009), which have been used to characterize model maps as well (Bednar and Miikkulainen, 2004). Biological maps tend to have a ring-shaped Fourier power spectrum (Blasdel, 1992a,b; Erwin et al., 1995; Fig. 2*C*), due to the regular repetition of orientation patches across the cortical surface. Orientation maps also contain pinwheels that may be identified, classified by polarity (whether orientation preference increases clockwise or counterclockwise) and analyzed; e.g., the average distance between nearest-neighbor pinwheels tends to be greater between those of matching polarity than those of opposite polarity (Müller et al., 2000).

Although these measures can each help quantify the realism of simulated maps, they do not offer a simple, unique reference value that can characterize all biological maps. A useful metric should assign biological maps to a narrow possible range of values, ideally with theoretical justification for what values should be expected in the biological system and why. Furthermore, a reliable metric should be consistently defined between species, if it is to avoid uncertainty in fitting species-dependent free parameters.

Accordingly, we developed a new map-quality metric based on the empirical observation that pinwheel count in biological orientation maps scales linearly with hypercolumn size across many different species (Kaschube et al., 2010). This linear relation gives a consistent number of pinwheels per hypercolumn area ($\Lambda^2$), implying a constant pinwheel density when averaged over sufficiently large cortical areas. This dimensionless, statistical measure of pinwheel distribution is thought to reflect a universal constant of map organization, converging to $\pi$ across carnivorans, primates, cats, and tree shrews (Kaschube et al., 2010; Keil et al., 2012). This value was predicted by a theoretical model of map organization and has strong empirical evidence, with a mean pinwheel density across four species (tree shrew, galago, cat, and ferret) statistically indistinguishable from $\pi$ (Kaschube et al., 2010); see data from three species in Figure 3*D*.

To establish the pinwheel density for any given map, the total pinwheel count must be determined. Pinwheel centers are located at the intersection of the zero contours of the real and imaginary components in the polar representation of orientation preference (Löwel et al., 1998). These contours are shown in Figure 3*A* as black and white lines for a sample simulated preference and selectivity map with approximately $\pi$ pinwheel density. The preference-only channel of this high-quality map is shown together with corresponding pinwheel density and pinwheel count in Figure 3*Ca*. All the maps shown in Figure 3 are for illustration only and are derived from simulations to be described later; simulated data are necessary to demonstrate the behavior of the metric for nonbiological maps.

For simulated maps using uniform random input statistics, a single number is sufficient to characterize the Fourier plots: the radius of the isotropic ring, computed using the methods described in the study by Kaschube et al. (2010). To eliminate disruptions in hypercolumn size due to border effects at the sheet edges, only the central $1.0 \times 1.0$ area from a simulated V1 sheet of area $1.5 \times 1.5$ is analyzed for all model maps presented. The first step is to integrate the spectral power as a function of radius, as shown by the histograms in the FFT plots. The peak values of these histograms are estimated using a least-squares fit of a Gaussian curve with additional linear and quadratic terms (Kaschube et al., 2010, their Equation 7, supplementary materials). The computed fits are shown by the red curves overlaying the histograms with the fitted peaks indicated by the blue arrows, marking the estimated ring radius. This value reflects the periodicity with which hypercolumns repeat across the map and is equal to the hypercolumn area in units of $\Lambda^2$ when squared, shown for a ferret map in Figure 2*C* and for simulated maps in Figure 3, *A* and *B*.

An identical analysis is shown for a lower-quality simulated orientation map in Figure 3*B*, where the map was generated by running a simulation outside its optimal operating range. Disruptions to the smooth organization of preference result in clumps of pinwheels, explaining the higher pinwheel density seen in Figure 3*Cb*. These clumps of pinwheels could in principle be eliminated arbitrarily as duplicates, but because they represent genuine areas of poorly organized preferences in these

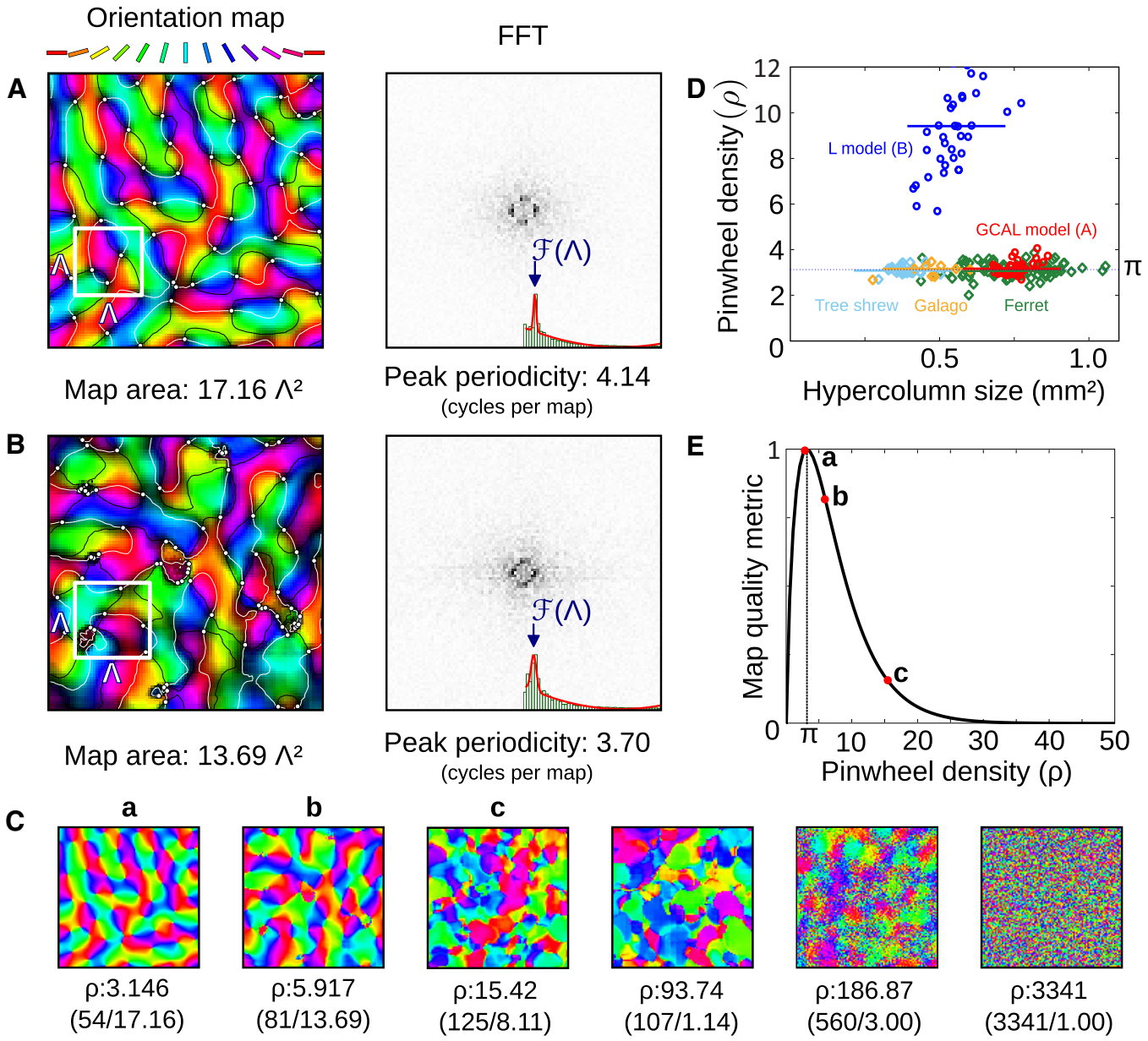


**Figure 3.** Evaluating map quality as the deviation from $\pi$ pinwheel density ($\rho$). ***A***, High-quality, realistic, orientation preference and selectivity map with approximately $\pi$ pinwheel density (3.146), from the final model to be discussed in this paper (GCAL). Corresponding preference-only map is shown in ***Ca***. Pinwheel density is defined as the average number of pinwheels (white circles) per hypercolumn area ($\Lambda^2$) indicated by the white boxed area (***A, B***). Pinwheels are identified at the intersection of the zero contours of the real and imaginary components in polar representation (white and black contours respectively). The periodicity of hypercolumns is estimated from the radius of the ring in the Fourier power spectrum (FFT) using the fitting method described in Kaschube et al. (2010). ***B***, A lower-quality map generated from the first model introduced in this paper (L), which has visible discontinuities in OR preference seen in ***Cb***. The greater pinwheel density (5.917) is due to a higher pinwheel count and a larger hypercolumn area due to more widely spaced orientation blobs. The histogram (FFT plot inset) indicates mean spectral power as a function of radius, the red line indicates the least-squares fit (Kaschube et al. 2010, their supporting online material, Eq. 7), and the blue arrow indicates the estimated peak spectral power radius (***A***, ***B***). ***C***, A selection of model maps ordered by pinwheel density with pinwheel count to hypercolumn area ratio, shown in parentheses. Lower-quality maps usually have higher estimated pinwheel counts and correspondingly higher pinwheel densities, with pinwheel counts so large for very poor maps as to be effectively undefined. ***D***, Pinwheel density of three species (diamonds) and simulated maps (circles) as a function of hypercolumn size [data replotted from the study by Kaschube et al. (2010)]. Horizontal lines indicate median values of each cluster with the medians of high quality model maps also clustered around $\pi$. ***E***, Normalized, heavily tailed gamma distribution used to transform pinwheel density into a suitable metric on a unit interval with values for maps a, b, and c.

noise-free simulations, they have been retained as indicators of poor map structure. This trend is illustrated in Figure 3*C* by a selection of maps of increasingly poor quality that illustrate how the pinwheel count and density tend to increase as map quality decreases.

The validity of using $\pi$ pinwheel density as a reference value is demonstrated in Figure 3*D*. The experimental values of pinwheel density for tree shrews, galagos, and ferrets are plotted as diamonds, using data reproduced from the study by Kaschube et al. (2010). The values for simulated maps are plotted as circles for the high- and low-quality map simulations ($n = 40$ and $n = 33$ respectively). The high-quality maps are obtained from the final model developed in this paper, while the lower-quality maps are computed using the initial, simplified model introduced in the next section. The horizontal lines through each cluster indicate the median value of the cluster points. The median is indistinguishable from the mean for all clusters except the low-quality map cluster, which has outliers of relatively high pinwheel density not visible in the plot.

It would be interesting to evaluate pinwheel density over development using the chronically recorded ferret data, illustrated in Figure 1. This measurement is not possible without access to the raw data, as prepro-

cessing may have introduced biases to the orientation structure that need to be carefully controlled for when assessing pinwheel density (Kaschube et al., 2010). The pinwheel density at the end of maturation in ferrets has been established using 82 maps recorded from adult animals, shown by a cluster of green diamonds centered around $\pi$ in Figure 3*D*.

For the poorest-quality maps, the pinwheel density is unbounded, as illustrated by Figure 3*C*. In these highly disorganized maps (to the right of Fig. 3*C*), an unrealistically large number of locations match the automated criterion for a pinwheel; few or any of these would satisfy any subjective definition of a pinwheel. Even so, the "pinwheel count" is still a useful measure in these cases, because such high numbers reliably indicate low-quality maps.

To turn pinwheel density into a useful metric between unity (high-quality maps) and zero (low-quality maps), a heavily tailed squashing function is needed, mapping all the maps with unrealistically high pinwheel counts to zero. We chose a normalized gamma distribution, shown in Figure 3*E*. The labels a, b, and c show how the final metric value is computed for the three example maps shown. The gamma distribution $\Gamma(k, \theta)$ is characterized by $k = 1.8$ and $\theta = \pi/(k - 1)$. This latter constraint on $\theta$ ensures that the mode of the curve occurs at $\pi$. This kernel is then normalized by the value evaluated at $\pi$ to ensure that maps with exactly $\pi$ pinwheel density have a metric value of 1. The only free parameter $k$ is set to the value $k = 1.8$ to allow detailed discrimination of high-quality maps without discarding all the information available for poor maps with a pinwheel density up to 30; other values will place more or less penalty for very poor maps with unrealistically high pinwheel densities, but will otherwise lead to similar results.

Of course, a value of unity on the pinwheel density metric (or any metric) is not sufficient by itself to guarantee that a given simulated map is indistinguishable from a biological recording. For instance, it should be possible to construct a synthetic map designed specifically to attain $\pi$ pinwheel density that nonetheless appears highly unnatural. Such deliberate "gaming" of the map metric is not possible using the simulations presented in this paper, as the maps emerge gradually from a developmental process, without modeler control over the specific placement of pinwheels in the final organization. Even so, it is important to verify that simulated maps with a high metric value satisfy the types of subjective criteria used to evaluate biological results, such as those introduced by Blasdel (1992b). For each result where maps are evaluated by metric, we have thus also performed subjective evaluations of map quality, such as assessment of whether the FFT is ring shaped, and whether there are saddle points, linear zones, 180° pinwheels, fractures, and so on. We have found no examples of simulated maps that achieve a realistic pinwheel density but do not also look realistic with respect to these other properties. The automated pinwheel density metric value thus correlates well with our overall subjective assessment of map quality. We have also ensured that all the plots presented in this paper are representative of those simulations that are not shown, for the same conditions.

*Model architecture.* The mechanistic models we will consider are variants of the original SOM (self-organizing map) algorithm introduced by von der Malsburg (1973). In this type of model, the inputs are actual visual images or patterns of spontaneous activity, which can be directly related to the visual environment or to the imaging of retinal activity patterns. Biologically plausible properties of single neurons can be integrated into the network to explain many of the observed phenomena, as opposed to just the geometric properties of the map pattern.

Unlike correlation-based learning models that use Hebbian learning over large batches of inputs (Linsker, 1986; Miller, 1994), self-organizing map models operate using incremental Hebbian learning rules. Although the mathematical structure of the final model is often less amenable to analysis than those of linear, feedforward networks, the incremental nature of self-organizing maps make them more suitable for studying map development. Incremental learning allows gradual changes in network organization to be tracked as a stream of inputs drive the development of the network forward.

The models presented here self-organize using the same principles as the LISSOM (Laterally Interconnected Synergetically Self-Organizing Map) algorithm (Miikkulainen et al., 2005), which was inspired by earlier SOM models (Kohonen, 1982; Obermayer et al., 1990). The simplest model presented here, model L (laterally connected), may be considered a simplified version of the LISSOM model, omitting ad hoc modeler-determined changes to the Hebbian learning rate, activation thresholds, and lateral excitatory radii over time. The fundamental operation of all these self-organizing map models has been explained in terms of dimensionality reduction, specifically a discretized approximation of the principal surface of the input (Ritter et al., 1992).

The architecture of the four models evaluated in this paper is shown in Figure 4. The models each consist of four sheets of neural units representing the input (retinal photoreceptors), the ON-center and OFF-center pathways from the photoreceptors to V1 via the retinal ganglion cells and the lateral geniculate nucleus, and V1. A sheet is a two-dimensional array of firing-rate point neurons, with activation and plasticity equations as described below. The simplest variant, model L (laterally connected) consists of four afferent connections (one from the photoreceptor sheet to each ON/OFF sheets and one from each ON/OFF sheet to V1) with lateral connectivity only in the V1 sheet.

The AL (adaptation, laterally connected) model has the same set of connections as L but includes homeostatic adaptation in V1, described by the equations below. The GCL (gain control, laterally connected) model has the same V1 sheet structure as L but includes lateral, divisive inhibition in the ON/OFF sheets that implement contrast-gain control. The final GCAL model includes both the homeostatic adaptation of the AL model as well as the contrast-gain control of the GCL model. Apart from these specific differences, each model shares the same parameters and mechanisms, and can thus be compared directly against the others.

As model L has the same basic architecture as the LISSOM model, the spatial extents of connections and weight patterns are taken from and described in the study by Miikkulainen et al. (2005). Models GCL and GCAL introduce one new spatial parameter, determining the lateral extent of the contrast-gain control mechanism in the LGN layers. No parameters need to be specified with very high precision for our conclusions to hold; in the GCAL model, two significant digits of precision are sufficient to develop qualitatively indistinguishable maps. We estimate that all parameters may be changed by ~10% without affecting the overall behavior (except for the homeostatic smoothing parameter, which may not be >1.0).

The models are implemented in the Topographica simulator, freely available at www.topographica.org. Topographica allows simulation parameters to be specified in measurement units that are independent of the level of detail used in any particular run of the simulation. To achieve this, Topographica provides multiple spatial coordinate systems, called sheet and matrix coordinates. Parameters (such as sheet dimensions) are expressed in continuous-valued sheet coordinates. In practice, of course, sheets are discretized using some finite matrix of units. Each sheet has a density, which specifies how many units (matrix elements) in the matrix correspond to a unit length in sheet coordinates. Each of these simulations used a density of 98 neural units per sheet coordinate in V1, and 24 units per sheet coordinate in the retinal photoreceptor and ON/OFF sheets. Results are independent of the density used, except at very low densities where discretization artifacts can become prominent.

In all simulations shown, the area of the V1 sheet plotted and analyzed has sheet-coordinate dimensions $1.0 \times 1.0$, thus consisting of $98 \times 98$ neural units. The underlying simulated V1 area is actually $1.5 \times 1.5$ ($147 \times 147$ neural units), which is then cropped to $1.0 \times 1.0$ for analysis and display, to eliminate border effects due to partial patterns of lateral connectivity for neurons near the edge of the cortical sheet. The ON/OFF channels and photoreceptors are similarly extended for simulation to $3.0 \times 3.0$ and $3.75 \times 3.75$ in sheet coordinates, respectively, to avoid having any afferent receptive field cropped off, and to avoid edge effects in the ON and OFF channels due to gain control connections when present. In Figure 4, all sheets are shown at the same size for visibility, but the actual area mapped topographically between sheets was the central $1.5 \times 1.5$ region of each sheet that corresponds to V1.

*Temporal properties.* As a simplification, we have ignored the detailed temporal properties of the subcortical neural responses and of signal propagation along the various types of connections. Instead, the model ON/OFF units have a constant, sustained output, and all connections have a constant delay, independent of the physical length of that connection. One training iteration in the model represents one visual fixation (for natural images) or a snapshot of the relatively slowly changing spatial pattern of spontaneous

activity (for retinal waves); i.e., an iteration consists of a constant retinal activation, followed by processing at the ON/OFF and cortical levels.

For one iteration, assume that input is drawn on the photoreceptors at time $t$, and the connection delay is defined as $\delta t = 0.05$. Then, at $t + 0.05$, the ON/OFF cells compute their initial activation; at time $t + 0.10$, the ON/OFF cells incorporate lateral inhibition for gain control (if present in this model); and at $t + 0.15$, V1 begins computing. At $t + 0.20$ and every 0.05 iterations until $t + 1.0$, V1 continues to compute in a series of settling steps, a fixed number sufficient to ensure that cortical activity is no longer changing significantly (data not shown). During this period, the retinal activity is assumed to be constant, to reduce computational requirements.

*Stimuli.* Images are presented to the model at each iteration by activating the retinal photoreceptor units. The activation value $\psi_i$ of unit $i$ in the photoreceptor sheet ($P$) is given by the gray-scale value in the chosen image at that point.

The images used here are either oriented and elongated two-dimensional Gaussian patterns, disk-shaped patterns of noisy activation, or natural image patches, which together cover a range of different input types so that we can evaluate the robustness of map development. The center coordinates and orientation of each Gaussian are chosen from a uniform random distribution and cover an area of the photoreceptor sheet that is a third wider and taller than V1, so that even V1 units near the borders will see an approximately uniform distribution of each part of the Gaussian blobs. For Gaussian patterns, the contrast is defined as the percentage of the input range 0.0 to 1.0.

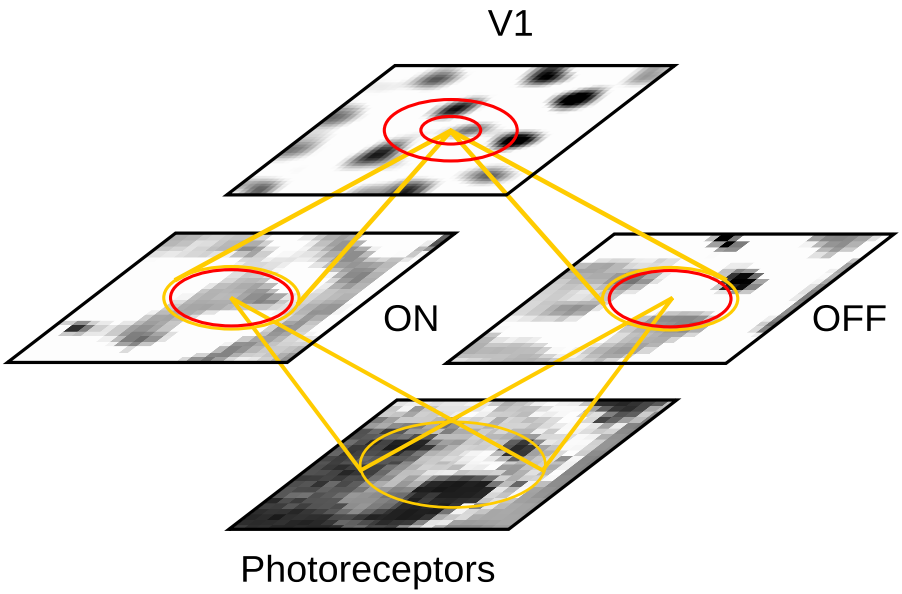


**Figure 4.** General model architecture. The four models discussed in this paper consist of two-dimensional arrays of computational units representing local groups of neurons at each visual processing stage. Connections to one unit in each sheet are shown with afferent connections in yellow and lateral connections in red. The V1 region shown here covers ~4 × 4 mm of ferret cortex, matching the cortical area represented by all the simulated orientation map plots in this paper. V1 units receive lateral excitatory (small circle) and lateral inhibitory (large circle) connections from nearby V1 units, leading to the "bubbles" of activity seen on the V1 sheet. V1 units receive afferent input from sheets representing the ON and OFF channels, representing the action of retinal ganglion cells and the lateral geniculate nucleus, which in turn receive input from the retinal photoreceptors. The difference-of-Gaussian afferent connections from the photoreceptors to the ON and OFF units form a local receptive field on the retina, and cause ON-center units to respond to light areas surrounded by dark, and OFF-center units to dark areas surrounded by light. Neighboring neurons in the ON and OFF sheets have different but overlapping receptive fields. Input to the retinal photoreceptors can be any type of patterned image, such as the small natural image patch shown here. The GCL and GCAL models have additional lateral inhibitory connections within the ON and OFF sheets, but otherwise all four models discussed share identical initial weight profiles, spatial extents, and projection strengths.

Noisy disk patterns create a circular region of activity with a radius that is a substantial portion of the size of the photoreceptor sheet to generate distinct edges; the activation is smoothed at the edges of the disk with a Gaussian blur (Bednar and Miikkulainen, 2004). Disk centers are chosen from a uniform random distribution larger than the retinal photoreceptor sheet size to allow for V1 units to see a uniform distribution of disk areas. Uniform zero-mean random noise is then added to the disk pattern.

Natural image patterns are photoreceptor-sheet-sized patches from images of natural objects and landscapes, from a data set by Shouval et al. (1996). The pictures were taken at Lincoln Woods State Park in Rhode Island, and scanned into a 256 × 256 pixel image. No corrections were used for the optical distortions of the instruments, and there was no preprocessing of the images. The data set used to approximate vertical goggle rearing shown in Figure 11 consists of the same set of image patches, but blurred by convolution with an anisotropic Gaussian kernel. The convolution kernel is a 128 × 128 matrix, consisting of a centered, vertical Gaussian pattern with an aspect ratio of 10 ($\sigma_x = 0.25$, $\sigma_y = 0.025$ in unit sheet coordinates) and total weight of unity.

*The ON/OFF sheets.* At each iteration, a new retinal input is presented, and the activation of each unit in each sheet is updated. Neurons in all sheets are firing-rate point neurons, with a state characterized by an activation level. For all models, the activation level $\eta$ for a unit at position $j$ in an ON/OFF sheet $O$ at time $t + \delta t$ is defined as follows:

$$\eta_{j,O}(t + \delta t) = f\left(\frac{\gamma_O \sum_{i \in F_{j,P}} \Psi_i(t)\omega_{ij}}{k + \gamma_S \sum_{i \in F_{j,S}} \eta_{i,O}(t)\omega_{ij,S}}\right). \quad (2)$$

The constant $\gamma_O = 14.0$ is an arbitrary multiplier for the overall strength of connections from the photoreceptor sheet to the ON/OFF sheets, chosen to give typical activations in the range 0.0 to 1.0, whereas $\gamma_S$ is the strength of the feedforward contrast-gain control; $\psi_i$ is the activation of unit $i$ in the two-dimensional array of neurons on the photoreceptor sheet from which ON/OFF unit $j$ receives input (its afferent connection field $F_{j,P}$), and $\eta_{i,O}(t)$ is the activation of other ON/OFF units on the previous time step (received over the suppressive connection field $F_{j,S}$). The activation function $f$ is a half-wave rectifying function that ensures the activation of ON/OFF units is always positive.

In the L and AL models where contrast-gain control is not applied, $k = 1$ and $\gamma_S = 0$, whereas in the GCL and GCAL models, $k = 0.11$ and $\gamma_S = 0.6$. The constant $k$ ensures that the output is always well defined for weak inputs, and $\gamma_S$ is chosen to rescale activation values so that the numerical results are comparable with and without gain control.

The weights $\omega_{ij}$ represent the fixed connection weights from photoreceptor $i$ to the ON or OFF unit $j$ defined with a standard difference-of-Gaussians kernel. The connection fields for ON units have a positive center and negative surround, and vice versa for OFF units. More precisely, the weight $\omega_{ij}$ from an ON-center cell at location (0, 0) in the ON sheet and a photoreceptor sheet in location $(x, y)$ on the photoreceptor sheet is given by the following:

$$\omega_{ij} = \frac{1}{Z_C}\exp\left(-\frac{x^2 + y^2}{2\sigma_C^2}\right) - \frac{1}{Z_S}\exp\left(-\frac{x^2 + y^2}{2\sigma_S^2}\right). \quad (3)$$

The width of the central Gaussian is defined by $\sigma_c = 0.037$, and $\sigma_s = 0.15$ determines the width of the surround Gaussian, where $Z_c$ and $Z_s$ denote the normalization constants that ensure the center and surround weights each always sum to 1.0. The weights for an OFF-center cell are the negative of the ON-center weights (i.e., surround minus center). The center of the connection field of each ON/OFF unit is mapped to the location in the photoreceptor sheet corresponding to the location of that unit in sheet coordinates, making the projection retinotopic.

The weights $\omega_{ij,S}$ in the denominator of Equation 2 specify the spatial profile of the lateral inhibition received from other ON/OFF units when contrast-gain control is active. The weights of these connections have a fixed, circular Gaussian profile so that a neuron located at (0, 0) in either the ON or OFF sheet will have the following weights:

$$\omega_{ij,S} = \frac{1}{Z_S} \exp\left(-\frac{x^2 + y^2}{2\sigma_S^2}\right), \quad (4)$$

where $(x, y)$ is the location of the presynaptic neuron, $\sigma_S = 0.125$ determines the width of the Gaussian, and $Z_S$ is a normalizing constant that ensures that the total of all the lateral inhibitory weights $\omega_{ij}$ to neuron $j$ sum to 1.0. When contrast-gain control is enabled, these recurrent lateral connections are activated once per iteration, before activity is sent to the V1 sheet.

*The V1 sheet.* Each V1 neuron in each model receives connections from three different connection types or "projections" ($p$), i.e., the afferent projection from the ON/OFF sheets (both channels concatenated into one input vector; $p = \mathrm{A}$), the recurrent lateral excitatory projection ($p = \mathrm{E}$), and the recurrent lateral inhibitory projection ($p = \mathrm{I}$) from other V1 neurons.

The contribution $C_{j,p}$ to the activation of unit $j$ from each projection type ($p = \mathrm{A}, \mathrm{E}, \mathrm{I}$) is calculated as follows:

$$C_{j,p}(t + \delta t) = \sum_{i \in F_{j,p}} \eta_{i,p}(t)\omega_{ij,p}, \quad (5)$$

where $\eta_{i,p}$ is the activation of unit $i$ taken from the set of neurons in V1 to which unit $j$ is connected (its connection field $F_j$), and $w_{ij,p}$ is the connection weight from unit $i$ in V1 to unit $j$ in V1 for the projection $p$. Afferent activity ($p = \mathrm{A}$) remains constant after the first update from the retina, but the other contributions change over 16 settling steps, depending on the activity in V1.

The contributions from all three projections to V1 (afferent and lateral) described above are combined using Equation 6 to calculate the activation of a neuron $j$ in V1 at time $t$:

$$\eta_{j,V}(t) = f\left(\sum_p \gamma_p C_{jp}(t)\right). \quad (6)$$

The projection strength scaling factors for each projection type $p$ are $\gamma_\mathrm{A} = 1.5$, $\gamma_\mathrm{E} = 1.7$, and $\gamma_\mathrm{I} = -1.4$ for all models, set to provide a balance between excitation and inhibition, and between afferent and lateral influences, to provide robust formation of activity bubbles that allows smooth maps to form. For models L and GCL, $f$ is a half-wave rectifying function that ensures positive activation values. In the case where single-neuron automatic adaptation is included (models AL and GCAL), $f$ has a variable threshold point ($\theta$) dependent on the average activity of the unit as described in the next subsection, but in all cases the gain is fixed at unity.

Once all 16 settling steps are complete, the settled V1 activation pattern is deemed to be the V1 response to the presented pattern. At this point we use the V1 response to update the threshold point ($\theta$) of V1 neurons (using the adaptation process described below) and to update the afferent and lateral inhibitory weights via Hebbian learning. V1 activity is then reset to zero, and a new pattern is presented. Note that both adaptation and learning could instead be performed at every settling step, but this would greatly decrease computational efficiency.

*Adaptation.* To set the threshold for activation, each neuron unit $j$ in V1 calculates a smoothed exponential average of its settled activity patterns ($\bar{\eta}_j$):

$$\bar{\eta}_j(t) = (1 - \beta)\eta_j(t) + \beta\bar{\eta}_j(t - 1). \quad (7)$$

The smoothing parameter ($\beta = 0.991$) determines the degree of smoothing in the calculation of the average. ($\bar{\eta}_j$) is initialized to the target average V1 unit activity ($\mu$), which for all simulations is $\bar{\eta}_{jA}(0) = \mu = 0.024$. The threshold is updated using the following:

$$\theta(t) = \theta(t - 1) + \lambda(\bar{\eta}_j(t) - \mu), \quad (8)$$

where $\lambda = 0.01$ is the homeostatic learning rate. The effect of this scaling mechanism is to bring the average activity of each V1 unit closer to the specified target. If the activity in a V1 unit moves away from the target during training, the threshold for activation is thus automatically raised or lowered to bring it closer to the target. Note that an alternative rule with only a single smoothing parameter (rather than $\beta$ and $\lambda$) could be formulated, but the rule as presented here makes it simple for the modeler to set a desired target activity $\mu$.

*Learning.* Initial connection field weights are isotropic 2D Gaussians for the lateral excitatory projection and uniformly random within a Gaussian envelope for afferent and lateral inhibitory projections. Specifically, a neuron located at (i, j) will have the following weights:

$$\omega_{ij} = \frac{1}{Z_p} u \exp\left(-\frac{x^2 + y^2}{2\sigma_p^2}\right), \quad (9)$$

where $(x, y)$ is the sheet-coordinate location of the presynaptic neuron, $u = 1$ for the lateral excitatory projection ($p = \mathrm{E}$), and $u$ is a scalar value drawn from a uniform random distribution for the afferent and lateral inhibitory projections ($p = \mathrm{A}, \mathrm{I}$), $\sigma_p$ determines the width of the Gaussian in sheet coordinates ($\sigma_\mathrm{A} = 0.27$, $\sigma_\mathrm{E} = 0.025$, $\sigma_\mathrm{I} = 0.075$), and $Z_p$ is a constant normalizing term that ensures that the total of all weights $\omega_{ij}$ to neuron $j$ in projection $p$ is 1.0. Weights for each projection are only defined within a specific maximum circular radius $r_p$ ($r_\mathrm{A} = 0.27$, $r_\mathrm{E} = 0.1$, $r_\mathrm{I} = 0.23$).

In the model, as images are presented to the photoreceptors, V1 afferent connection weights $\omega_{ij,A}$ from the ON/OFF sheets are adjusted once per iteration (after V1 settling is completed) using a simple Hebbian learning rule. This rule results in connections that reflect correlations between the presynaptic ON/OFF unit activities and the postsynaptic V1 response. Hebbian connection weight adjustment at each iteration is dependent on the presynaptic activity, the postsynaptic response, and the Hebbian learning rate:

$$\omega_{ij,p}(t) = \frac{\omega_{ij,p}(t - 1) + \alpha\eta_j\eta_i}{\sum_k(\omega_{kj,p}(t - 1) + \alpha\eta_j\eta_k)}, \quad (10)$$

where for unit $j$, $\alpha$ is the Hebbian learning rate for the afferent connection field $F_j$. Unless it is constrained, Hebbian learning will lead to ever-increasing (and thus unstable) values of the weights (Rochester et al., 1956). In all of the models, the weights are constrained using divisive postsynaptic weight normalization (Eq. 10), which is a simple and well-understood mechanism. Afferent connection weights from ON and OFF units are normalized together in the model. We expect that a more biologically motivated homeostatic mechanism for normalization such as multiplicative synaptic scaling (Turrigiano, 1999; Turrigiano and Nelson, 2004; Sullivan and de Sa, 2006) or a sliding threshold for plasticity (Bienenstock et al., 1982) would achieve similar results, but we have not tested these.

The learning rates for the afferent projection, lateral excitatory projection, and lateral inhibitory projections are $\alpha_\mathrm{A} = 0.1$, $\alpha_\mathrm{E} = 0.0$, and $\alpha_\mathrm{I} = 0.3$, respectively. The density-specific value used in the equation above is then calculated as $\alpha = \alpha_\mathrm{A}/\tau_\mathrm{A}$, where $\tau_\mathrm{A}$ is the number of connections per connection field in the afferent projection. To increase computational efficiency, lateral excitatory connections do not learn during development in the simulations presented here. The effect of lateral learning has been explored in detail in previous similar models (Miikkulainen et al., 2005), and the maps generated by the GCAL model are visually indistinguishable regardless of whether or not lateral excitatory connections are plastic (data not shown).

*Analysis of model maps.* Model orientation maps are calculated based on the vector average method (Blasdel and Salama, 1986; Miikkulainen et al., 2005). Sine grating inputs that cover the full range of parameter values (combinations of all orientations, frequencies, and phases) are presented, and for each orientation, the peak response of the neuron is recorded. The orientation preference is calculated by constructing a vector for each orientation $\theta$ (between 0 and 180°), with the peak response as the length and $2\theta$ as its orientation. These vectors are summed and the preferred orientation is calculated as half of the orientation of the summed vector. The selectivity is given by the magnitude of the summed vector. Average orientation selectivity values are the mean value of orientation selectivity across all units in the map and are normalized by dividing by the maximum average selectivity measured across all simulations reported in this paper.

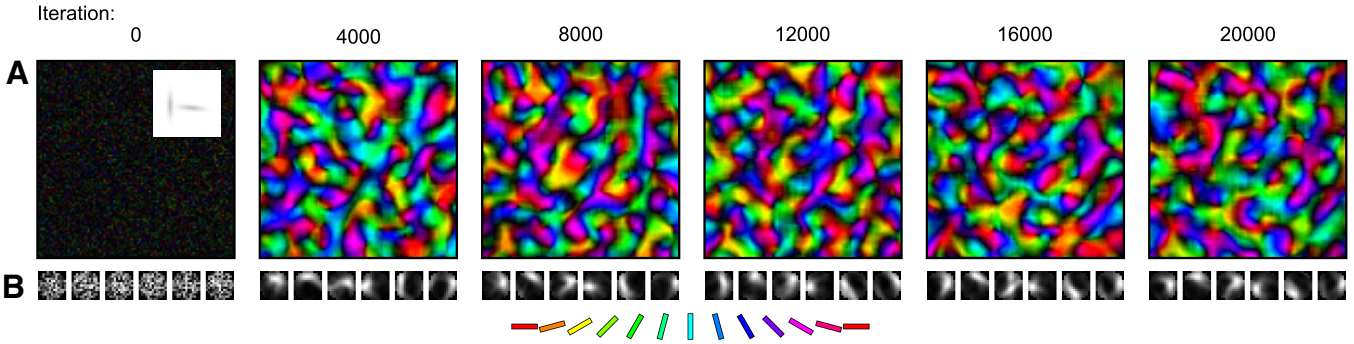


**Figure 5.** Model L develops maps, but is not stable. ***A***, Model development at six different iteration time points for a single simulation. Self-organization was driven by two elongated Gaussian patterns at 25% contrast per iteration, with an example inset into the plot for iteration 0. Polar orientation maps from the beginning of development to the final map at iteration 20,000 are shown. Each unit is color coded according to orientation preference, as shown by the color key. The brightness and saturation of the color indicates the strength of orientation tuning of the afferent connections, and each panel corresponds to ~4 × 4 mm of visual cortex (a 1.0 × 1.0 area in sheet coordinates). ***B***, Afferent connections from the ON sheet to V1 are shown for an arbitrary set of V1 units throughout development. Initially random connections are strengthened and weakened by Hebbian learning, forming orientation-selective receptive fields, but the map patterns change significantly over time. These results, while an improvement over existing models that have additional, biologically implausible mechanisms for reducing stability to improve their final map organization, represent a baseline for the results of the later models (AL, GCL, and GCAL). The behavior of L is analyzed further in Figure 6, and the last map shown corresponds to Figure 6*E*.

Orientation tuning curves are measured by presenting sine gratings with 20 different orientations, each at eight different phases and for a range of contrasts. For each contrast, the responses of each neuron unit are measured, and the maximum response at each orientation over all phases is recorded. The tuning curve is constructed from these maximum responses.

To ensure consistent map measurement across all conditions, orientation maps in Figures 3–9 are measured before lateral interactions and the V1 activation function take affect (i.e., on the afferent input activity only). Maps for models with gain control change little if these steps are included. However, for models without gain control, if activation thresholds and lateral interactions were simulated during map measurement, maps would need to be measured at many different contrasts, because orientation tuning in these models is not contrast invariant, and therefore orientation selectivity is difficult to determine. Instead, we use the orientation selectivity of the afferent connections as a measure of the map selectivity in all cases. This simplification allows fair and consistent comparison of selectivity values between the simulations of all four models. Similarly, connection field plots show the weights of connections between the ON sheet and V1 at different times during development. These weights are used as an approximation to the receptive fields that can be measured using a computationally intensive reverse correlation process that takes into account the full connectivity, which again must control for contrast in the models without gain control. The true orientation selectivity of neurons in the full recurrent network is reflected in the orientation tuning curves for the GCAL model (as shown in Fig. 12).

## Results

In this section we present results from four closely related developmental models of orientation map formation, all sharing the architecture illustrated in Figure 4. As described in Materials and Methods, each model is a network of single-compartment firing-rate units, each receiving afferent and lateral connections that are modified by Hebbian learning. The models share the same four sheets of simulated neurons representing the pathway from the retina to V1, with the same dimensions, connection radii, initial weights, and all other parameter settings.

The robustness of each model will initially be analyzed across a wide range of input contrasts for an artificial input pattern (randomly positioned and oriented elongated two-dimensional Gaussians). These simple patterns help make our reasoning about how each mechanism affects map development clear and intuitive and are sufficient for evaluating each model's robustness to input contrast. In turn, contrast is used as a well-defined and easily characterized proxy for a variety of changes in input patterns that could affect how much V1 is activated for a given input, and could thus affect stability and robustness. Once all four models have been evaluated with respect to contrast for Gaussians, we will show that the results still hold (with no changes in model parameters) for more complex changes in the input images that are more difficult to characterize.

For each model, we evaluate the orientation selectivity, map stability, and map quality, and relate the results to data from ferrets. The first model is a simple laterally connected model of orientation map development that we will refer to as model L. Model L retains the essential features of previous models in this genre (Sirosh and Miikkulainen, 1997; Burger and Lang, 1999, 2001; Miikkulainen et al., 2005), while omitting biologically unrealistic mechanisms like lateral radius shrinking that also reduce stability, as well as eliminating arbitrary manual interventions like threshold changes or learning rate changes over development.

Subsequent models build on L, adding single-neuron homeostatic threshold adaptation (model AL), feedforward gain control (model GCL), and finally both mechanisms together (model GCAL). Justification for each mechanism is discussed as it is introduced, but the details of these two mechanisms are described in Materials and Methods. The results show that as well as reproducing stable development, the final GCAL model is robust to extreme changes in the input statistics, even during the process of development.

### Model L: a simple model of map development

Results from simulated development in the L model are shown in Figure 5. Each V1 model neuron receives excitatory connections from an already retinotopically aligned set of ON/OFF neurons and excitatory and inhibitory connections from other V1 neurons (Fig. 4). At each iteration, two elongated and oriented Gaussian patterns are presented on the model retina. The afferent and lateral connections lead to the formation of isolated bubbles of activity in V1 in response to the input pattern. Initially random afferent weights between the ON/OFF units and V1 then adapt based on the presynaptic and postsynaptic activity, via a simple Hebbian learning rule with divisive normalization (Eq. 10). Neurons in each bubble thus learn their topographically corresponding input pattern, while neurons in other bubbles learn different patterns. Over the course of development (20,000 iterations),

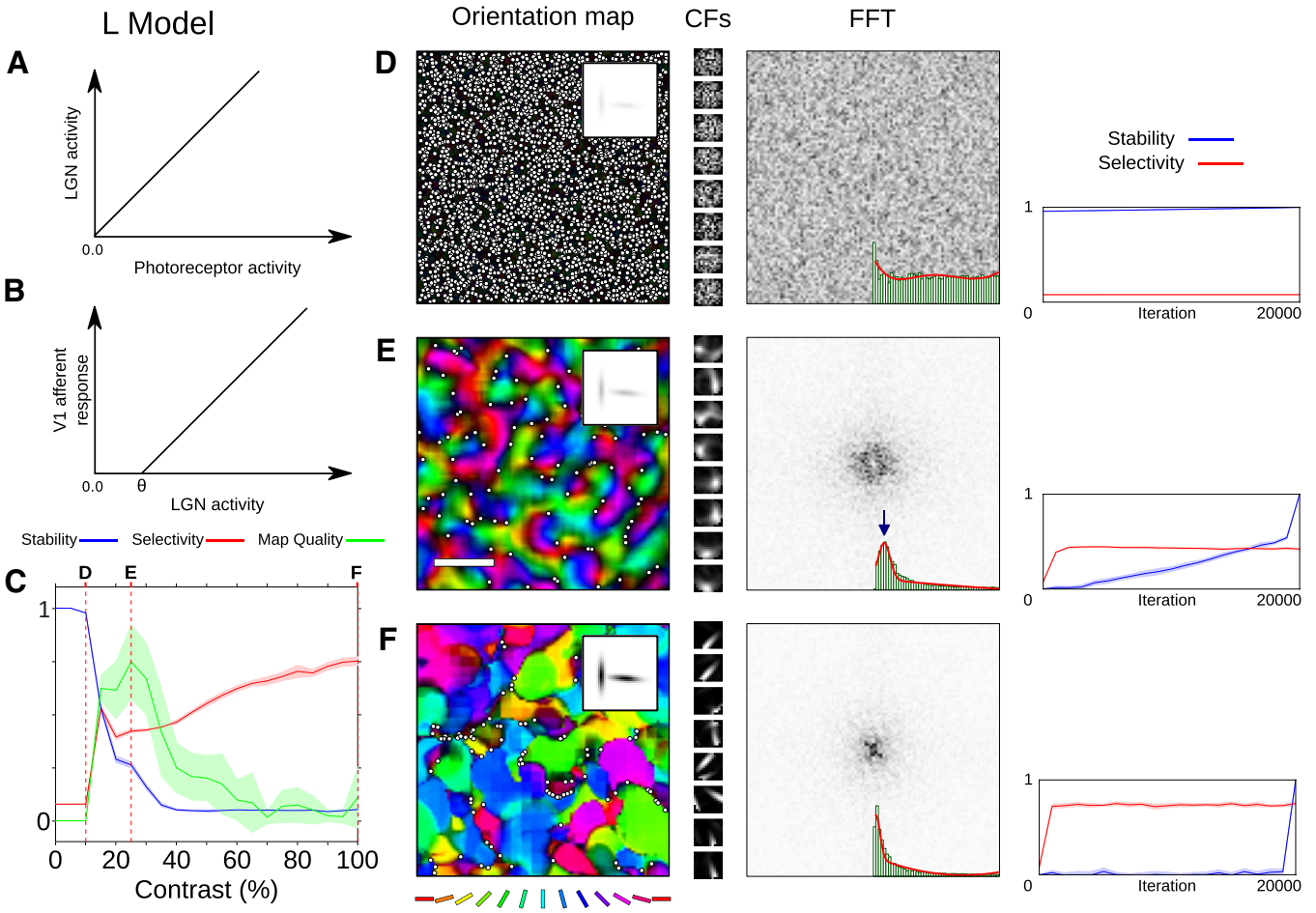


**Figure 6.** Model L: basic model has relatively poor stability, map quality, and robustness to contrast. ***A***, ***B***, The transfer functions from photoreceptor activity to LGN activity (***A***) and from LGN activity to V1 afferent response (***B***). Both transfer functions have unit slope, and V1 units have a fixed threshold of $\theta = 0.2$. ***C***, The mean map stability (green), selectivity (red), and map quality (blue) as a function of contrast, across 10 simulations with randomized input sequences. Shaded areas indicate ±95% confidence intervals. For low-contrast inputs, neurons are not activated, and thus no learning occurs (point D); at slightly higher contrasts neurons have higher stability and selectivity (point E), but at the highest contrasts (point F) neurons are very unstable. Neurons have a relatively high selectivity and stability in a small range of contrasts for which the model has been tuned (contrasts 15–30%). ***D–F***, Organization of model L at the end of development (iteration 20,000) at 10% (***D***), 25% (***E***), and 100% (***F***) contrast. Polar orientation preference maps with estimated pinwheel positions, along with sample inputs (inset) and afferent connection fields (CFs) from the ON sheet to V1 neurons for an arbitrary selection of model neurons (evenly spaced along the vertical midline of the map), are shown. The corresponding two-dimensional FFTs are shown, with 1D the spectral power histogram (green), function fit (red), and estimated peak position (blue arrow) used to estimate the hypercolumn distance (white scale bar). These values determine the value of the map metric, which is fairly poor even for the best L maps due to the large number of pinwheels identified. On the right, the stability and selectivity are plotted as a function of simulation time for that contrast, showing the average across all 10 random seeds with the ±95% confidence intervals. ***D***, For low-contrast inputs (contrast 10%) orientation maps do not develop, yielding nominally high stability values that are meaningless because the selectivity remains low. ***E***, Maps are well ordered, and development is somewhat stable, within the "tuned" range of contrasts (contrast 25%, as in Fig. 5). ***F***, Orientation maps for high-contrast patterns (100% contrast) are highly disorganized, with sharp boundaries between hypercolumns and a non-ring-shaped FFT. Afferent connections are highly orientation-selective imprints of the elongated Gaussians presented to the photoreceptor sheet, and map development is highly unstable, indicated by an early rise in selectivity without a corresponding increase in stability. Note that the final value of each stability plot will always be 1.0; the final map compared with itself has a stability index of unity. Overall, although L develops maps, it fails to be robust to contrast, develops relatively poor quality maps, and is not very stable compared to the ferret data.

nearly all receptive fields become orientation selective, and a topographic organization for orientation is formed (Fig. 5). This topographic organization arises because throughout the development process, nearby neurons are similarly activated, and therefore develop similar orientation preferences. This process has been well documented in previous models (Burger and Lang, 1999; Miikkulainen et al., 2005).

L is a greatly simplified model, explicitly designed to demonstrate this process of self-organization using a minimal set of mechanisms. The afferent input to V1 is a linear rectified function of the image contrast (since the output of ON/OFF units is a linear function of the overall input from the retinal photoreceptors), as shown in Figure 6*A*. The activity of each LGN unit is the dot product of the photoreceptor input and a normalized difference-of-Gaussian ON or OFF receptive field, and the relationship between LGN activity and V1 afferent response is also linear. Unlike the LGN units, a V1 unit will not respond until the sum of its input activities exceeds a fixed minimum threshold, $\theta$ (Fig. 6*B*).

The robustness of this model to changes in input stimulus is evaluated by the behavior of the model as a function of input contrast, without changing any other parameters. Although there are many other dimensions over which these inputs could be varied (e.g., number of Gaussian patterns, size of pattern, amount of noise), robustness to contrast is an intuitive feature that can be easily evaluated and related to the visual system. The analysis of this simple model is shown in Figure 6*C*. For each contrast, the average stability over development (measured at intervals of 1000 iterations), the average orientation selectivity of afferent connections in the final map (iteration 20,000), and the average map quality of the final map

are computed by taking the mean results of 10 different random seeds controlling the pattern of training inputs. These three measures give a good indication of the stability, robustness, and quality of map development for each input image contrast. A fixed set of three contrasts has been chosen for illustration and analysis in all four models: the low-contrast point (Fig. 6*D*), the medium-contrast point (*E*), and the high-contrast point (*F*). The medium contrast level of 25% is also shown in Figure 5 and was selected as the peak of the map metric for the L model.

Figure 6*C* shows that low-contrast inputs to the L model (15–20% contrast) can result in higher stability but lower map quality than shown in Figure 5. At these low contrasts, neurons become moderately selective, and a small degree of stability can be achieved because, as the receptive fields of the model neurons form, connections that have been strengthened by Hebbian learning continued to strengthen throughout development, rather than being overwritten by the strengthening of a different set of connections. Figure 3*D* illustrates that the pinwheel densities at these contrasts are reliably distinguishable from biological maps; average results from 33 simulations of the L model at 15% contrast are shown.

In this model, contrasts of 10% and below (Fig. 6*D*) lead to a complete lack of map development. If the presynaptic ON/OFF activity is not strong enough to consistently exceed the necessary fixed threshold $\theta$ to activate V1 neurons, connections will not change, and receptive fields will not form. Such a developmental process is thus completely stable at the cost of achieving any degree of selectivity. Although this is a trivial property of the L model, the ability of the early visual system to ensure that V1 is sufficiently activated is an important consideration for understanding visual development.

As input image contrast increases, both the stability of map development and the overall level of map organization degrades rapidly, resulting in maps such as the one shown in Figure 6*F* at high contrast. The map organization of the orientation map in Figure 6*F* can be seen to be very poor, as reflected by the non ring-shaped structure in the Fourier power spectrum. These observations hold across all 10 randomized simulation runs, as demonstrated by the low value of the $\pi$ pinwheel density metric for all high-contrast measurements (Fig. 6*C*).

In general, the L model is not robust to contrast. Unlike the experimental data shown in Figure 1*A*, selectivity is achieved rapidly, but map stability over development is not maintained. There are several key properties of this simple model that make it a poor model of robust map development in animals. First, the linearity of activity in the LGN and V1 with respect to photoreceptor activity ensures increases in contrast directly lead to corresponding increases in activity in ON/OFF units as well as in V1. Second, the lateral interactions in the model depend on the level of activation of V1 units and further amplify activity in V1 as input image contrast increases. Third, as the afferent weights in the model develop, receptive fields of V1 neurons become more orientation selective and therefore match the patterns in the input more closely, further increasing the postsynaptic responses of V1 neurons. These three properties ensure that during development, when inputs are high contrast, both ON/OFF and V1 activity will also be high.

With a fixed Hebbian learning rate, the consequence of simultaneously elevated presynaptic and postsynaptic activity (from the high ON/OFF and V1 responses respectively) ensures that the amount of Hebbian learning is increased (Eq. 10). Therefore, with high-contrast input, connections between ON/OFF and V1 cells are more likely to be overwritten as differently oriented patterns are presented, leading to continual reorganization of receptive fields and a corresponding loss of stability and overall map quality.

Conversely, selectivity of receptive fields increases as contrast is increased. This is because, as contrast increases, connections begin to strengthen and weaken on a faster time scale, eventually becoming indicative of the most recent pattern on the retinal photoreceptors (which in this case will be a highly selective, elongated Gaussian pattern) rather than learning to represent longer-term correlations in the afferent input.

This simple model, although capable of forming orientation-selective receptive fields and smooth topographic organization for orientation (given a particular set of parameters), is not robust even to small changes in the input image contrast. Since experimental results indicate that map development is robust and stable over a time scale relatively consistent between animals, and that the input activity properties are likely to change during development (for example, before and after eye opening), it is necessary for the visual system to have some underlying mechanism that compensates for changes in the type of driving input.

The following sections show how feedforward gain control and automatic adaptation of the neuronal activation thresholds can improve the stability of orientation map development and robustness in response to changes in the input image properties. We then show how inclusion of both these simple and biologically motivated mechanisms results in the development of other emergent properties of V1 neurons, such as contrast-invariant orientation tuning.

### Model AL: homeostatic adaptation regulates postsynaptic activity

In the L model, the fixed activity threshold across all V1 units was found to constrain robustness to contrast. For any positive threshold value, there will be a range of low contrasts at which the input to V1 units is below threshold and no development occurs. At high contrasts, the postsynaptic activity of the V1 units will be unbounded, resulting in map instabilities due to rapid Hebbian learning constantly reorganizing receptive fields across the map.

In biological V1, the firing rates of different cortical neurons cannot be characterized by a single, linear (and therefore unbounded) function of synaptic input. Unlike the linear model units with fixed threshold, real neurons are thought to have intrinsic homeostatic plasticity, whereby each neuron regulates the level of synaptic input required to generate an action potential by changing the number and distribution of its ion channels (Davis and Bezprozvanny, 2001; Daoudal and Debanne, 2003; Zhang and Linden, 2003; Schulz, 2006; Grubb and Burrone, 2010; Kuba et al., 2010). Such mechanisms allow each neuron to adapt its firing threshold to maintain a suitable average activity as incoming synaptic drive varies over time.

In the AL model, a simple model of homeostatic adaptation is introduced to the cortical sheet, allowing individual V1 units to adjust their firing thresholds to establish a suitable average target activity (Eq. 8). The response of the ON/OFF units remains linear with respect to the photoreceptor activity (Fig. 7*A*); the new relationship between ON/OFF response and V1 afferent activity is illustrated by Figure 7*B*.

The quality and robustness of AL across input contrast is greatly improved by the introduction of adaptive thresholds, suggesting that intrinsic homeostatic adaptation plays an important role in robust map development (Fig. 7*C*). With adaptation, maps have more realistic pinwheel densities, and stability drops off more slowly at high contrasts. At the low-contrast point

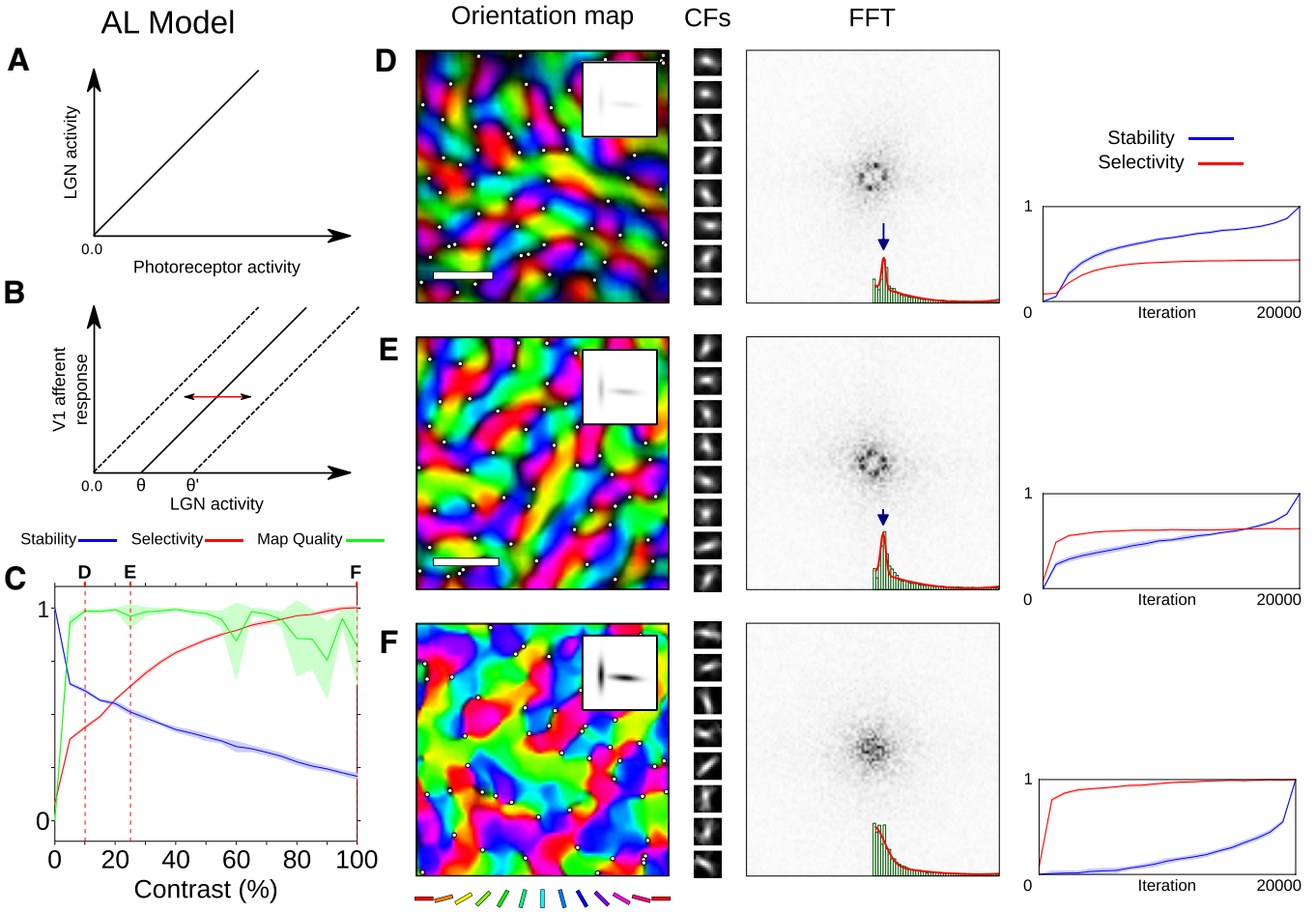


**Figure 7.** Model AL: adding homeostatic threshold adaptation improves selectivity, stability, and map quality across all contrasts. All plotting conventions (colors, symbols, and scale bars) are as in Figure 6. ***A***, The transfer function from photoreceptor activity to LGN activity remains unchanged. ***B***, Each V1 unit now possesses an independent adaptive threshold $\theta$ that is automatically adjusted to maintain a fixed target activity. ***C***, Maps are more stable across contrast, with higher selectivities at high contrast than the L model. Map quality is relatively high throughout, with a drop at high contrasts. ***D***, The AL model can respond and self-organize at lower contrasts by lowering the adaptive threshold of V1 units. ***E***, AL self-organizes into higher-quality maps than L at identical contrasts, though stability has suffered compared to ***D***. ***F***, Adaptation greatly improves the map quality relative to the L model at high contrasts. However, the map still suffers from sharp boundaries due to highly selective connection fields (CFs) that are imprints of the elongated Gaussians presented to the photoreceptor layer, and the FFT becomes non-ring-shaped at high contrast. Stability is also very poor (with selectivity achieved long before stability), because the afferent weights continually reorganize at high contrasts. Thus, homeostatic adaptation offers significant benefits over the L model, but is not sufficiently robust or stable to account for the animal data.

shown in Figure 7*D*, self-organization is able to proceed as the adaptive thresholds fall, ensuring activity in the V1 sheet and compensating for the lower input levels.

At the mid-contrast point shown in Figure 7*E*, selectivity and stability are higher and the maps are better organized than in the L model. This is demonstrated by the higher map metric, which reflects the greater spacing between pinwheels in the orientation map and the increasingly ring-shaped Fourier power spectrum.

At high contrasts, selectivity continues to increase but map quality and stability start to fall. The orientation map shown in Figure 7*F* is highly selective everywhere, including around the identified pinwheel locations. The result is a map with sharp boundaries between regions of different orientation preference instead of smooth transitions. The variance in pinwheel density increases as the ring structure in the FFT degrades, due to the hypercolumn distance becoming poorly defined in Fourier space. The high selectivity is also evident in the highly elongated weight profiles of the afferent connectivity from the ON/OFF sheets to the V1 units.

The failure of the AL model at high contrast is due to the same fundamental problem identified in the L model: rapid Hebbian learning in the afferent connectivity from the LGN to V1. Although the postsynaptic activity is now homeostatically regulated, which keeps the intracortical circuitry working in a well-defined range, the presynaptic activity in the ON and OFF layers remains unbounded. The result is an effectively very high afferent learning rate, causing recent inputs to be memorized rather than being incorporated smoothly into the map, and destroying stability. Moderating the postsynaptic activity can only partially address this imbalance, so long as the activity in the presynaptic ON and OFF sheets remains unbounded.

### Model GCL: gain control compensates for differences in input strength

Rather than compensating for high presynaptic activity by regulating postsynaptic excitability, an alternative and perhaps more direct approach would be to limit the range of possible activity levels reaching V1. This change would ensure that learning in the afferent connections between the ON/OFF sheets and V1 cannot be disrupted by high presynaptic activity.

Contrast-gain control, also known as normalization, is a well-documented phenomenon whereby neurons have a nonlinear

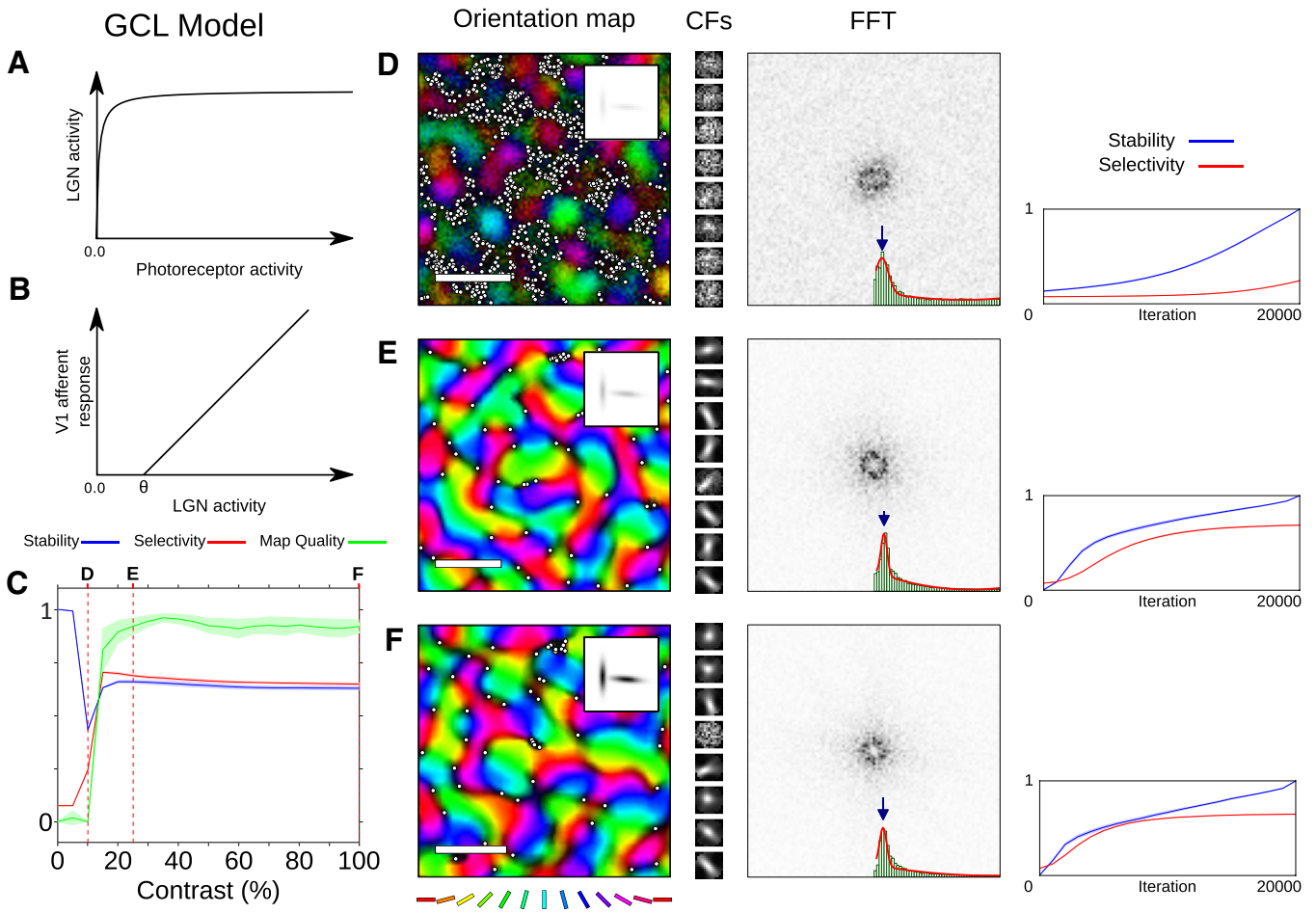


**Figure 8.** Model GCL: adding contrast-gain control independently improves stability and map quality. All plotting conventions (colors, symbols, and scale bars) are as in Figure 6. ***A***, Contrast-gain control in the ON/OFF sheets results in a nonlinear transfer function, compressing unbounded photoreceptor inputs into a bounded range of LGN activities. ***B***, V1 units share the same fixed threshold as those in the L model. ***C***, Map quality is improved across all contrasts and map stability no longer degrades as contrast increases. Selectivity remains both high and stable with increasing contrast. ***D***, At the low-contrast point, GCAL can just begin to self-organize as contrast-gain control boosts enough of the afferent signal over the fixed V1 threshold for some Hebbian learning to occur. ***E***, GCL self-organizes into higher quality maps than L at the same 25% contrast level. Unlike the AL model, some small areas of the map fail to self-organize properly, resulting in regions of low selectivity and clusters of pinwheels. ***F***, For high (100%) contrast inputs, map quality remains consistently high, connection fields (CFs) remain well formed, the map remains smooth and the FFT is appropriately ring shaped. However, small areas of the map fail to self-organize or develop selectivity, as shown by the noisy connection field. Overall, gain control supports robust and stable map development across contrasts, but some neurons are left behind as the others develop, and cannot reach threshold, leading to unevenly organized maps.

response to input strength, compressing a wide range of inputs into a smaller range of responses (Bonin et al., 2005). There is a wealth of experimental evidence supporting the idea that contrast-gain control first arises early in the visual pathway (Shapley and Victor, 1978; Derrington and Lennie, 1984; Sclar, 1987; Truchard et al., 2000; Baccus and Meister, 2002; Alitto and Usrey, 2004). A wide variety of candidate mechanisms have been proposed (Carandini et al., 2002; Geisler and Albrecht, 1997; Finn et al., 2007; Anderson et al., 2000; for review, see Carandini and Heeger, 2012).

In the GCL model, contrast-gain control is implemented using divisive inhibitory lateral connections between model ON/OFF neurons, providing an "extraclassical" supressive surround (Felisberti and Derrington, 1999; Bonin et al., 2005; Alitto and Usrey, 2008). We would expect to obtain similar results using the other mechanisms that have been proposed, but have not tested them. The addition of gain control to the L model ensures that the afferent input to V1 is no longer a linear function of the input image contrast, as shown in Figure 8*A*. Instead, the afferent input to V1 saturates with increasing contrast.

Figure 8*C* shows that gain control in the ON/OFF sheets is sufficient to greatly increase the robustness and stability of the orientation map development process. By compensating for differences in the presynaptic input to V1, stability is greatly improved, with high map quality across most contrasts. Stability no longer degrades at high contrast, selectivity remains stable even at high contrast, and map quality remains consistently high.

At the low-contrast point shown in Figure 8*D*, contrast-gain control has enabled some self-organization, boosting the weak input signal over the firing threshold in the cortical layer. The orientation map is not as well organized as the corresponding AL map (Fig. 7*D*), but this result does demonstrate that this contrast-gain control mechanism amplifies weak afferent signals for low-contrast inputs.

At the medium contrast level shown in Figure 8*E*, the orientation map is comparable to the equivalent AL map. One common type of artifact observed in GCL are the pinwheel clusters found in low-selectivity regions, which do not appear in maps that have adaptive thresholds in the cortical layer (Fig. 7). These pinwheel clusters persist even in the high-contrast regime shown in Figure 8*F*.

The origin of these artifacts is illustrated by the afferent weight pattern shown in Figure 8*F* (fourth pattern down). This set of weights is noisy and has failed to self-organize, indicating areas of the map lagging in their development.

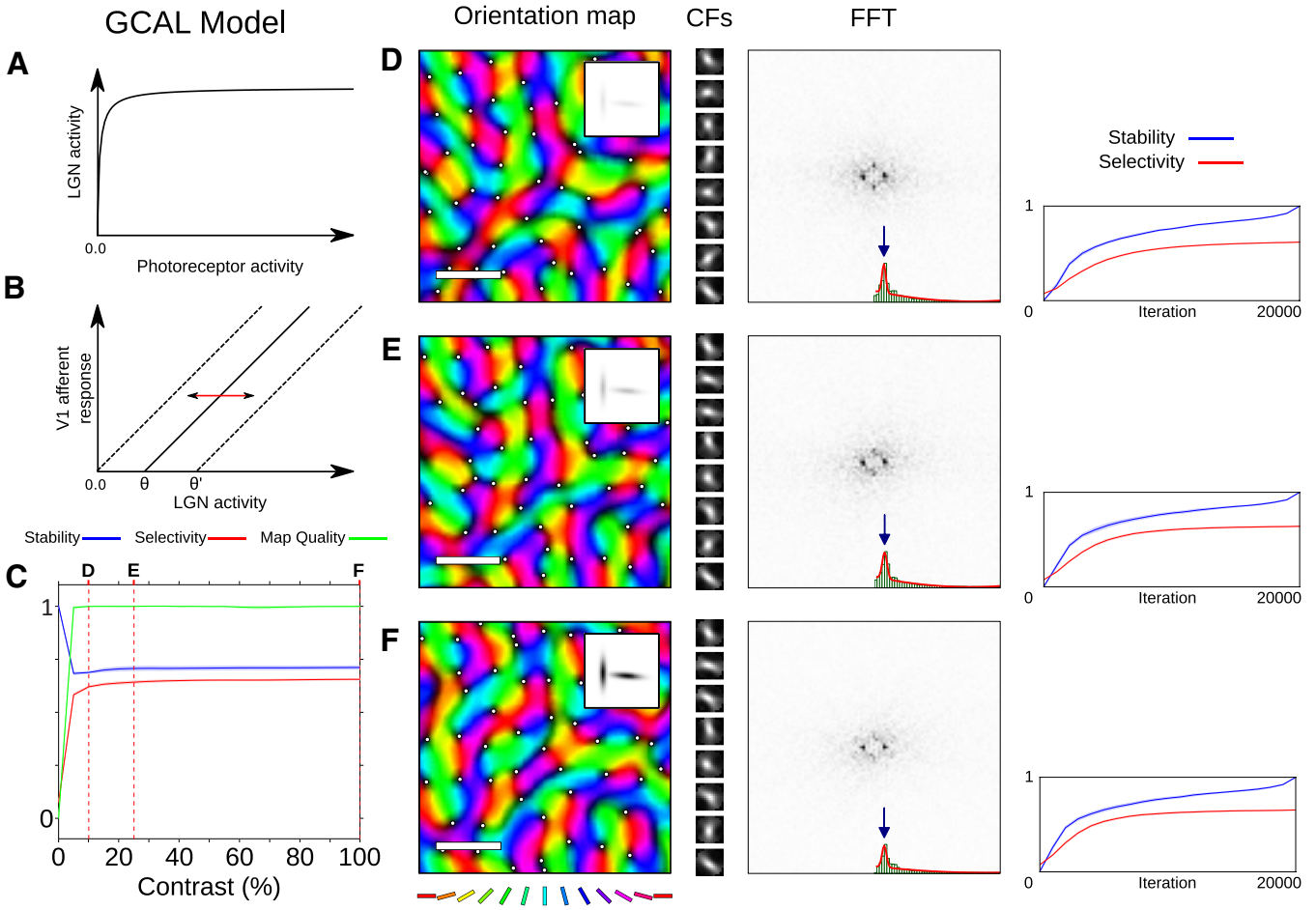


**Figure 9.** Model GCAL: combining homeostatic adaptation with gain control yields high-quality, selective, stable maps at all contrasts. All plotting conventions (colors, symbols, and scale bars) are as in Figure 6. ***A***, The contrast-gain control mechanism introduced in GCL is retained. ***B***, V1 units now have the same adaptive threshold ($\theta$) as in the AL model. ***C***, Map quality metric indicates that all GCAL maps have very close to $\pi$ pinwheel density for all simulated orientation maps (also see Fig. 3*D*), with high stability and selectivity across all contrasts. ***D–F***, Maps are properly self-organized at all contrasts, with ring-shaped FFTs. At high contrast, GCAL remains smooth and does not suffer from low-selectivity pinwheel clusters or noisy connection fields (CFs) due to the introduction of the homeostatic threshold. Stability increases alongside selectivity, indicating highly stable map development. The GCAL model robustly generates high-selectivity, highly stable, high-quality maps.

These areas of poor organization arise when neighboring cortical regions develop orientation selectivity slightly more quickly, increasing the postsynaptic activity of these surrounding regions. This increase in the activity of neighboring units increases the strength of the suppressive lateral interactions, further suppressing the development of the region already lagging in self-organization. This positive-feedback loop ensures that some areas of the map fail to develop properly. Thus, although GCL is a clear improvement over both L and AL, it does not robustly organize into biological maps that include all of the simulated neurons.

### Model GCAL: adaptation compensates for differences in V1 activation

Homeostatic adaptation and contrast-gain control independently improved map quality, selectivity, and robustness, but both mechanisms have been found to have specific shortcomings. The final GCAL model combines the ON/OFF contrast-gain control mechanism from GCL (Fig. 9*A*) with the adaptive threshold from AL in the cortical layer (*B*).

Analysis of this model in Figure 9*C* demonstrates that the combination of these two mechanisms results in a model that is extremely robust to contrast, while organizing much more smoothly and consistently at any contrast than the other models. All maps at all visible contrasts achieve $\pi$ pinwheel density, as shown by the map metric value. As an illustration, the data for 40 GCAL simulations at 100% contrast have been replotted in Figure 3*D*, where the cluster of red circles for the GCAL model clearly fits into the range of pinwheel density values seen in ferret maps. All FFT plots are ring shaped and comparable to the biological maps, and none of the maps suffer from the artifacts identified in the AL and GCL models. Selectivity and map stability are achieved rapidly once sufficient contrast is available, and remain constant across all contrasts.

The maps at the low-, medium-, and high-contrast points are qualitatively indistinguishable; all afferent weights develop properly without becoming overly selective at the expense of map quality. Combining gain-control and homeostatic adaptation achieves a trade-off between selectivity and map quality that results in highly realistic orientation preference and selectivity maps that are extremely robust to variations in contrast.

Evidently, homeostatic adaptation complements gain control by adjusting the postsynaptic target activity of each V1 unit, ensuring the long-term smooth development across the map over many visual inputs and thus preventing any particular region of the map from being poorly activated and thus lagging behind in the self-organization process. Homeostasis helps ensure consistent map development despite fluctuations in the input statistics across multiple presentations, by retaining a weighted history of previous activation.

Contrast-gain control in turn complements homeostatic adaptation by compressing the dynamic range of each individual

input, regulating the presynaptic input between the ON/OFF sheets and the cortical layer, and ensuring the rate of Hebbian learning in the afferent connections matches the rate of development of lateral cortical connectivity.

Even if the long-term average presynaptic activity in the ON/OFF sheets is appropriate for normal Hebbian learning in the afferent connections, the high dynamic range within a single visual input would continue to disrupt the afferent connectivity on time scales too rapid for the homeostatic threshold to compensate. Contrast-gain control assists the action of homeostatic adaptation over long time scales, normalizing the dynamic range of activity of individual presentations and allowing the activity threshold to respond to long-term trends in the input and not to the transient variance within individual presentations.

These two mechanisms perform two different operations on different time scales that independently increase the robustness of development to input contrast. Together these two operations complement each others' operation to adjust the average activity and dynamic range of the presynaptic and postsynaptic neural activity to ensure robust map development progresses smoothly and regularly on both long and short time scales.

### Stable and robust development using realistic inputs in the GCAL model

The previous sections looked only at robustness to image contrast, but biological maps appear to be robust against not only the strength, but also the frequency and type of patterns, with similar maps developing with and without eyes open and with and without dark rearing. We now examine how the GCAL model previously tested with the abstract Gaussian input patterns can be used for other types of input patterns, without changing any of the model parameters.

When an animal opens its eyes, the driving input to the LGN and V1 changes from spontaneously generated patterns to natural visual input (Cragg, 1975; Beckmann and Albus, 1982; Tavazoie and Reid, 2000; Hooks and Chen, 2006; Huberman et al., 2006). As described above, despite this change, orientation maps measured at eye opening retain their orientation preferences, and therefore mature into a final adult map that is similar to the map measured at eye opening. We can model pre-eye-opening development in the GCAL model using input image patterns that resemble intrinsically generated activity [loosely modeled after retinal waves (Wong, 1999) and represented as noisy disk-shaped patterns randomly placed on the retinal photoreceptors at each iteration (Bednar and Miikkulainen, 2004)]. After eye opening, development can be modeled using retinal inputs extracted from a data set of natural images, from Shouval et al. (1996).

Figure 10 shows results from the GCAL model, which includes both gain control in the ON/OFF channels and homeostatic adaptation, compared with experimental results from chronic optical imaging in ferret. In the model, when the input type changes from noisy disks to natural images (at iteration 6000), there is no decrease in the map orientation stability index (Fig. 10*E*), even though selectivity continues to increase. The initial map structure generated before model "eye opening" is maintained after eye opening, despite the change in the nature of the driving activity. Note that no model parameters have been changed from the previous simulations with Gaussian input stimuli; the model is robust even to extreme changes in the input images without the need for any compensatory changes in parameters.

It would not be possible to use these more realistic inputs in the L or AL models without a significant retuning of the parameters. Without contrast-gain control, individual image-patch presentations with high contrast will disrupt the afferent connectivity from the LGN to the V1 sheets, as in Figures 6*F* and 7*F*. The GCL model is more robust to contrast changes, yet, as shown in Figure 8*F*, development of individual neurons is not robust, and thus only the GCAL model allows all neurons to develop robustly across a wide range of inputs.

The stability of ferret map development is measured in Figures 2, *A* and *B*, and 10*D*. We cannot directly compare iterations in the model with days in the experimental data, because the precise date of any initial starting point in the ferret is unclear. However, we can qualitatively compare the overall pattern of stable development for the experiment over the first 10,000 iterations, which covers the period when selectivity and stability rise before saturating in the model. In all the measured experimental cases, selectivity values increase in conjunction with stability to their maximum values over the course of development, which is also true of the simulated data. In some ferrets in which the eyes are surgically opened before the time of natural eye opening, very faint maps were already seen and were already more similar to the final map than to the control condition. Similarly, model maps that have begun to develop before eye opening are similar to the final map (Fig. 10*E*). These results are similar to the average development for all eight ferrets shown in Figure 2*B* and are consistent with binocular deprivation experiments in cat (Crair et al., 1998).

Of course, map stability could be achieved trivially by decreasing the ability of neurons to adapt once an orientation map is initially formed before eye opening (for example, by forcibly decreasing the Hebbian learning rate, as in many models). However, as shown in various experimental studies, maps continue to adapt and mature after eye opening, and ultimately come to reflect some of the underlying statistics of the natural visual input. For example, Figure 11*A* (reprinted from Tanaka et al., 2009) shows experimentally measured biological maps with and without continuous exposure to one particular orientation, using a goggle-rearing oriented-blurring paradigm. These experimental results suggest that when exposed to one particular orientation after eye opening, orientation maps in V1 reorganize by expanding the domains in the map that maximally respond to the goggle-reared orientation, while reducing the areas responding to the unexposed orientations. We can test this property in the GCAL model by using post-eye-opening image data sets with different orientation statistics. Such results are illustrated in Figure 11*D*, where the map is developed using the same set of natural image patches used in Figure 10 after they have each been convolved with a vertically elongated Gaussian kernel, approximating the anisotropy introduced by goggle rearing. The resulting orientation map also contains a much higher proportion of vertical-preferring neurons. The differences between the model map before and after eye opening, and as the map develops, show that, as in the experimental data, domains of the model V1 preferring the predominant orientation have expanded as the map has matured. The GCAL model is therefore robust and stable, but also able to adapt to changes in the input statistics.

Finally, because the mechanisms added to make the GCAL model were not arbitrary, but were chosen based on a wide range of well-established experimental work, it is possible to make specific links between the behavior of the GCAL model's neurons and experimental data. For example, contrast-gain control mechanisms have been proposed to be involved in contrast-invariant tuning, whereby the tuning curves for V1 neurons retain the same width across contrasts (Sclar and Freeman, 1982; Skottun et al.,

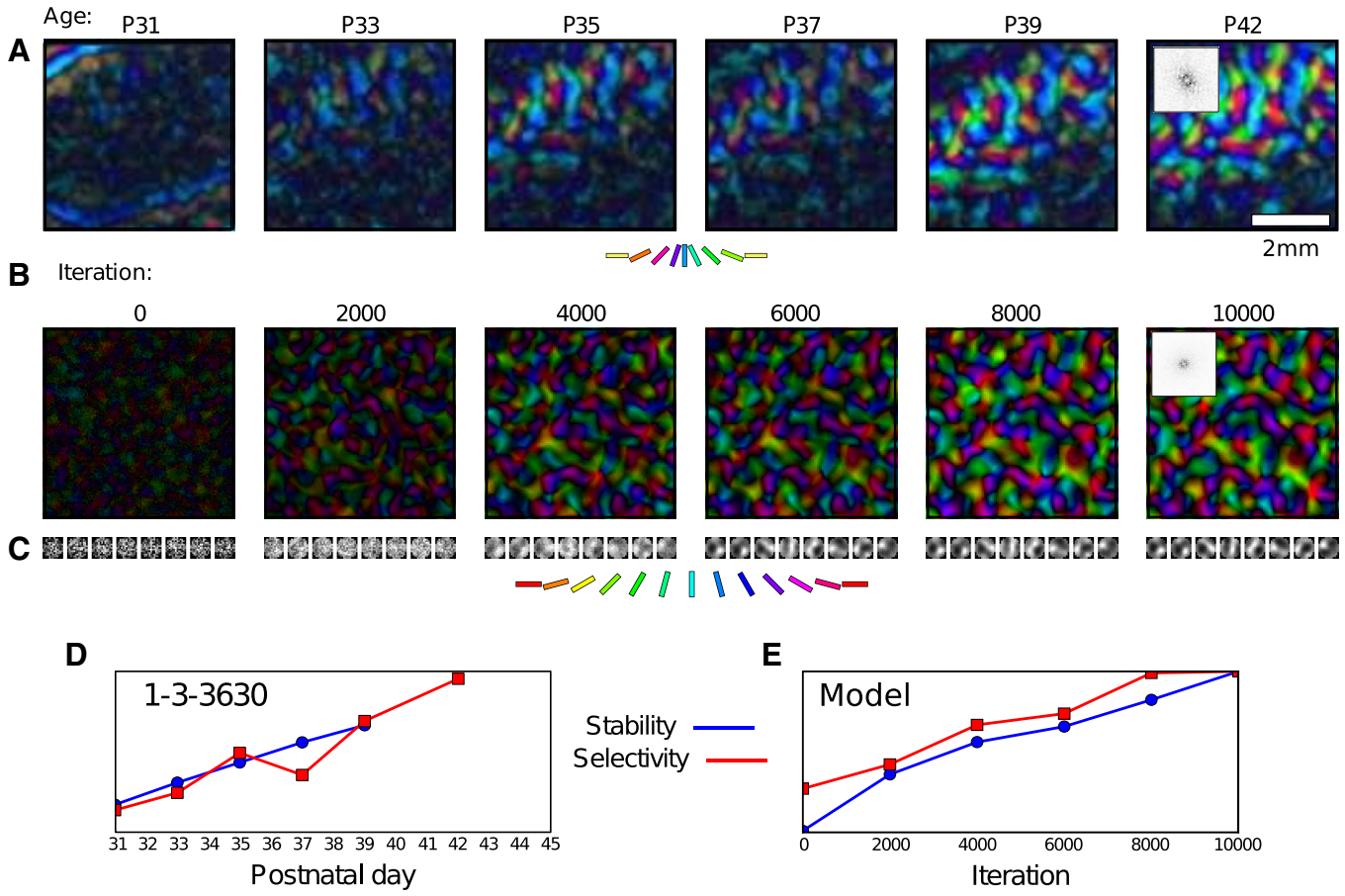


**Figure 10.** Stable map development before and after eye opening. ***A***, Polar orientation maps recorded using chronic optical imaging at different ages in one ferret [Fig. 1*A*, ferret 1-3-3630; reprinted from the study by Chapman et al. (1996)]. Scale bar, 2 mm. ***B***, Simulated GCAL polar orientation maps. Noisy disk patterns drive the map development until 6000 iterations, after which natural images are presented to the model retina. ***A***, ***B***, Both the ferret and model map have a ring-shaped FFT (inset in the final map plot of each). ***C***, Afferent connections from the ON sheet to V1 are shown for an arbitrary set of model V1 units throughout development, to illustrate how neurons become more selective over time. ***D***, Orientation stability indices (Eq. 1) across development for the ferret from ***A***, replotted from Figure 2*B* for comparison. ***E***, SI for the simulation shown in ***B***. As selectivity develops in both the ferret and GCAL, the map smoothly increases in stability, indicating a highly stable process of development. Stability can also be seen by tracking individual features of the orientation map across time, again for the ferret and for GCAL. These results show that the mechanisms in GCAL are sufficient to account for the observed levels of stability in ferret map development.

1987). Figure 12 shows that the GCAL model's self-organized orientation-selective neurons have achieved robust contrast-invariant tuning. This property arises in the model because the contrast-gain control in the ON/OFF channels increases the responses to low-contrast relative to high-contrast stimuli. The recurrent lateral interactions in V1 then ensure that differences in the responses to preferred and nonpreferred orientations are amplified (Antolík, 2010). Responses to nonpreferred orientations will therefore always be smaller than those to preferred orientations. Similar methods of achieving contrast-invariant orientation tuning have been described in previous nondevelopmental models (Geisler and Albrecht, 1997; Anderson et al., 2000; Carandini et al., 2002; Finn et al., 2007), but to our knowledge GCAL is the first model to show how contrast-invariant tuning using this method can emerge robustly over development for a full map of V1 neurons.

## Discussion

We have shown that the integration of a small number of simple, well-known, biologically realistic mechanisms is sufficient to reproduce stable and robust map development. Contrast-gain control in the early visual pathway and homeostatic maintenance of activity levels have the effect of regulating both the presynaptic activity and postsynaptic activity of V1 neurons. This regulation can ensure that throughout development, activity levels, and therefore the rate of Hebbian learning, remain stable. In this way, receptive fields of individual neurons can approach a final organized structure without being overwritten many times, thus achieving the stability and robustness observed in experimental studies. These mechanisms also potentially underlie contrast-invariant orientation tuning of single neurons.

Other activity-dependent models such as the elastic net have previously demonstrated stable development for orientation and for ocular dominance under certain conditions (Keil and Wolf, 2011; Keil et al., 2010). However, these models are formulated at a more abstract level not suitable for identifying important mechanisms like homeostasis and gain control. In particular, they do not simulate responses to individual input patterns such as natural images or retinal waves, and thus cannot address robustness with respect to changes in the input pattern types.

In the real biological system, the differences between input types may be more (or less) extreme than those presented here, leading to two possibilities. First, if the differences in the types of inputs that drive the real visual system are more extreme, further adaptation mechanisms may be required. For instance, if the den-

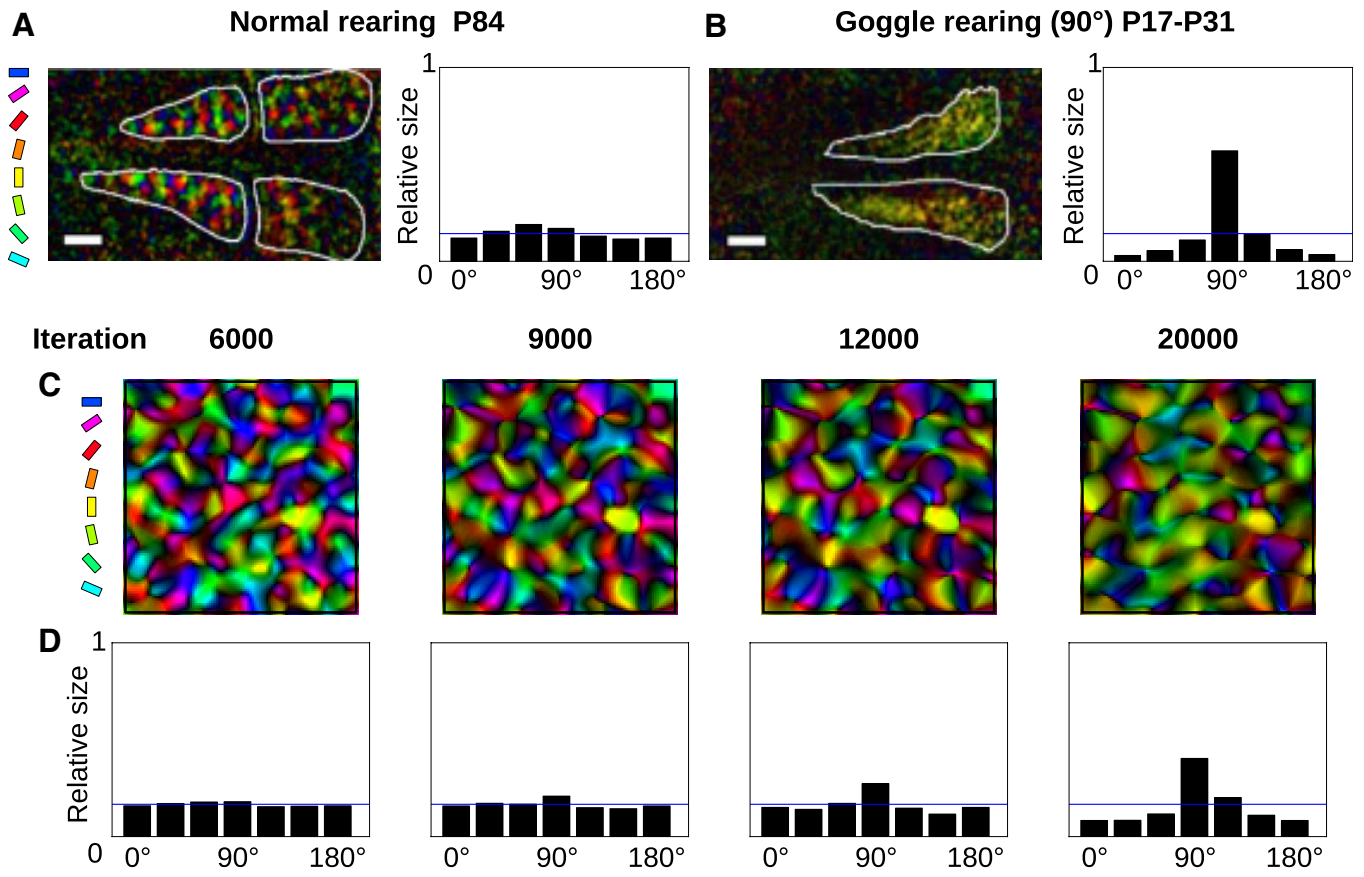


**Figure 11.** Experimental and GCAL model orientation maps developed in orientation-biased environments. ***A***, Orientation map measured in a normally reared kitten at postnatal day 84 and orientation histogram. The color and brightness indicate the preferred orientation and the orientation magnitude, respectively. The color code for preferred orientations is shown to the side. Scale bars: 2 mm. ***B***, Orientation map measured in a kitten at postnatal day 42 after 13 d of goggle rearing with vertical lines. Orientation histogram now shows a strong bias toward vertical orientations. Reprinted from Tanaka et al. (2009). ***C***, Orientation preference maps during development from a GCAL simulation driven by noisy disk patterns until 6000 iterations, after which natural images that have been anisotropically blurred vertically are presented to the model retinal photoreceptors. ***D***, Histograms show an expansion of yellow (vertically preferring) regions in the orientation after eye opening, reflecting the statistics of the natural image input and reproducing the results observed in biological map development.

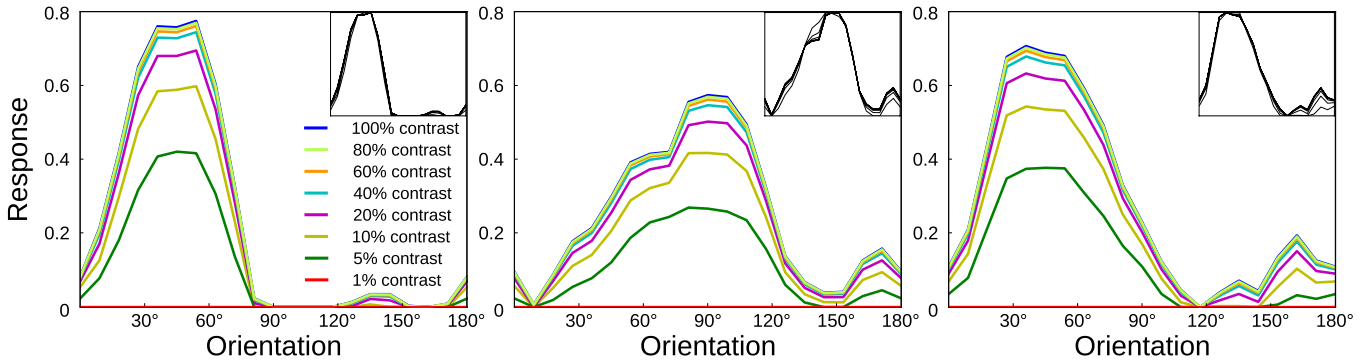


**Figure 12.** GCAL model: contrast-invariant orientation tuning. Example orientation tuning curves for three representative model neurons, measured at the indicated contrasts. Orientation tuning width remains constant despite changes in contrast (except for small deviations at the lowest contrasts), as confirmed by the normalized tuning curves (inset).

sity of edges in visual scenes varies dramatically across different rearing environments (e.g., between desert and jungle environments), it may be necessary for the adaptation mechanism to take both presynaptic and postsynaptic activity into account together within each cell. We have implemented such a mechanism in our modeling, but found that it was not necessary to handle the range of input types considered here, while being less obvious how to implement locally in V1 or LGN neurons. Alternatively, if real input differences are less extreme, the mechanisms introduced here may not both be required, although this is an unlikely possibility given that there is substantial experimental evidence for both of the simple mechanisms involved. The observation that these mechanisms are also consistent with many other single-neuron properties, such as contrast-invariant orientation selectivity and automatic maintenance of average activity, suggests that the visual system has achieved many different properties using similar underlying mechanisms.

The ability of neurons to learn and adapt requires changes in synaptic connectivity, yet it is also essential that neurons are neither excessively active or silenced. Both excessive activity and

silence would lead to a loss of information at each level of the neural pathway (Turrigiano and Nelson, 2004), and so homeostatic adaptation not only contributes to stable and robust map development, but also helps to maintain the transfer of visual information. More complex homeostatic mechanisms may also stabilize map development, for example, by regulating the distribution of activity rather than just the target average value (Triesch, 2005) or by operating on the synaptic weights directly via a process such as synaptic scaling (Tetzlaff et al., 2011). In any case, the idea that neurons use some measure of their average activity over time to trigger homeostasis and regulate their responses is widely accepted, and the specific simple homeostatic mechanism used in the GCAL model is sufficient to demonstrate how such regulation improves stability and robustness.

In the visual pathway, it is possible that both homeostatic adaptation and contrast-gain control are present at every stage of processing. In the GCAL model, we have chosen to apply the explicit contrast-gain control mechanism only to the ON/OFF sheets and homeostatic adaptation only to the V1 units. It would have been possible to add both of these mechanisms to all of the stages of the model, but this would have made the effect of each mechanism harder to analyze and interpret, and we do not believe such a change would affect our results significantly.

The GCAL model depends heavily on lateral interactions. Lateral interactions in V1 arise because of lateral recurrent connections between neurons in V1 (Gilbert et al., 1990; Bosking et al., 1997; primarily in layer 2/3). In other simple self-organizing map algorithms, these specific lateral interactions are replaced by a more convenient and computationally efficient but highly abstract mechanism. For example, in the self-organizing map model (Obermayer et al., 1990; Kohonen 1982; Farley et al., 2007), the algorithm finds the maximally active neuron in the cortex based on the distance between the input pattern and each neuron's set of afferent weights. Afferent weights are then adapted only in a specified circular (Gaussian) neighborhood around the maximally active unit. There are therefore no specific lateral interaction strengths in the SOM algorithm, weight changes depend only on the position of the maximally active unit, and the cortical activity does not depend on the contrast of the input stimulus. As shown in this study, we believe that it is important to include specific lateral interaction strengths in mechanistic models of V1 to understand the constraints on the real system.

Although lateral interactions are important for the operation of the GCAL model, the architecture is deliberately simplified and does not reflect the detailed anatomy of cortical connectivity. In animals, lateral interactions involve a complex circuit based on long-range excitation and disynaptic inhibition, with different effects at different contrasts (Hirsch and Gilbert, 1991; Weliky et al., 1995). A model related to GCAL demonstrates how this more elaborate circuit could give similar results (Law, 2009), but requires many more parameters and more complicated analysis methods. Similarly, a related model shows how simulating multiple V1 laminae can explain both simple and complex cells in V1, allowing neurons with random phase preferences to develop in the simple cells and realistic orientation maps in the complex cells (Antolík and Bednar, 2011). These other models improve on the realism of the GCAL model, but because of their greater complexity and the larger associated parameter spaces, they are more difficult to analyze and understand, and so the GCAL model is more useful for studying phenomena that do not require those additional circuit elements.

Apart from the lateral connectivity, another feature of the SOM and other algorithms that is not shared by the GCAL model is that each SOM V1 neuron begins with initial connections to the whole of the retina. This starting point requires that both retinotopic and orientation-selective receptive fields develop simultaneously. Achieving smooth development of both these features in a computational model requires several global ad hoc mechanisms. For example, the processes of lateral neighborhood decrease and learning rate decrease in the SOM algorithm help ensure that self-organization does not fall into a local minimum where the mapping is not retinotopic (i.e., they provide an "annealing" of the map). These ad hoc mechanisms themselves result in model orientation map instability, such as a decrease in the size of orientation domains and a large reorganization of the map over time. However, real V1 maps are arguably formed sequentially, with a coarse retinotopic map already in place before the orientation map (for review, see Huberman et al., 2008). In the model algorithm presented here, there is an initial coarse retinotopic map in V1, which avoids the need for global ad hoc mechanisms to achieve smooth map development.

The removal of ad hoc mechanisms and addition of mechanisms explicitly regulating activity levels makes the GCAL model easier to understand with numerical simulations (as we have done here), and we expect the same fundamental theoretical principles of operation to apply to the GCAL model as for other self-organizing map models (von der Malsburg, 1973; Kohonen, 1982). These properties have been analyzed extensively in previous work with these related models. In particular, the GCAL model achieves good coverage and continuity in its representation of the input feature space (Swindale et al., 2000) by forming sparse, decorrelated activity bubbles in response to each input, just as in the LISSOM model (Miikkulainen et al., 2005). This process has been shown to fold a high-dimensional feature space onto a two-dimensional surface, as a discretized approximation of the principal surfaces of the input (Ritter et al., 1992), and is a form of dimensionality reduction (Durbin and Mitchison, 1990). We argue that the GCAL model should be seen as a mechanistic implementation of the well-established principles of self-organization, allowing them to proceed robustly for realistic inputs of variable strengths and types.

Apart from the specific models introduced, we showed how pinwheel density can be used to assess the quality of simulated orientation maps. Of course, pinwheel density is just one of many factors that could be measured and compared between maps, but we argue that it is a useful metric for several reasons. First, there is a unique, cross-species reference value, which allows objective decisions about map quality. Having a pinwheel density close to $\pi$ is a necessary, if not always sufficient, criterion for a map to be considered biological. Second, reliable high performance on this measure requires that the maps be smooth (thus implicitly incorporating alternative metrics like LHI; Nauhaus et al., 2008) and periodic (thus implicitly incorporating the ringness in the Fourier power spectrum). Third, the reference value of $\pi$ was measured empirically through careful analysis of the raw imaging data that ensured that $\pi$ pinwheel density is a genuine, structural property of the underlying orientation map organization, not simply an artifact of the particular filtering process used. This reference value can thus be compared directly against values from simulated maps, which do not need filtering or other preprocessing that requires human judgment.

The proposed metric could be extended to include additional criteria for map quality, but those of which we are aware have

significant issues that make them unsuitable for characterizing model maps. For instance, systematic bias in the orientation histograms has been quantified for biological maps (Coppola et al., 1998; Müller et al., 2000). However, our results and those of Tanaka et al. (2009) in Figure 11 show that these biases depend crucially on the visual input statistics in both models and animals, rather than revealing inherent properties of the maps. Because the visual statistics have not yet been measured for laboratory environments, and are likely to vary significantly between labs, the levels of map bias found in a particular study are not a suitable reference value for a general model. Similarly, Müller et al. (2000) and Kaschube et al. (2010) show that biological maps have slightly different average distances between pinwheels of the same polarity (with orientation increasing clockwise or counterclockwise) or opposite polarities, which is also true of the high-quality maps from the GCAL model (data not shown). However, there is no clear reference value for this difference, which varies across maps and species. Calculating a meaningful value for polarity biases also requires precise localization of independent pinwheels, which is difficult to automate for any maps other than high-quality maps like those from GCAL. Thus, the pinwheel polarity distribution gives relatively little information for distinguishing between maps of varying quality. However, each of these analyses can be useful alongside the proposed metric, and whenever reference values that hold across laboratories and across species become available, the metric can be extended to include these.

With recent advances in calcium imaging (e.g., two-photon imaging), we anticipate novel, high-resolution experimental results concerning the time course of map development that we can relate to the GCAL model. Although rodent species are most commonly used in existing calcium-imaging studies, these techniques are beginning to yield new results for species that develop orientation maps, such as the macaque monkey (Nauhaus et al., 2012). Using genetically encoded calcium indicators, it has also become possible to use calcium imaging for chronic recordings (Lütcke et al., 2013). Chronic calcium recordings of orientation map development would complement the existing optical imaging results, offering insights into the process of map formation at cellular resolution.

The GCAL model may also be applicable for understanding cortical development in species that do not form continuous orientation maps, such as rodents Ohki et al. (2005). The homeostatic adaptation and contrast-gain control mechanisms remain applicable even with salt-and-pepper organization of preferences, and it has been shown that this type of organization can emerge in models related to GCAL (Law, 2009). It may be possible to adapt the GCAL model to account for other developmental phenomena involving orientation-selective cells, such as the convergence of ipsilateral eye and contralateral eye orientation preference in binocular cells of mice (Wang et al., 2010). The GCAL model's robust mechanisms should allow the transition between intrinsic activity and visually driven activity to be simulated with little or no retuning of model parameters.

Finally, the realistic features of the GCAL model make this model a useful starting point for investigating properties of map development using a wide range of inputs. Previous state-of-the-art high-dimensional models of topographic map development, such as the LISSOM model (Miikkulainen et al., 2005), required a complete retuning of a large number of parameters depending on the input patterns used. However, there is a relatively small number of free parameters of the GCAL model (Hebbian learning rate, target V1 activity, strength of RGC/LGN inhibition, and V1 lateral interaction strengths). Moreover, these values hold for a wide range of inputs without any tuning.

Fundamentally, we believe that contrast-gain control and homeostatic adaptation are important basic principles underlying topographic map development, even though they have primarily been considered only at the single-neuron and small-network levels in previous work.

## Notes

Supplemental material for this article is available at www.topographica.org. The models are implemented in the Topographica simulator, freely available at www.topographica.org. This material has not been peer reviewed.

## References


Alitto HJ, Usrey WM (2004) Influence of contrast on orientation and temporal frequency tuning in ferret primary visual cortex. J Neurophysiol 91:2797–2808. CrossRef Medline

Alitto HJ, Usrey WM (2008) Origin and dynamics of extraclassical suppression in the lateral geniculate nucleus of the macaque monkey. Neuron 57:135–146. CrossRef Medline

Anderson JS, Carandini M, Ferster D (2000) Orientation tuning of input conductance, excitation, and inhibition in cat primary visual cortex. J Neurophysiol 84:909–926. Medline

Antolík J (2010) Unified developmental model of maps, complex cells and surround modulation in the primary visual cortex. PhD thesis, School of Informatics, The University of Edinburgh, Edinburgh, UK.

Antolík J, Bednar JA (2011) Development of maps of simple and complex cells in the primary visual cortex. Front Comput Neurosci 5:17. Medline

Baccus SA, Meister M (2002) Fast and slow contrast adaptation in retinal circuitry. Neuron 36:909–919. CrossRef Medline

Barrow HG, Bray AJ, Budd JM (1996) A self-organizing model of 'color blob' formation. Neural Comput 8:1427–1448. CrossRef Medline

Beckmann R, Albus K (1982) The geniculocortical system in the early postnatal kitten: an electrophysiological investigation. Exp Brain Res 47:49–56. Medline

Bednar JA, Miikkulainen R (2004) Prenatal and postnatal development of laterally connected orientation maps. Neurocomputing 58–60:985–992.

Bienenstock EL, Cooper LN, Munro PW (1982) Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. J Neurosci 2:32–48. Medline

Blakemore C, Cooper GF (1970) Development of the brain depends on the visual environment. Nature 228:477–478. CrossRef Medline

Blakemore C, Van Sluyters RC (1975) Innate and environmental factors in the development of the kitten's visual cortex. J Physiol 248:663–716. Medline

Blasdel GG (1992a) Differential imaging of ocular dominance and orientation selectivity in monkey striate cortex. J Neurosci 12:3115–3138. Medline

Blasdel GG (1992b) Orientation selectivity, preference, and continuity in monkey striate cortex. J Neurosci 12:3139–3161. Medline

Blasdel GG, Salama G (1986) Voltage-sensitive dyes reveal a modular organization in monkey striate cortex. Nature 321:579–585. CrossRef Medline

Bonin V, Mante V, Carandini M (2005) The suppressive field of neurons in lateral geniculate nucleus. J Neurosci 25:10844–10856. CrossRef Medline

Bosking WH, Zhang Y, Schofield B, Fitzpatrick D (1997) Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex. J Neurosci 17:2112–2127. Medline

Burger T, Lang EW (1999) An incremental Hebbian learning model of the primary visual cortex with lateral plasticity and real input patterns. Z Naturforschung C 54:128–140. Medline

Burger T, Lang EW (2001) Self-organization of local cortical circuits and cortical orientation maps: a nonlinear Hebbian model of the visual cortex with adaptive lateral couplings. Z Naturforsch C 56:464–478. Medline

Carandini M, Heeger DJ (2012) Normalization as a canonical neural computation. Nat Rev Neurosci 13:51–62. CrossRef Medline

Carandini M, Heeger DJ, Senn W (2002) A synaptic explanation of suppression in visual cortex. J Neurosci 22:10053–10065. Medline

Chapman B, Bonhoeffer T (1998) Overrepresentation of horizontal and vertical orientation preferences in developing ferret area 17. Proc Natl Acad Sci U S A 95:2609–2614. CrossRef Medline

Chapman B, Gödecke I (2000) Cortical cell orientation selectivity fails to develop in the absence of ON-center retinal ganglion cell activity. J Neurosci 20:1922–1930. Medline

Chapman B, Stryker MP (1993) Development of orientation selectivity in ferret visual cortex and effects of deprivation. J Neurosci 13:5251–5262. Medline

Chapman B, Stryker MP, Bonhoeffer T (1996) Development of orientation preference maps in ferret primary visual cortex. J Neurosci 16:6443–6453. Medline

Coppola DM, White LE, Fitzpatrick D, Purves D (1998) Unequal representation of cardinal and oblique contours in ferret visual cortex. Proc Natl Acad Sci U S A 95:2621–2623. CrossRef Medline

Cragg BG (1975) The development of synapses in the visual system of the cat. J Comp Neurol 160:147–166. CrossRef Medline

Crair MC, Gillespie DC, Stryker MP (1998) The role of visual experience in the development of columns in cat visual cortex. Science 279:566–570. CrossRef Medline

Daoudal G, Debanne D (2003) Long-term plasticity of intrinsic excitability: learning rules and mechanisms. Learn Mem 10:456–465. CrossRef Medline

Davis GW, Bezprozvanny I (2001) Maintaining the stability of neural function: a homeostatic hypothesis. Annu Rev Physiol 63:847–869. CrossRef Medline

Derrington AM, Lennie P (1984) Spatial and temporal contrast sensitivities of neurones in lateral geniculate nucleus of macaque. J Physiol 357:219–240. Medline

Durbin R, Mitchison G (1990) A dimension reduction framework for understanding cortical maps. Nature 343:644–647. CrossRef Medline

Erwin E, Obermayer K, Schulten K (1995) Models of orientation and ocular dominance columns in the visual cortex: a critical comparison. Neural Comput 7:425–468. CrossRef Medline

Farley BJ, Yu H, Jin DZ, Sur M (2007) Alteration of visual input results in a coordinated reorganization of multiple visual cortex maps. J Neurosci 27:10299–10310. CrossRef Medline

Felisberti F, Derrington AM (1999) Long-range interactions modulate the contrast gain in the lateral geniculate nucleus of cats. Vis Neurosci 16:943–956. Medline

Finn IM, Priebe NJ, Ferster D (2007) The emergence of contrast-invariant orientation tuning in simple cells of cat visual cortex. Neuron 54:137–152. CrossRef Medline

Geisler WS, Albrecht DG (1997) Visual cortex neurons in monkeys and cats: detection, discrimination, and identification. Vis Neurosci 14:897–919. CrossRef Medline

Gilbert CD, Hirsch JA, Wiesel TN (1990) Lateral interactions in visual cortex. Cold Spring Harb Symp Quant Biol 55:663–677. CrossRef Medline

Gödecke I, Kim DS, Bonhoeffer T, Singer W (1997) Development of orientation preference maps in area 18 of kitten visual cortex. Eur J Neurosci 9:1754–1762. CrossRef Medline

Goodhill GJ (2007) Contributions of theoretical modeling to the understanding of neural map development. Neuron 56:301–311. CrossRef Medline

Grubb MS, Burrone J (2010) Activity-dependent relocation of the axon initial segment fine-tunes neuronal excitability. Nature 465:1070–1074. CrossRef Medline

Hirsch JA, Gilbert CD (1991) Synaptic physiology of horizontal connections in the cat's visual cortex. J Neurosci 11:1800–1809. Medline

Hooks BM, Chen C (2006) Distinct roles for spontaneous and visual activity in remodeling of the retinogeniculate synapse. Neuron 52:281–291. CrossRef Medline

Huberman AD, Speer CM, Chapman B (2006) Spontaneous retinal activity mediates development of ocular dominance columns and binocular receptive fields in V1. Neuron 52:247–254. CrossRef Medline

Huberman AD, Feller MB, Chapman B (2008) Mechanisms underlying development of visual maps and receptive fields. Annu Rev Neurosci 31:479–509. CrossRef Medline

Kaschube M, Schnabel M, Löwel S, Coppola DM, White LE, Wolf F (2010) Universality in the evolution of orientation columns in the visual cortex. Science 330:1113–1116. CrossRef Medline

Keil W, Wolf F (2011) Coverage, continuity, and visual cortical architecture. Neural Syst Circuits 1:17. CrossRef Medline

Keil W, Schmidt KF, Löwel S, Kaschube M (2010) Reorganization of columnar architecture in the growing visual cortex. Proc Natl Acad Sci U S A 107:12293–12298. CrossRef Medline

Keil W, Kaschube M, Schnabel M, Kisvarday ZF, Lwel S, Coppola DM, White LE, Wolf F (2012) Response to comment on universality in the evolution of orientation columns in the visual cortex. Science 336:413. CrossRef

Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69. CrossRef

Kuba H, Oichi Y, Ohmori H (2010) Presynaptic activity regulates Na(+) channel distribution at the axon initial segment. Nature 465:1075–1078. CrossRef Medline

Law JS (2009) Modeling the development of organization for orientation preference in primary visual cortex. PhD thesis, School of Informatics, The University of Edinburgh, Edinburgh, UK.

Linsker R (1986) From basic network principles to neural architecture: Emergence of orientation columns. Proc Natl Acad Sci U S A 83:8779–8783. CrossRef Medline

Löwel S, Schmidt KE, Kim DS, Wolf F, Hoffsümmer F, Singer W, Bonhoeffer T (1998) The layout of orientation and ocular dominance domains in area 17 of strabismic cats. Eur J Neurosci 10:2629–2643. CrossRef Medline

Lütcke H, Margolis DJ, Helmchen F (2013) Steady or changing? long-term monitoring of neuronal population activity. Trends Neurosci 36:375–384. CrossRef Medline

Miikkulainen R, Bednar J, Choe Y, Sirosh J (2005) Computational maps in the visual cortex. New York: Springer.

Miller KD (1994) A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between ON- and OFF-center inputs. J Neurosci 14:409–441. Medline

Müller T, Stetter M, Hübener M, Sengpiel F, Bonhoeffer T, Gödecke I, Chapman B, Löwel S, Obermayer K (2000) An analysis of orientation and ocular dominance patterns in the visual cortex of cats and ferrets. Neural Comput 12:2573–2595. CrossRef Medline

Nauhaus I, Benucci A, Carandini M, Ringach DL (2008) Neuronal selectivity and local map structure in visual cortex. Neuron 57:673–679. CrossRef Medline

Nauhaus I, Nielsen KJ, Disney AA, Callaway EM (2012) Orthogonal micro-organization of orientation and spatial frequency in primate primary visual cortex. Nat Neurosci 15:1683–1690. CrossRef Medline

Obermayer K, Ritter H, Schulten K (1990) A principle for the formation of the spatial structure of cortical feature maps. Proc Natl Acad Sci U S A 87:8345–8349. CrossRef Medline

Ohki K, Chung S, Ch'ng YH, Kara P, Reid RC (2005) Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex. Nature 433:597–603. CrossRef Medline

Ritter H, Martinetz T, and Schulten KJ (1992) Neural computation and self-organizing maps: an introduction. Reading, MA: Addison-Wesley.

Rochester N, Holland JH, Haibt LH, Duda WL (1956) Tests on a cell assembly theory of the action of the brain, using a large digital computer. IRE Trans Inform Theory 2:80–93. CrossRef

Schulz DJ (2006) Plasticity and stability in neuronal output via changes in intrinsic excitability: it's what's inside that counts. J Exp Biol 209:4821–4827. CrossRef Medline

Schummers J, Mariño J, Sur M (2004) Local networks in visual cortex and their influence on neuronal responses and dynamics. J Physiol Paris 98:429–441. CrossRef Medline

Sclar G (1987) Expression of "retinal" contrast gain control by neurons of the cat's lateral geniculate nucleus. Exp Brain Res 66:589–596. CrossRef Medline

Sclar G, Freeman RD (1982) Orientation selectivity in the cat's striate cortex is invariant with stimulus contrast. Exp Brain Res 46:457–461. Medline

Sengpiel F, Stawinski P, Bonhoeffer T (1999) Influence of experience on orientation maps in cat visual cortex. Nat Neurosci 2:727–732. CrossRef Medline

Shapley RM, Victor JD (1978) The effect of contrast on the transfer properties of cat retinal ganglion cells. J Physiol 285:275–298. Medline

Shouval H, Intrator N, Law CC, Cooper LN (1996) Effect of binocular cor-

tical misalignment on ocular dominance and orientation selectivity. Neural Comput 8:1021–1040. CrossRef Medline

Sirosh J, Miikkulainen R (1994) Cooperative self-organization of afferent and lateral connections in cortical maps. Biol Cybern 71:66–78.

Sirosh J, Miikkulainen R (1997) Topographic receptive fields and patterned lateral interaction in a self-organizing model of the primary visual cortex. Neural Comput 9:577–594. CrossRef Medline

Skottun BC, Bradley A, Sclar G, Ohzawa I, Freeman RD (1987) The effects of contrast on visual orientation and spatial frequency discrimination: a comparison of single cells and behavior. J Neurophysiol 57:773–786. Medline

Sullivan TJ, de Sa VR (2006) Homeostatic synaptic scaling in self-organizing maps. Neural Netw 19:734–743. CrossRef Medline

Swindale NV (1996) The development of topography in the visual cortex: a review of models. Network 7:161–247. CrossRef Medline

Swindale NV, Shoham D, Grinvald A, Bonhoeffer T, Hübener M (2000) Visual cortex maps are optimized for uniform coverage. Nat Neurosci 3:822–826. CrossRef Medline

Tanaka S, Ribot J, Imamura K, Tani T (2006) Orientation-restricted continuous visual exposure induces marked reorganization of orientation maps in early life. Neuroimage 30:462–477. CrossRef Medline

Tanaka S, Tani T, Ribot J, O'Hashi K, Imamura K (2009) A postnatal critical period for orientation plasticity in the cat visual cortex. PLoS ONE 4:e5380. CrossRef Medline

Tavazoie SF, Reid RC (2000) Diverse receptive fields in the lateral geniculate nucleus during thalamocortical development. Nat Neurosci 3:608–616. CrossRef Medline

Tetzlaff C, Kolodziejski C, Timme M, Wörgötter F (2011) Synaptic scaling in combination with many generic plasticity mechanisms stabilizes circuit connectivity. Front Comput Neurosci 5:47. Medline

Triesch J (2005) A gradient rule for the plasticity of a neuron's intrinsic excitability. In: International Conference on Artificial Neural Networks (ICANN) 2005, Lecture Notes in Computer Science, Vol 3696 (Duch W, et al., eds.), pp 65–70 Berlin: Springer-Verlag.

Truchard AM, Ohzawa I, Freeman RD (2000) Contrast gain control in the visual cortex: monocular versus binocular mechanisms. J Neurosci 20:3017–3032. Medline

Turrigiano GG (1999) Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same. Trends Neurosci 22:221–227. CrossRef Medline

Turrigiano GG, Nelson SB (2004) Homeostatic plasticity in the developing nervous system. Nat Rev Neurosci 5:97–107. CrossRef Medline

von der Malsburg C (1973) Self-organization of orientation-sensitive cells in the striate cortex. Kybernetik 14:85–100. CrossRef Medline

Wang BS, Sarnaik R, Cang J (2010) Critical period plasticity matches binocular orientation preference in the visual cortex. Neuron 65:246–256. CrossRef Medline

Weliky M, Kandler K, Fitzpatrick D, Katz LC (1995) Patterns of excitation and inhibition evoked by horizontal connections in visual cortex share a common relationship to orientation columns. Neuron 15:541–552. CrossRef Medline

White LE, Fitzpatrick D (2007) Vision and cortical map development. Neuron 56:327–338. CrossRef Medline

White LE, Coppola DM, Fitzpatrick D (2001) The contribution of sensory experience to the maturation of orientation selectivity in ferret visual cortex. Nature 411:1049–1052. CrossRef Medline

Wiesel TN, Hubel DH (1963) Single-cell responses in striate cortex of kittens deprived of vision in one eye. J Neurophysiol 26:1003–1017. Medline

Wong RO (1999) Retinal waves and visual system development. Annu Rev Neurosci 22:29–47. CrossRef Medline

Zhang W, Linden DJ (2003) The other side of the engram: experience-driven changes in neuronal intrinsic excitability. Nat Rev Neurosci 4:885–900. CrossRef Medline

frontiers in
NEUROINFORMATICS

ORIGINAL RESEARCH ARTICLE
published: 30 December 2013
doi: 10.3389/fninf.2013.00044

# An automated and reproducible workflow for running and analyzing neural simulations using Lancet and IPython Notebook

***Jean-Luc R. Stevens*[1]*, Marco Elver[2] and James A. Bednar[1]***

[1] *School of Informatics, Institute for Adaptive and Neural Computation, University of Edinburgh, Edinburgh, UK*
[2] *School of Informatics, Institute for Computing Systems Architecture, University of Edinburgh, Edinburgh, UK*

***Edited by:***
*Andrew P. Davison, CNRS, France*

***Reviewed by:***
*Padraig Gleeson, University College London, UK*
*Thomas G. Close, Okinawa Institute of Science and Technology Graduate University, Japan*

****Correspondence:***
*Jean-Luc R. Stevens, School of Informatics, Institute for Adaptive and Neural Computation, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, UK*
*e-mail: jlstevens@inf.ed.ac.uk*

Lancet is a new, simulator-independent Python utility for succinctly specifying, launching, and collating results from large batches of interrelated computationally demanding program runs. This paper demonstrates how to combine Lancet with IPython Notebook to provide a flexible, lightweight, and agile workflow for fully reproducible scientific research. This informal and pragmatic approach uses IPython Notebook to capture the steps in a scientific computation as it is gradually automated and made ready for publication, without mandating the use of any separate application that can constrain scientific exploration and innovation. The resulting notebook concisely records each step involved in even very complex computational processes that led to a particular figure or numerical result, allowing the complete chain of events to be replicated automatically. Lancet was originally designed to help solve problems in computational neuroscience, such as analyzing the sensitivity of a complex simulation to various parameters, or collecting the results from multiple runs with different random starting points. However, because it is never possible to know in advance what tools might be required in future tasks, Lancet has been designed to be completely general, supporting any type of program as long as it can be launched as a process and can return output in the form of files. For instance, Lancet is also heavily used by one of the authors in a separate research group for launching batches of microprocessor simulations. This general design will allow Lancet to continue supporting a given research project even as the underlying approaches and tools change.

**Keywords: IPython, pandas, reproducibility, workflow, simulation, batch computation, provenance, big data**

## 1. INTRODUCTION

Computational neuroscience is a rapidly developing scientific field that relies on a large ecosystem of software tools that is continually evolving as high-performance computing infrastructure is updated. Every computational neuroscientist must therefore keep up with new developments in neuroscience, software engineering, and computer hardware while advancing novel computational theories of the nervous system. The drive to explore different scientific hypotheses rapidly has made Python the language of choice for many researchers due to its flexibility and wide range of libraries already provided. Despite this fast pace of change, it is crucial that results remain reproducible once they are obtained, if computational neuroscientists are to have long-term confidence in the integrity of their work.

The formidable challenges associated with developing replicable scientific publications in a rapidly advancing field are well recognized by the computational neuroscience community. The difficulties include problems replicating results between simulators (Crook et al., 2013) and insufficiently constrained model parameters in publications (Nordlie et al., 2009), along with an important debate about the distinction between replicability and reproducibility (Drummond, 2009; Freire et al., 2011). Fundamentally, neuroscience is concerned with the study of dynamic, history dependent biological systems of exceedingly high dimensionality. Although computational models abstract away most of the complexity of nervous systems by necessity, it is still a formidable challenge to communicate this type of work to other scientists while also capturing the key properties of the biological system under study. These broad issues must be addressed by the community as a whole, and cannot be solved by any one piece of software.

The approach we present to improve reproducibility is by offering a small number of useful utilities that first aim to improve a researcher's scientific productivity. If properly designed and useful enough to become a core part of a researcher's regular workflow, it is hoped that such tools will allow reproducible science to emerge naturally as researchers seek to increase productivity. This approach is in sharp contrast to more heavyweight automated scientific workflow systems (Curcin and Ghanem, 2008; Freire et al., 2014) that can be effective for mature research areas but would be constraining for this young and ever-changing field.

We developed the Lancet package as a small set of flexible, lightweight components that allow a researcher to generate and analyze large data sets more efficiently. These components are designed to help improve research efficiency by allowing the user to capture the essence of a scientific task with very little

code and by catching errors early on, before expensive computational processes begin. By distilling a problem into a small number of short, declarative specifications, the researcher can focus on important scientific details, spending less time worrying about issues of implementation. Every component in Lancet is written to satisfy an immediate need; the end goal of generating automated, reproducible results should then be satisfied as a natural outcome of a clean and efficient solution to a problem.

By design, Lancet is a general utility, allowing it to work with any external tool or simulator. This ensures that as tools change or as researchers switch between software and platforms, the code written with Lancet remains unchanged. This generality is strictly enforced by the requirements of one of the authors, who is successfully applying Lancet outside the domain of computational neuroscience, i.e., to run simulations of varying microprocessor architectures. Lancet is pure, platform-independent Python with minimal dependencies, and supports both Python 2 and Python 3. Together, these properties should help ensure that code written using this utility will remain viable for the foreseeable future.

The goal of this new package is to allow reproducible, agile workflows to develop organically when used together with other tools, namely a suitable version control system and IPython Notebook. Since version 0.12 of IPython (Pérez and Granger, 2007), a notebook feature has been provided which allows code, data, and figures to be interactively explored while maintaining a complete record of the source code. Lancet is designed to integrate well with IPython and the pandas library (pandas.pydata.org), without having either of these two projects as a core dependency.

The next section introduces the components of Lancet, starting with a very small toy example of a workflow that begins with an initial specification and ends in a simple analysis. Section 3 provides an overview of the three main types of components offered in Lancet. At every stage, we show how these components make research tasks easier to complete by making the intentions of exploratory and publication-specific code clearer and more succinct. With the basic design established, Section 4 presents the full reproducible workflow, showing how Lancet can help turn reproducible science into practical reality when used together with IPython Notebook and other popular tools such as Git and the pandas data analysis library. To demonstrate that this workflow is both practical and relevant to a real research project, we then briefly describe how it was used to generate all the results in Stevens et al. (2013), recently published in the Journal of Neuroscience.

## 2. BASIC LANCET EXAMPLE

Python is a flexible, interpreted language that comes with many modules that extend the functionality of the base language. Closely related modules are collected into packages, some of which are included together with Python in the standard library and others that are available as third party libraries. The new Lancet package is designed to work together with the many excellent Python packages already available for scientific computing, to help capture and simplify a researcher's workflow. Lancet integrates particularly well with the interactive IPython notebook environment, which improves on Python's facilities for exploratory research and works across multiple platforms (Linux, MacOS, Windows). More information about Lancet, including installation instructions, may be found on Lancet's website (http://ioam.github.io/lancet).

To introduce Lancet, we will first look at a minimal, toy example of a Python-based workflow with Lancet, listed in **Figure 1**. This example uses the simple factor command (included in GNU coreutils) to find the prime numbers that lie within a specific range of integers. Although brief, this example demonstrates how to use an initial specification of a parameter space to obtain results

```
>>> import lancet

>>> example_name   = 'prime_quintuplet'
>>> integers       = lancet.Range('integer', 100, 115, steps=16, fp_precision=0)
>>> factor_cmd     = lancet.ShellCommand(executable='factor', posargs=['integer'])
# Runs locally. A QLauncher could be used to launch jobs with Grid Engine.
>>> lancet.Launcher(example_name, integers, factor_cmd, output_directory='output')()

# Collate and print the the primes in the input range of integers
>>> def load_factors(filename):
...     "Return output of 'factor' command as dictionary of factors."
...     with open(filename, 'r') as f:
...         factor_list = f.read().replace(':', '').split()
...     return dict(enumerate(int(el) for el in factor_list))

>>> output_files   = lancet.FilePattern('filename', './output/*-prime*/streams/*.o*')
>>> output_factors = lancet.FileInfo(output_files, 'filename',
...                                  lancet.CustomFile(metadata_fn=load_factors))
>>> primes = sorted(factors[0] for factors in output_factors.specs
                    if factors[0]==factors[1]) # i.e., if the input integer equals the first factor
>>> primes
[101, 103, 107, 109, 113]
```

**FIGURE 1 | A simple, end-to-end workflow using Lancet to factorize a range of integers, highlighted using the three colors used in the bullet points at the start of Section 2.** This simple example factorizes a list of integers with the factor command, with no other dependencies. The five prime numbers found are an example of a prime quintuplet, the closest admissible constellation of five consecutive prime numbers.

collated across 16 independent jobs. Section 4 will show how this approach fits into an agile, exploratory workflow. Meanwhile, even this simple example illustrates some of the key component types that are commonly applicable to many research tasks:

- *What you aim to achieve*. It is common to define a parameter space to be explored by some simulator or analysis tool. In **Figure 1** this is the list of integers to factorize, highlighted in red. This level of specification expresses the scientific goal and is normally both *tool-independent* and *platform-independent*. Given a parameter space, it is conceivable that the desired results may be achieved using alternative software tools executed on different platforms. When exploring a parameter space, the key information is specified by the set of parameters explored and not by the details of the software used.
- *How you intend to achieve your goal*. This refers to the target software that runs a model or performs an analysis. In **Figure 1** this is the factor command which factorizes integers, as highlighted in green. This type of specification is often *platform-independent* but *tool-dependent*, encapsulating how a specific piece of software is to be invoked with tool-dependent arguments, independent of the computational platform on which the software is run.
- *Where you want to execute the task*. If the software can run on multiple different platforms, there may be alternative ways to execute the tool. Executing a task in a particular environment is normally *platform-dependent* but *tool-independent*. In **Figure 1** the factor command is executed locally using the Launcher class supplied by Lancet, highlighted in blue. By switching to the QLauncher class, the exact same task could be executed in parallel on a Grid Engine cluster without changing the rest of the code.

Of course, it is difficult to appreciate the advantages of using Lancet, if one simply wants to factor 16 small integers in Python. These advantages would be much more apparent if a multidimensional parameter space were to be explored with a complex neural simulator, as described below. Even so, non-Lancet Python code for launching these simple factor runs is likely to be longer, more error-prone and harder to read. Iteration over the input parameter space and output files (highlighted in red) would probably be expressed as multiple for loops, losing the flat structure of the example. Specification of the simulator (highlighted in green) and the code needed to execute it (highlighted in blue) would be interleaved and complex calls to the subprocess module would be required to execute jobs. Switching from local execution to Grid Engine would no longer be trivial.

This example demonstrates how Lancet can help free the researcher from such implementation details. Substantial code would also be needed to reproduce the way Lancet keeps your output files consistently organized (within timestamped folders by default) with a common directory structure, whether working locally or on a cluster. After executing the listing in **Figure 1**, a .info file will be generated together with the output, recording which Python version was used, the operating system on which the jobs were run, and the version of Lancet, alongside other useful metadata. Other information supplied by the user, such as the task description, versions of libraries and executables used, and other comments may be easily passed down to the metadata field of the .info file for storage. Lancet also offers a simple function that helps record version control information and improves reproducibility by maintaining an explicit log of all the parameters used. As shown later in **Figure 5**, all of this can be expressed clearly, succinctly, and declaratively, even for realistically complex sets of simulations.

## 3. USING LANCET TO RAPIDLY SPECIFY A TASK

The example in **Figure 1** briefly introduced the three core class hierarchies in Lancet. In this section, each of the three types is examined in greater detail, before in the next section we consider how Lancet can assist the natural development of an agile, reproducible workflow with IPython Notebook. A list of all the components available to the user, split into the three class families, is shown in **Table 1**.

First, Arguments declaratively specify the parameter space to be covered by a set of runs (see e.g., the Range object at the top of **Figure 1**, highlighted in red), or specify filenames and data of interest on the filesystem. The latter object type allows data on disk to be collated for analysis in Python, or for launching the next stage of a pipeline workflow.

Next, a Command class handles the interface to an external tool, allowing the rest of Lancet to remain simulator-independent. The example shown in **Figure 1** uses a ShellCommand, which is supplied with Lancet for basic support of command-line programs. For supporting complex tools and simulators, Command can be subclassed while reimplementing only a constructor and a call method. As a workflow develops over time, it is likely that a user will want to make a custom Command to allow full control over important tools being used, but the other components of Lancet will not normally need to be extended for most users.

Finally, a Launcher pulls together the Arguments and Command objects to launch the specified jobs on a particular platform. Currently, jobs can be run either locally with the Launcher, or with Grid Engine using the QLauncher. As the Launcher object accepts the other two core component types as arguments and is a fully declarative object (as are all Lancet components), a Launcher object fully specifies the intended

**Table 1 | Lancet components available for specifying jobs.**

| Arguments | Command | Launcher |
|---|---|---|
| Args, List, Log, Range, FilePattern, FileInfo, SimpleGradientDescent | ShellCommand, TopoCommand*, RunBatchCommand* | Launcher QLauncher |

*All Arguments are subclasses of Args and specify static sets of parameters, except for SimpleGradientDescent which is an example of dynamic parameter optimization. ShellCommand is generic and included with Lancet whereas the Command classes marked by an asterisk are included with the Topographica simulator; other tools may offer their own custom Command classes. The Launcher class runs jobs locally, but other options are easy to implement, such as the QLauncher class for use on clusters.*

parameter space, the command to execute, and the platform to execute it on.

### 3.1. SUCCINCTLY SPECIFYING A PARAMETER SPACE WITH LANCET

**Figure 2** demonstrates some of the fundamental properties of all Arguments objects. These objects express parameter spaces that will result in many sets of parameter values to be passed to an external analysis tool or simulator, e.g., as command-line arguments. These are simple, compositional objects designed to express declarations of intent, independently of the other two types of Lancet component.

Part A of **Figure 2** shows the most basic and explicit example of an Arguments definition, using an Args object to specify a static set of arguments. The list of dictionaries format is a verbose and completely flexible specification. However, this style of definition is neither succinct nor declarative, and therefore is not recommended unless absolutely necessary. Nonetheless, this constructor illustrates two key points: argument values are always paired with the corresponding argument name, and Lancet Args objects have a similar structure to the DataFrame objects used by the pandas data analysis library. As DataFrames accept an identical data format in the constructor, Lancet Args objects allow easy conversion to DataFrames via the dframe property (if the pandas library is available). This easy transition to the highly flexible pandas DataFrames data structure is a key part of enabling the agile workflow described in the next section. These objects are easy to create and automatically display themselves as HTML tables in the IPython Notebook environment.

Part B of **Figure 2** expresses an identical parameter space using a more readable, less error-prone approach that clearly conveys the intended structure of the parameter space. In the explicit format shown in part A, the first argument 'arg1' remains constant with a value of 1.0 whereas the argument 'arg2' ranges over the numbers 1.0, 2.0 and 3.0. As a result, this parameter space is conveniently described as the Cartesian product of a constant argument for 'arg1' and a Range object that defining a range of values for 'arg2'.

The Cartesian product (also called the "cross product") of different arguments is a natural way to specify parameter spaces, supported by Lancet Arguments via the multiplication operator. In imperative code, these appear as nested for loops where each parameter is iterated by one of the loops. The Cartesian product of Args(arg1=1) and the Range object is therefore a succinct way of declaring a parameter space with one argument kept constant as the second argument spans a range of values. Note that the Args object accepts arbitrary keyword arguments, allowing any constant values for named parameters to be easily declared.

Part C of **Figure 2** shows a generic example of what a parameter space might look like in a simple, hypothetical neural simulation. A range of excitatory and inhibitory strengths is covered and a homeostatic mechanism is toggled on and off using the List

```
>>> from lancet import Args, List, Range

# A. An explicit yet error-prone way of specifying three sets of arguments
>>> args1 = Args([{'arg1':1.0,'arg2':1.0},  {'arg1':1.0,'arg2':2.0}, {'arg1':1.0,'arg2':3.0}])
>>> args1.dframe        # Pandas DataFrame. Displays an HTML table in Notebook.
   arg1  arg2
0     1     1
1     1     2
2     1     3

# B. Equivalent to the above but less error-prone with the intent expressed more clearly
>>> args = Args(arg1=1) * Range('arg2', 1,3, steps=3)
>>> args.show()         # List arguments from slowly to fast varying.
0: arg1=1, arg2=1
1: arg1=1, arg2=2
2: arg1=1, arg2=3

# C. Generic example of a parameter space for some neuroscience simulation
>>> parameters = ( Range('exc', 1, 3, steps=10)
...              * Range('inh', 1, 3, steps=10)
...              * List('homeostasis', [True,False]))

>>> parameters.summary()
Items: 200
Varying Keys: 'exc', 'inh', 'homeostasis'

# D. Concatenation allows parameter spaces to be extended. For instance, a special case can be appended.
>>> all_parameters = parameters + Args(exc=1.0, inh=0.0, homeostasis=True)
```

**FIGURE 2 | Arguments express parameter spaces succinctly and declaratively. (A)** Example illustrating the most basic, most explicit use of the Args class to specify three sets of sequential arguments. **(B)** A more succinct and less error-prone way of specifying the same arguments. **(C)** An example expressing a parameter space for use with a hypothetical neural simulator. This parameter space covers a range of excitation and inhibition strengths, while toggling a homeostatic mechanism. **(D)** The concatenation operator allows arguments specified by Arguments objects to be sequenced, allowing special cases to be incrementally appended to a parameter space.

declaration. Although simple, this object expresses 200 different argument sets (each leading to an independent simulation), as shown by the summary method.

Finally, in part D of **Figure 2**, the second compositional operator for Arguments objects is shown. The addition operator can concatenate (or sequence) Arguments objects together. The result is an object that first covers the parameter space of the first Arguments object before spanning the parameter space of the second Arguments object. This is a useful way to segment a parameter space in a piece-wise manner, allowing special cases to be easily added or the behavior at singularities to be investigated.

Using the Cartesian product and concatenation operations on the three basic Arguments objects, Args, List, and Range, many common parameter spaces can be expressed in a readily understood, compositional format. Arguments composed out of these basic objects have the property that the parameter space explored is known ahead of time, before jobs are executed. Although this is typical for many research tasks, Lancet also allows parameter spaces to be explored in an online fashion, where results returned by the jobs determine what portion of the parameter space is to be explored at the next step. Online parameter space exploration algorithms can be implemented in Lancet by subclassing DynamicArguments.

**Figure 3** illustrates how Lancet can be used to dynamically explore a simple parameter space using the SimpleGradientDescent component. This instance of DynamicArguments is designed to demonstrate how a simple gradient descent algorithm operating on a single, scalar argument can operate in Lancet. In **Figure 3**, ShellCommand is used to run a short script that evaluates the function $f(x) = (x-3)^2$ on the input argument $\times$ when executed. SimpleGradientDescent then explores the local parameter space from the starting point $x = 0$ in steps of magnitude stepsize. Driven by the output of the script, SimpleGradientDescent descends the local gradient in $\times$ until it terminates at the local minimum, $x = 3$. In practice, well-established optimization procedures are likely to be more useful than this example class, such as those available in scipy.optimize, when trying to optimize parameter spaces that are not solvable analytically. Thus SimpleGradientDescent should be considered as one example of the types of DynamicArguments that can be implemented for advanced parameter space exploration procedures such as hill climbing or genetic algorithms.

In summary, the Arguments objects are declarative, composable objects that can vary from simple declarations of constant argument values to complex optimization procedures. In addition to the Arguments objects presented so far, Lancet offers FilePattern Arguments for matching filenames. The filenames found may then be used as arguments for a simulator, or used to specify a list of files for loading into the Python environment. There are also other more specialized Arguments objects such as Log, which allows previously explored parameter spaces to be loaded from the .log files saved by Lancet when running external tools.

### 3.2. SPECIFYING HOW LANCET SUPPORTS YOUR EXTERNAL TOOLS

There are many different simulators and analysis tools used in computational neuroscience, each constantly being developed and updated. Some popular neural simulators include Brian

```
>>> import os, stat, json, lancet

# Minimum of f(x) = (x-3)^2 is zero at x = 3
>>> code= """#!/usr/bin/env python
... import json, sys
... x = float(sys.argv[1])
... print json.dumps((x-3)**2)
... """

>>> script = os.path.join(os.getcwd(), 'simple_function.py')
>>> with open(script, 'w') as f:  f.write(code)
>>> os.chmod(script, os.stat(script).st_mode | stat.S_IXUSR)

>>> minimizer = lancet.SimpleGradientDescent('x', stepsize=1.0, output_extractor=json.loads)
>>> command = lancet.ShellCommand(script, posargs=['x'])
>>> lancet.Launcher('Minimize', minimizer, command, output_directory='output')()

>>> minimizer.summary()
Varying Keys: 'x'
Maximum steps allowed: 100
Step 0: Initially exploring arguments [{x=1.0},{x=-1.0}].
Step 1: Exploring arguments [{x=2.0},{x=0.0}] after receiving input(s) [4.0, 16.0].
Step 2: Exploring arguments [{x=3.0},{x=1.0}] after receiving input(s) [1.0, 9.0].
Step 3: Exploring arguments [{x=4.0},{x=2.0}] after receiving input(s) [0.0, 4.0].
Step 4: Terminated after receiving input(s) [1.0, 1.0].
Successfully converged. Minimum value of 0.0 at x=3.0.
```

**FIGURE 3 | Lancet allows dynamic exploration of parameter spaces using components of type DynamicArguments.** In this example, the minimum of $f(x) = (x-3)^2$ is found using SimpleGradientDescent, starting from $x = 0$ and terminating at the minimum where $x = 3$.

(Goodman and Brette, 2008), Neuron (Hines and Carnevale, 1997), and NEST (Gewaltig and Diesmann, 2007), each of which uses different custom command-line interfaces. The most general approach to support such a wide range of tools is to treat them as external executables run on the command line. If a command-line specification is impractical or not supported by a particular tool, it is straightforward to write a Command that instead writes the specification for a run to a file to be read by the external program.

Even if you have the option of working exclusively with Python, such as for the Brian simulator, there can be clear advantages to writing your Python scripts as independent tools that can be invoked on the command line. Firstly, doing so ensures that independent runs are genuinely separate, sandboxing execution into separate processes to guarantee that independent jobs will not interact in unexpected ways. This requirement for process independence is explicit when running jobs on a cluster (for instance, when using Grid Engine). It is therefore useful to define a command-line interface to your Python scripts (perhaps using the argparse module) if you want code that can be executed both locally and in parallel on a cluster. Finally, defining a clear command-line interface can help document your code and allows useful standalone utilities to be pulled out of your code base.

When invoking tools with a standard command-line interface, Lancet supplies ShellCommand which can help avoid writing explicit interfacing code in many situations. For instance, ShellCommand is used to invoke the factor command in **Figure 1**. The ShellCommand is an instance of a Command that defines how Lancet can invoke an external tool via the command line. ShellCommand only supports communication via command-line arguments, but other Command classes may e.g., generate specification files appropriate to the chosen tool.

For interfacing with complex external software, users will often need to write a new Command subclass to extend Lancet's functionality for the new tool. Writing such a class is straightforward, as the subclass only needs to implement a constructor and a __call__ method. The __call__ method is supplied with arguments generated by an Arguments object in dictionary format (along with optional runtime information) and the Command must then return a list of strings suitable for Python's subprocess.Popen class. If the tool needs to load arguments from file, the Command may also save part of the parameter list specification to disk in an appropriate format before the command is executed. As described in the Discussion section, a special Command type could also be used to group small, lightweight jobs to avoid startup overhead.

Such interfacing code is designed to be simple, allowing the user to easily support new tools as required. These new components can then be supplied in a "Lancet extension" which may be bundled with the external software. For instance, the Topographica project (Bednar, 2009) offers a sophisticated Command subclass in a file named lancext.py. This component can invoke the simulator with a particular model file and defines Python analysis and measurement code for execution across a specified list of simulation times. Note that in this particular use case, although the Command passes the model file path to the command line, all parameters are specified on the command line rather than in the model file.

The lancext.py code is sufficiently flexible to support day-to-day exploratory work using the simulator, and was used throughout the development of the results in Stevens et al. (2013). The Command used is called RunBatchCommand, and is highlighted in green in **Figure 5**. The overall approach is general enough to be applicable to any simulator or tool, ranging from simple programs like factor, to complex neural simulators like Topographica, or even for running complex software outside the scope of computational neuroscience, such as time-consuming microprocessor simulations.

### 3.3. SPECIFYING YOUR CHOSEN COMPUTATIONAL PLATFORM

The parameter space and the chosen tool are defined independently and do not interact until a platform is chosen by selecting a Launcher object. The purpose of a Launcher is to take an Arguments object declaring a parameter space and feed the instantiated arguments to the Command, which then passes the appropriate command specification back to the Launcher, which executes the tool on the appropriate platform. As all the components needed to launch jobs and generate data form the arguments of the Launcher, the printed representation (also known as the repr) of the Launcher captures a complete specification of how the output files are created.

As Lancet itself only uses cross-platform portions of the Python library, code that uses Lancet can work across operating systems (Linux, MacOS, Windows). One reason to subclass Command to support a given tool is to ensure appropriate command-line invocations are generated across different operating systems. Simple tools with a consistent format of command-line invocation can instead be safely launched with ShellCommand, on any operating system.

Lancet currently provides a basic Launcher class for running jobs locally, and a subclass QLauncher that launches jobs with Grid Engine. Although the jobs are launched in very different ways, both classes ensure that the output is organized consistently. This approach ensures that the rest of the researcher's code can be used as-is across all the available platforms. For instance, code that needs to locate output files can use the same approach regardless of whether the files were generated locally or on a cluster. This is an essential feature for an agile workflow: as your requirements grow, it is important to have the option to painlessly transition from readily accessible local computational resources to a high-throughput cluster that can run your jobs in parallel, and then back again for debugging.

Lancet's QLauncher component wraps the Grid Engine qsub command and has been extensively tested on an open-source variant of the original Grid Engine system (Son of Grid Engine, version 8.0.0e). QLauncher assumes only the basic options applicable across the various versions of Grid Engine (Sun/Oracle/Univa Grid Engine) and should be usable on any machine where a Grid Engine qsub command is available. More information about Grid Engine and the Son of Grid Engine project may be found at http://arc.liv.ac.uk/SGE/.

In addition to making the process of switching between platforms easy, Launchers help save important information alongside the output data that help ensure reproducibility and assist in later analysis. The .info file contains metadata which records

important details requested from the version control system, the active Python and Lancet versions, operating system information and the complete representation of the source Launcher. The .log file contains an explicit list of all parameters used, allowing output to be quickly associated with the parameters used to generate it. This feature provides scientific provenance information for data analysis, which is crucial because the files output by a tool do not necessarily include the scientifically relevant parameters that were used to generate that data.

## 4. A REALISTIC, AGILE, AND EVOLVABLE WORKFLOW

Having introduced the general facilities offered by Lancet, we now examine how it can enable an agile and reproducible workflow using IPython Notebook. The use of external Python packages as appropriate is encouraged, and in particular the pandas library has proven very useful for analyzing data. To keep track of the code in the various Python scripts and IPython notebooks that appear as the workflow develops, it is also encouraged to keep a log of development by means of frequent code commits. Lancet works well together with distributed version control systems like Git and Mercurial, or with management and tracking tools tailored towards scientific use, such as Sumatra (Davison, 2012).

Note that our proposed workflow using Lancet does not aim to be prescriptive or impose requirements on the user. It is our view that the researcher must primarily choose the tools that allows the most productive research possible. Our goal is therefore to make Lancet general and useful, allowing each researcher to organically develop their own workflow according to their own particular needs. By incorporating more Lancet components into your workflow over time, the code can become more succinct while increasing the overall level of automation and reproducibility. A schematic of how the workflow evolves over time is shown in **Figure 4** and the stages of a typical research project using Lancet and IPython Notebook are now described:

1. An excellent way to start exploratory research is by creating a new IPython notebook. This offers an unconstrained environment where new ideas can be rapidly coded, tested and discarded as necessary. Using text and Markdown cells, notes can be interleaved with code to keep track of new ideas that relate either to scientific material or to coding. In this exploratory phase, the notebook is likely to be fairly disorganized and

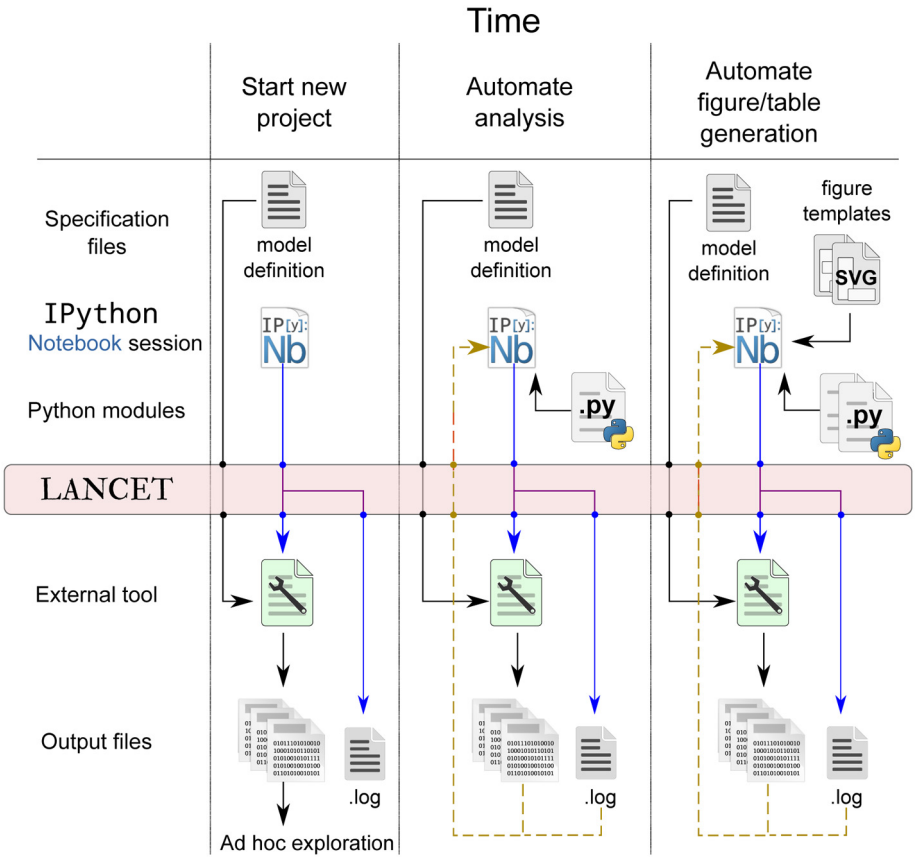


**FIGURE 4 | Lancet captures a full declarative specification of the parameters, tools and platform employed, each time data is generated at every stage of the workflow.** Early in the project, the output files may be rapidly explored in an *ad hoc* way that does not need to be automated or reproducible, as illustrated in the first column. As the research project matures, more of the analysis and plotting procedure may be pulled back into IPython Notebook where it can be automated (middle column). Finally, as the research nears publication, SVG templates may be used to ease the automatic generation of publication figures, as shown in the last column.

rapidly changing with many unrelated code snippets, outdated textual notes, HTML links, and other content (such as images) referencing external resources and documentation. Even so, even this early stage can be captured by committing the notebook to version control, preserving any progress made even though the user has not yet used any specific tool for reproducibility beyond the standard notebook.

2. Once a simulator or analysis tool has been chosen, small parameter spaces can be defined using the Arguments objects to be executed locally. If there is no Command available for the chosen tool, it is likely that ShellCommand will be sufficient to begin with. Otherwise, only a few lines of code are needed to subclass Command and satisfy the immediate requirements. At this stage, the output can be explored in an *ad hoc* manner, e.g., by inspecting files with a file manager or image viewer, as illustrated by the first column of **Figure 4**.
3. Lancet will store the repr (Python's term for an object's representation string) of the Launchers used along with the data in the .info files, maintaining a declarative record of how all the data was generated over time. As the project grows, it becomes crucial that version control is used to track notebook and code contents. A helper utility vcs_metadata is offered by Lancet that allows Git, Mercurial, or SVN version control information to be automatically stored in the .info files.
4. As the IPython Notebook is a very flexible environment for plotting and exploration, it quickly becomes worth writing small sections of Python code to automate away any *ad hoc* data inspection steps. It is also easy to load your data into the IPython notebook and rapidly generate plots with matplotlib. In particular, parameters associated with the loaded data can be brought into the notebook session by specifying a .log file to a Log Arguments object. This Log object may be used to re-run previously explored parameters, but also offers a convenient way to inspect and browse parameters previously logged by Lancet. By calling the dframe method of a Log object, a pandas DataFrame is generated that will present the logged parameters as an HTML table, offering a simple alternative to the web interface functionality offered by tools such as Sumatra. This stage is illustrated by the middle column of **Figure 4**.
5. Although small parameter spaces and local runs are often suitable initially when rapidly testing and debugging code, it is rare that this will prove sufficient for the whole project. As the code gets longer and more stable, it should be split out into Python modules to keep the notebook short and readable. As the code matures, parameter spaces tend to grow and simulation runs get longer and slower to obtain higher quality data sets. As the computational requirements increase, running simulations locally may become prohibitively slow, making it worth switching to a cluster if available. Lancet is designed to make such a transition painless: after switching Launcher for QLauncher and supplying a few basic settings appropriate to the cluster environment, the same code will immediately run in parallel on the cluster.
6. If a new Command class was implemented to support the external tool, this class may have matured to the stage where it is sufficiently general and flexible to become a reusable component, in which case it should also migrate to a separate file. By sharing this code with other Lancet users, the need to implement Commands will be alleviated in future as more and more tools are supported.
7. This particular stage of a research project may be quite prolonged, ending only when a particularly worthwhile avenue of research has been found. As the emphasis moves from exploration to publication, a particular subset of the code written is likely to become relevant. This code can be cleaned up and factored out into a Python module to keep the notebook manageable and to express the intentions of the developing paper clearly. Key plot types that are likely to become part of published figures may also be moved into a separate module.
8. In the final stages of developing a paper for submission, it can become cumbersome to generate complex, publication-quality figures using matplotlib alone. For this reason, to generate the final Figures in Stevens et al. (2013), a different approach was used—a small utility was written that allows SVG templates to be quickly authored in the Inkscape graphics editor. This utility then can then embed vector assets dynamically generated by Matplotlib to create the final, publication quality figure. At this stage, the notebook should embody a completely automated and reproducible workflow for published work, as illustrated by the final column of **Figure 4** and demonstrated for Stevens et al. (2013).

The key characteristic of this proposed workflow is that although the final outcome is an IPython notebook that captures and automates all the steps needed to generate a published result, there is no stage where the researcher needs any motive other than a desire to increase productivity. Writing a new Command to interface with a new external tool (if such a class is not already available) may at first appear more trouble than writing a simple, *ad hoc* script such as a shell script, a Python script using subprocess, or a script in some other language such as Perl. But the key difference is that the initial Command is normally trivial, using a few lines of code to return a fixed list of strings to the command line.

Unlike *ad hoc* scripts that can rapidly become unmanageable, a new Command class remains maintainable as it becomes more general and useful, remaining viable across multiple research projects. Implementing such an object allows the same, clean declarative representation to be seamlessly used with either local simulations or when working on a cluster. A workflow that relies on scripting solutions to individual problems as they appear is likely to become unreadable over time, and is unlikely to be reused between projects. To illustrate, the RunBatchCommand and associated classes implemented for the Topographica simulator now offer significantly more functionality for batch simulations than was initially available with the simulator. Although the latest Topographica Lancet extension is still under 500 lines of code and documentation, it has helped make regular research work with this simulator much easier than before.

So far, the declarative, reproducible nature of Lancet objects has only been demonstrated with very simple examples. **Figure 5** shows the full specification for a batch of Topographica simulations used in Stevens et al. (2013) in the form of a launcher repr. This newly created object can be run to regenerate the same data,

```
>>> info = json.load(open(info_path, 'r'))
>>> eval(info['launcher'])
QLauncher(
   batch_name='Fig05_06_seed102',
   args=Range(key='contrast', start_value=0, end_value=100, steps=21, fp_precision=2)
       * Args(cortex_density=98.0, lgn_density=24.0, retina_density=24.0, area=1.5)
       * Args(num_phase=8, num_orientation=20)
       * Args(times=[0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000,
                     11000, 12000, 13000, 14000, 15000, 16000, 17000, 18000, 19000, 20000])
       * Args(retinal_waves=0, figure='Fig05_06', input_seed=102,
              dataset='Gaussian', gain_control=False, homeostasis=False),
   command=RunBatchCommand(
      executable='/home/user/topographica/topographica',
      tyfile='./gcal.ty',
      analysis=Analysis(
         paths=['/home/user/topographica/models/stevens.jn13'],
         analysis_fns=[
            AnalysisFn(jn13_figures.lib.measurement.measure_FF),
            AnalysisFn(jn13_figures.lib.measurement.pinwheel_analysis),
            AnalysisFn(jn13_figures.lib.measurement.stability_analysis),
            AnalysisFn(jn13_figures.lib.measurement.afferent_CFs)]
         )
   ),
   output_directory='output'
)
```

**FIGURE 5 | A real example of recreating a launcher from the complete, declarative specification saved to the .info file.** The repr (the string representation) of the launcher is shown above, matching the corresponding string saved in the .info file. This example fully specifies 21 Topographica simulations used to generate Figures 5 and 6 from Stevens et al. (2013). Using a version control system also allows the state of the executed code (simulator, analysis, measurement code etc) to be restored based on the information stored in the .info file.

without needing the notebook that originally launched it. The printed representation of the Launcher object shown in **Figure 5** contains a real example of how the RunBatchCommand component is used in practice.

In this example, the .info file in one of the output directories is loaded using the json library and the contents of the launcher key is evaluated. As the repr of a Launcher is always saved to the .info file and this repr is a complete, declarative object that is a valid Python expression, running eval(info['launcher']) creates a new Launcher with identical behavior to the original. This object is easily inspected and captures the full set of parameters, including the path to the simulator executable, the executed Topographica model file, and a list of analysis functions to be executed repeatedly over the course of each simulation run.

Calling this object without supplying any arguments in a cluster environment would relaunch the 21 Topographica simulations necessary to regenerate Figure 5 from Stevens et al. (2013). This code will reproduce identical results, as long as the Topographica simulator is working correctly. If the results change due to differences in the simulator code, the recorded version control information allows all the code to be restored to the same state as when the data was originally generated. Note that the code listing in **Figure 5** is only one of the launchers needed to reproduce all the Figures in Stevens et al. (2013). In total, 842 simulation jobs were specified with Lancet to generate all the figures of the paper. Each job (simulation and analysis) takes over an hour to complete, so the full set of jobs takes several days to complete when running on a cluster, but the entire specification is still compact and human-readable.

## 5. DISCUSSION

This paper has demonstrated a lightweight, flexible, and pragmatic approach to achieving scientific reproducibility without constraining innovation. There are many other approaches also available, ranging from just writing a complete Python script to automate all your tasks, to using a heavyweight workflow-automation system. These more ambitious workflow engines are in regular use by large commercial organizations and research groups in some fields (Freire et al., 2011), but are not currently common in computational neuroscience. Such workflow engines are typically designed to manage complex workflows with long pipelines, involving many different people. In contrast, the workflow presented here is designed to be minimalistic, suitable for small groups of researchers who wish to keep their research work flexible and do not want to embrace more complex and prescriptive workflow tools.

Our aim is to show that for a general class of exploratory research in Python, using IPython Notebook and Lancet together allows for an agile workflow that very naturally gradually becomes more reproducible and automated over time. The final result of this process is a set of IPython notebooks that fully reproduce published scientific results, without constraining the user at any stage of the process. Lancet deliberately does not prescribe any fixed way of doing research, and every component offered to the user should be evaluated on the basis of how well it improves immediate research efficiency.

As a historical note, each of the components of Lancet was originally developed to satisfy the needs of a real research project spanning multiple years, not simply to try to achieve

reproducibility after the fact. In this project, many hundreds of simulations were executed locally using Lancet, and tens of thousands of jobs were launched on a cluster. But unlike the custom, *ad hoc* scripts that would normally be the result of such a project, Lancet was designed from the start to work just as well for completely different scientific domains, to ensure that the concepts and tools would be general and meaningful long into the future.

As a general tool, Lancet does not become any less relevant to research in computational neuroscience. To the contrary, having a general approach ensures that the essence of a workflow is valid over time as the underlying simulator tools come and go. The flexible and compositional nature of Lancet objects is suited to fast, exploratory research of interest to the computational neuroscience community using Python. Even though Lancet is newly available, it has already formed the basis for a complete scientific publication, made publicly available as an IPython notebook that automatically reproduces all the scientific results of the paper. This notebook allows all the code and results to be presented in a clear, automated way, and may be viewed and downloaded from the models/stevens.jn13 subdirectory of Topographica's GitHub repository.

For a tool that aims to be general, it is unsurprising that some functionality overlaps with other projects, given the many excellent third party libraries available for Python. For instance, there are several projects that offer sophisticated interfaces with Grid Engine, such as pythongrid and drmaa-python. IPython itself includes the IPython.parallel package which can help accelerate the pace of interactive work on a cluster. Some of the goals of Lancet's Arguments objects are shared by the parameters module of the NeuroTools package, which also allows parameter spaces to be defined. What distinguishes Lancet from these other libraries is that it offers all the tools needed to span an entire agile workflow with a collection of independent, declarative objects that work together.

Various workflow tools already exist with the computational neuroscientist in mind. VisTrails (Freire et al., 2014) is a scientific workflow and provenance system that integrates well with Python projects, taking a GUI-centric approach. The Mozaik framework (Antolík and Davison, 2013) is designed to encapsulate the workflows relevant to researchers who use spiking neural models. In contrast to these projects, Lancet is lightweight, with almost no dependencies, and is not tied to any particular set of simulator tools or workflows. Researchers exclusively using the appropriate spiking simulators may find Mozaik to be more specialized for their needs than Lancet, while Lancet is suitable for those who desire a more interactive workflow or need to use a broader class of tools or tools that are expected to change over time.

Projects like Sumatra (Davison, 2012) take a far more general approach for achieving reproducibility, tailoring functionality offered by version controls to the needs of the scientist. In this way, Sumatra offers functionality that is orthogonal to Lancet, allowing both tools to be used successfully together. Lancet's approach aims for the middle of the spectrum between Sumatra and Mozaik, capturing declarative specifications within Python code that assists with automation and reproducibility without losing generality. Lancet is BSD-licensed and supports Python 3, and helps the researcher exploit well-established tools such as IPython Notebook and pandas in a way that makes day-to-day research easier and ultimately makes results more reproducible.

Lancet is also extremely extensible. The interface between Lancet objects has been deliberately kept simple, to allow new components to be added whenever required. The Command class allows Lancet to work with new external tools, invoking the tool appropriately for each set of arguments specified. In some situations, individual jobs may run quickly relative to the time for setup and initialization, making it inefficient for Lancet to span the parameter space directly. In such cases, Lancet can instruct the tool to cover the parameter range itself, with Lancet only specifying starting and stopping points (e.g., Args(start = 0, end = 5)). If necessary, the Command object could then use these values to build a range specification in a format the tool can use.

The process of executing jobs may also be customized to satisfy specific needs. For instance, there are currently two types of Launcher, one for running jobs locally and one for running jobs on Grid Engine. Other types of Launcher may be written to extend Lancet to new platforms. For instance, it should be very straightforward to write a Launcher that launches jobs over SSH, or one that allocates computational resources on demand with Amazon EC2. This new Launcher would then fit seamlessly into the other components offered by Lancet.

The Arguments objects are also designed to be extensible. Although the basic objects offered are already suitable for many research requirements, new Arguments objects can be written if desired. By building a new DynamicArguments component, Lancet can be used for more complex, online parameter space exploration, utilizing optimization techniques such as hill climbing or genetic algorithms. Currently, SimpleGradientDescent is the only such object supplied with Lancet, designed to demonstrate how more practical algorithms may be quickly implemented. It is hoped that the ability to employ optimization algorithms as necessary will extend the utility of Lancet and that by making use of mature, third party libraries, users will easily be able to rapidly implement the optimization procedures necessary to solve their problems.

Of course, it is important to remember that Lancet is just one small part of a toolset for achieving reproducibility. More-basic tools like Python, pandas, and matplotlib are crucial for making it practical to automate scientific tasks, which is a prerequisite for being able to capture the process for later playback. Distributed version control systems like Git and Mercurial make it easy to capture the state of anything that can be expressed in text. IPython Notebook and matplotlib make it feasible to explore and analyze results in a text-based way that can be captured by the VCS. Lancet simply helps tie these together with launching runs and collating the results, to fill in the missing pieces that allow the entire process to become reproducible in practice. In that way, it addresses the fundamental barrier to reproducibility, which is the large and extra investment of time and effort that would be needed to automate and preserve tasks once the research has been published.

Essentially, what Lancet offers are the missing utilities that make it easy to capture all the required steps within a single IPython notebook, from initial exploration to published results. Using Lancet you can quickly specify and launch jobs, keep output

files consistently organized, switch from local execution to working on a cluster, record metadata and other key information together with your data, and load simulation output back into the notebook for analysis and plotting. By keeping everything under version control, the entire scientific process can then be captured, providing a flexible and agile yet reproducible research workflow.

The IPython notebooks that fully and automatically reproduce Stevens et al. (2013) are publicly available from the GitHub repository of the Topographica project (www.topographica.org) in the models/stevens.jn13 directory (https://github.com/ioam/topographica/tree/master/models/stevens.jn13). Lancet itself is freely available under a BSD license and may be downloaded from http://ioam.github.io/lancet/. Other examples of using Lancet are available at these Web sites.

## ACKNOWLEDGMENTS

The work has made use of resources provided by the Edinburgh Compute and Data Facility (ECDF; www.ecdf.ed.ac.uk). Thanks to Philipp Rüdiger for his helpful comments and suggestions.

## FUNDING

This work was supported in part by grants EP/F500385/1 and BB/F529254/1 to the University of Edinburgh Doctoral Training Centre in Neuroinformatics and Computational Neuroscience (www.anc.ed.ac.uk/dtc) from the UK EPSRC, BBSRC, and MRC research councils.

## REFERENCES

Antolík, J., and Davison, A. P. (2013). Integrated workflows for spiking neuronal network simulations. *Front. Neuroinform.* 7:34. doi: 10.3389/fninf.2013.00034

Bednar, J. A. (2009). Topographica: building and analyzing map-level simulations from Python, C/C++, MATLAB, NEST, or NEURON components. *Front. Neuroinform.* 3:8. doi: 10.3389/neuro.11.008.2009

Crook, S. M., Davison, A. P., and Plesser, H. E. (2013). "Learning from the past: approaches for reproducibility in computational neuroscience," in *20 Years of Computational Neuroscience*, ed J. Bower (New York, NY: Springer), 73–102.

Curcin, V., and Ghanem, M. (2008). "Scientific workflow systems - can one size fit all?" in *Cairo International Biomedical Engineering Conference (CIBEC)* (Cairo: IEEE Computer Society), 1–9.

Davison, A. (2012). Automated capture of experiment context for easier reproducibility in computational research. *Comput. Sci. Eng.* 14, 48–56. doi: 10.1109/MCSE.2012.41

Drummond, C. (2009). "Replicability is not reproducibility: nor is it good science," in *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th International Conference on Machine Learning* (Montreal: SITE, University of Ottawa). Available online at: http://www.site.uottawa.ca/ICML09WS.

Freire, J., Bonnet, P., and Shasha, D. (2011). Exploring the coming repositories of reproducible experiments: challenges and opportunities. *Proc. VLDB Endow.* 4, 1494–1497. Available online at: http://www.vldb.org/pvldb/vol4/p1494-freire.pdf

Freire, J., Koop, D., Chirigati, F., and Silva, C. (2014). "Reproducibility using VisTrails," in *Implementing Reproducible Computational Research*, eds V. Stodden, F. Leisch, and R. Peng (Boca Raton, FL: Chapman & Hall/CRC), (in press). Available online at: http://www.crcpress.com/product/isbn/9781466561595

Gewaltig, M.-O., and Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia* 2:1430. doi: 10.4249/scholarpedia.1430

Goodman, D. F. M., and Brette, R. (2008). Brian: a simulator for spiking neural networks in Python. *Front. Neuroinform.* 2:5. doi: 10.3389/neuro.11.005.2008

Hines, M. L., and Carnevale, N. T. (1997). The NEURON simulation environment. *Neural Comput.* 9, 1179–1209. doi: 10.1162/neco.1997.9.6.1179

Nordlie, E., Gewaltig, M.-O., and Plesser, H. E. (2009). Towards reproducible descriptions of neuronal network models. *PLoS Comput. Biol.* 5:e1000456. doi: 10.1371/journal.pcbi.1000456

Pérez, F., and Granger, B. E. (2007). IPython: a system for interactive scientific computing. *Comput. Sci. Eng.* 9, 21–29. doi: 10.1109/MCSE.2007.53

Stevens, J.-L. R., Law, J. S., Antolík, J., and Bednar, J. A. (2013). Mechanisms for stable, robust, and adaptive development of orientation maps in the primary visual cortex. *J. Neurosci.* 33, 15747–15766. doi: 10.1523/JNEUROSCI.1037-13.2013

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 04 November 2013; accepted: 13 December 2013; published online: 30 December 2013.*

*Citation: Stevens J-LR, Elver M and Bednar JA (2013) An automated and reproducible workflow for running and analyzing neural simulations using Lancet and IPython Notebook. Front. Neuroinform. 7:44. doi: 10.3389/fninf.2013.00044*

*This article was submitted to the journal Frontiers in Neuroinformatics.*

# HoloViews: Building Complex Visualizations Easily for Reproducible Science

Jean-Luc R. Stevens‡†∗, Philipp Rudiger‡†, James A. Bednar‡

**Abstract**—Scientific visualization typically requires large amounts of custom coding that obscures the underlying principles of the work and makes it difficult to reproduce the results. Here we describe how the new HoloViews Python package, when combined with the IPython Notebook and a plotting library, provides a rich, interactive interface for flexible and nearly code-free visualization of your results while storing a full record of the process for later reproduction.

HoloViews provides a set of general-purpose data structures that allow you to pair your data with a small amount of metadata. These data structures are then used by a separate plotting system to render your data interactively, e.g. within the IPython Notebook environment, revealing even complex data in publication-quality form without requiring custom plotting code for each figure.

HoloViews also provides powerful containers that allow you to organize this data for analysis, embedding it whatever multidimensional continuous or discrete space best characterizes it. The resulting workflow allows you to focus on exploring, analyzing, and understanding your data and results, while leading directly to an exportable recipe for reproducible research.

**Index Terms**—reproducible, interactive, visualization, notebook

## Introduction

Scientific research alternates between stretches of speculative, exploratory investigation and periods where crucial findings are distilled and disseminated as publications or reports. The exploratory phase typically involves running many different analyses with interactive plotting tools before the important aspects of the data are determined. The final results are then typically prepared as static figures for dissemination, often putting together many subfigures into a complicated figure that reveals multiple interrelated aspects of the results.

Current software tools provide relatively poor support for this dual exploring/reporting nature of scientific research, severely limiting scientific progress. On the one hand, developing new exploratory visualizations typically requires large amounts of custom software coding, which is slow, error-prone, and distracts from the actual scientific analysis. Moreover, this process typically involves a large amount of trial and error, generating transitory code and analyses that make it difficult to later reproduce the steps that led to any particular result [Cro13]. Switching to different tools for final, non-interactive, publication-quality figures exacerbates this problem, further disconnecting the reported results from the process by which they were created. This lack of reproducibility is a serious handicap both for progress within a single lab and for the community as a whole, making it nearly impossible for researchers to build on each others' work even for purely computational projects [Cro13].

Here we will describe a new Python software package built to address these problems directly, by providing simple tools for gradually building elaborate visualizations and analyses interactively yet reproducibly. HoloViews supports immediate exploration of data as it is obtained, without requiring custom coding, and then supports incrementally revealing more complex relationships between datasets, culminating in the final publication of fully reproducible scientific results.

In this paper we will focus on the high-level design principles that allow HoloViews to achieve these goals and we encourage the reader to visit holoviews.org for concrete examples. As detailed below, we show how this is achieved by enforcing a strict separation in the declaration of the semantic properties of the data and the specification of plotting options, allowing the user to declaratively specify their intent and let HoloViews handle the visualization.

### *The interactive interpreter*

To understand this approach, we need to consider the history of how we interact with computational data. The idea of an interactive programming session originated with the earliest LISP interpreters in the late 1950s and remains a popular way to interact with dynamic languages such as Python.

However, like most such command prompts, the standard Python prompt is a text-only environment. Commands are entered by the user, parsed, and executed, with results displayed as text. This offers immediate feedback and works well for data that is naturally expressed in a concise textual form. Unfortunately, this approach begins to fail when the data cannot be usefully visualized as text, as is typical for the large datasets now commonplace. In such instances, a separate plotting package offering a rich graphical display would normally be used to present the results outside the environment of the interpreter, via a graphical user interface.

This disjointed approach reflects history: text-only environments, where interactive interpreters were first employed,

† *These authors contributed equally.*
∗ *Corresponding author: jlstevens@ed.ac.uk*
‡ *Institute for Adaptive and Neural Computation, University of Edinburgh*

appeared long before any graphical interfaces. To this day, text-only interpreters are standard due to the relative simplicity of working with text. Proprietary attempts to overcome these limitations, such as the Mathematica Notebook [Wol03], have remained constrained by limited interoperability and a lack of standardized open formats. Other approaches focusing explicitly on reproducibility involve building a recipe for reproducing results only at the end of the scientific project [knitr], when it is often too late to capture the important steps involved. Here we consider how graphical output can be integrated fully into an interactive workflow, addressing both exploration and reproducibility simultaneously.

### *Fixing the disconnect between data and representation*

At the same time as text-based interpreters have failed to overcome the inherent limitations of working with rich data, the web browser has emerged as a ubiquitous means of interactively working with rich media documents. In addition to being universally available, web browsers have the benefit of being based on open standards that remain supported almost indefinitely. Although early versions of the HTML standard only allowed passive page viewing, the widespread adoption of HTML5 has made it possible for anyone to interact with complex, dynamic documents in a bi-directional manner.

The emergence of the web browser as a platform has been exploited by the Python community and the scientific community at large with tools such as the IPython Notebook [Per07] and SAGE MathCloud [Ste05]. These projects offer interactive computation sessions in a notebook format instead of a traditional text prompt. Although similar in design to the traditional text-only interpreters, these notebooks allow embedded graphics or other media (such as video) while maintaining a record of useful commands in a rich document that supports the gradual development of a document with interleaved code, results, and exposition.

Yet despite the greatly improved interactive capabilities of these tools, the spirit of the original interpreter has not yet been restored: there is still an ongoing disconnect between data and its representation. This artificial distinction is a lingering consequence of text-only displays, forcing a strict split between how we conceptualize "simple" and "complex" data. Although the IPython notebook now offers the means to give objects rich media representations, few packages have so far embraced this and none have supported easy composition of related figures. As a result the most common way to visualize complex data remains for the user to specify a detailed list of steps to get subfigures using an external plotting package such as Matplotlib [Hun07], then often combining subfigures using a GUI-based image editor.

Here we introduce HoloViews, a library of simple classes designed to provide an immediately available representation for even complex data in notebooks, analogous to the way simple datatypes are displayed in interactive sessions. HoloViews is not a plotting package; instead, it offers a set of useful data structures paired with rich, customizable visual representations that display effortlessly in the IPython Notebook environment. The result is research that is more interactive, concise, declarative, and reproducible. Figure 1 shows a self-contained example of building a complex visualization showing the declaration of an `Image` object followed by an example of how to compose HoloViews objects together.

## Design principles

The core design principle of HoloViews is to *automatically* and *transparently* return and display declarative data structures to the user for immediate feedback without requiring additional code. Although this concept is familiar and intuitive when interactively working with simple data types, it is worth reviewing explicitly what is going on so that the appropriate graphical extension of these ideas is clear.

When executing an addition operation like `1 + 2.5` at a Python prompt, the expression is parsed, converted into bytecode, and then executed, resulting in the float value `3.5`. This floating-point value is immediately returned to the user in the appropriate displayable representation, giving the user immediate feedback. Of course, this representation is not the float itself, but the string `"3.5"`. Such strings are automatically generated by the interpreter, via the displayed object's `__repr__` method.

The Python interpreter also provides such automatic, immediate feedback for more complex data types like large NumPy arrays, but for such data the displayed string has very little utility because it is either incomplete or impractical. In a terminal, this restriction is a result of the `__repr__` method only supporting a text-based display value. Using HoloViews in the IPython Notebook, you can give your array a more useful, interpretable default visual representation as an image, curve, or similar plot according to the following principles:

- It must be easy to assign a useful and understandable default representation to your data. The goal is to keep the initial barrier to productivity as low as possible -- data should simply reveal itself.
- These atomic data objects (elements) should be almost trivially simple wrappers around your data, acting as proxies for the contained arrays along with a small amount of semantic metadata (such as whether the user thinks of some particular set of data as a continuous curve or as a discrete set of points).
- Any metadata included in the element must address issues of *content* and not be concerned with *display* issues -- elements should hold essential information only.
- There are always numerous aesthetic alternatives associated with rich visual representations, but such option settings should be stored and implemented entirely separately from the content elements, so that elements can be generated, archived, and distributed without any dependencies on the visualization code.
- As the principles above force the atomic elements to be simple, they must then be *compositional* in order to build complex data structures that reflect the interrelated plots typical of publication figures.

The outcome of these principles is a set of compositional data structures that contain only the essential information underlying potentially complex, publication-quality figures.

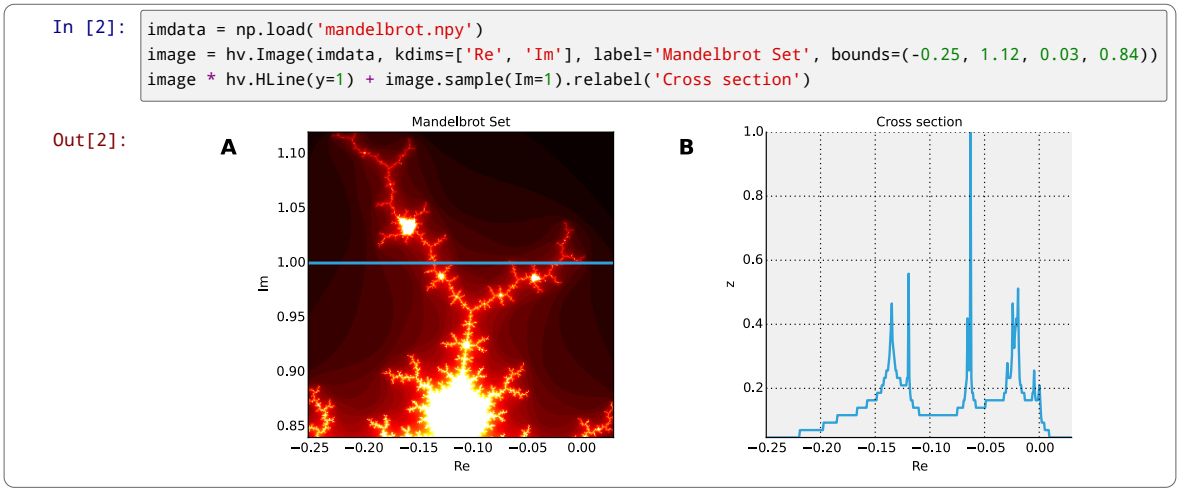


***Fig. 1:*** *Example of a composite HoloViews data structure and how it is displayed in an IPython Notebook session. The* `imdata` *array loaded using Numpy corresponds to the displayed portion of the Mandelbrot set.* ***A.*** *The* `Image` *element displays* `imdata` *overlaid via the* `*` *operator with a horizontal line element (*`HLine`*).* ***B.*** *A* `Curve` *element generated via the* `.sample()` *method of the image, showing a cross-section of the fractal along the indicated blue horizontal line. The curve is concatenated with the* `Overlay` *in* ***A*** *via the* `+` *operation.*

These data structures have an understandable, default visualization that transparently reveals their contents, making them a useful proxy for the data itself, just as the text `3.5` is a proxy for the underlying floating-point value. This default visualization may then be customized declaratively to achieve the desired aesthetics, without complicating the objects themselves.

In the next section we will discuss the data structures that hold the important content. Starting with the simple primitive elements, we examine how they can be composed into complex figures and embedded in high-dimensional spaces for exploration. Along the way we will discover how our implementation realizes the design principles outlined and manages to keep the state of the data separate from its visual representation.

## Data Structures

In this section we discuss the data structures that hold the raw data and the essential semantic content of interest. The Elements section introduces each of the primitives, and the Collections section explains how they can be combined. Finally, we will discuss working with Elements embedded in high-dimensional continuous or discrete spaces.

### *Elements*

The atomic classes that wrap raw data are the `Element` primitives. These classes are named by the natural representation they suggest for the supplied data, with `Image`, `Curve`, and `Scatter` being some simple examples. These elements are easily constructed as they only require the raw data (such as a NumPy array) to display.

In Figure 1, we have some examples of the Element primitives. On the left, in subfigure **A**, we see the `Image` primitive containing a two-dimensional NumPy array. This `Image` is declared by supplying the NumPy array `imdata` along with the optional metadata, including a suitable label and a declaration of the bounding region in the complex plane. The visual output is automatically generated and shows that the array is a part of the Mandelbrot set. Our object merely holds the supplied NumPy array, which remains easily accessed via the `.data` attribute. In part **B** of Figure 1 we have an example of a `Curve` containing a horizontal cross section of the image, as computed by the `sample` method.

Although the names of the `Elements` suggest that these objects are about visualization, they are primarily concerned with content and *not* display. The visually meaningful class names offer a convenient way to intuitively understand the dimensionality of the data in terms of an appropriate visual representation. For instance, in Figure 1 **A**, the name `Image` conveys the notion that the contained data is in the form of a two-dimensional NumPy array that can be meaningfully displayed as an image.

The particular `Image` shown in Figure 1 **A** was constructed as a visualization of the Mandelbrot Set, defined in the complex plane. In particular, the `kdims` argument declares that the *x*-axis is along the real axis and that the *y*-axis is along the imaginary axis. This information is then reflected in the visual output by assigning the appropriate axis labels. This semantic information is also passed to the `Curve` object generated by sampling the image using `image.sample(Im=1)`.

This `Curve` object is also able to pass on this semantic information to other Elements with different visual representations so that they faithfully reflect the space in which the Mandelbrot Set is defined. For instance, you can pass the curve directly to the constructor of the `Scatter` or `Histogram` elements and a new visual representation of the resulting object will retain the original semantic dimension labels. This type of operation merely changes the representation associated with the supplied data.

Note that in the declarations of `Image`, the dimensions of the axes are declared as key dimensions (`kdims`). Key dimensions correspond to the independent dimensions used to index or slice the element, with the remaining dimensions

called value dimensions (`vdims`). In the case of this image, there is a single value dimension, for the values in the supplied NumPy array, which are then visualized using the default colormap of the `Image` elements (the 'hot' color map).

As key dimensions are indexable and sliceable, we can slice the `Image` to select a different subregion of the Mandelbrot Set. Continuous values are supported when slicing an `Image` and the result is then a new `Image` containing the portion of the original NumPy array appropriate to the specified slice. The mapping between continuous space and the discrete array samples is specified by the bounds, allowing us to apply the slice `[-0.2:0,0.85:1.05]` to select the corresponding part of the complex plane. The first component of this slice selects the first key dimension (the real axis `'Re'`) from *-0.2* to *0.0* while the second component of the slice selects the second key dimension (the imaginary axis `'Im'`) from *0.85* to *1.05*. You can apply a similar slice along the real axis to select a portion of the curve object shown in Figure 1 **B**.

There are many additional element classes, one for each of the common visual representations for data. These elements form an extensible library of primitives that allow the composition of data structures with complex, meaningful visualizations. Within the set of all elements, you can cast your data between representations so long as the number of key and value dimensions is consistent. You can then index and slice your elements along their respective key dimensions to get new elements holding the appropriately sliced data of interest.

### *Collections*

The elements are simple wrappers that hold the supplied data and allow a rich, meaningful default representation. An individual element is therefore a data structure holding the semantic contents corresponding to a simple visual element of the sort you may see in a publication. Although the elements are sufficient to cover simple cases such as individual graphs, raster images, or histogram, they are not sufficient to represent more complex figures.

A typical published figure does not present data using a single representation, but allows comparison between related data items in order to illustrate similarities or differences. In other words, a typical figure is an object composed of many visual representations combined together. HoloViews makes it trivial to compose elements in the two most common ways: concatenating representations into a single figure, or overlaying visual elements within the same set of axes.

These types of composition are so common that both have already been used in Figure 1 as our very first example. The `+` operation implements concatenation, and `*` implements overlaying elements together. When you compose an object using the `+` operator, a default four-column layout is used but you can specify the desired number of columns using the `.cols` method. Layouts are easily specified but also support multiple options for customizing the position and sizing of elements.

When we refer to subfigures 1 **A** and 1 **B**, we are making use of labels generated by HoloViews for representing a composite data structure called a `Layout`. Similarly, subfigure 1 **A** is itself a composite data structure called an `Overlay` which, in this particular case, consists of an `Image` element overlaid by the `HLine` element.

The overall data structure that corresponds to Figure 1 is therefore a `Layout` which itself contains another composite collection in the form of an `Overlay`. The object in Figure 1 is in fact a highly flexible, compositional tree-based data structure: intermediate nodes correspond either to `Layout` nodes (`+`) or `Overlay` nodes (`*`), with element primitives at the leaf nodes. Even in this potentially complex tree, all the raw data corresponding to every visual element is conveniently accessible via key or attribute access by selecting a leaf element using its path through the tree, and then inspecting the `.data` attribute, making it simple to declare which part of a complex dataset you want to work with at a given time.

As any element may be a leaf of such a tree, there needs to be an easy way to select subtrees or leaf elements. This is achieved with a semantic, two-level labeling system using "group" and "label" strings supported throughout HoloViews. We have seen an example of a label string in Figure 1, where it was used to title the image "Mandelbrot Set". The textual representation of the layout in Figure 1 (see Out[6] of Figure 4) shows how the supplied label is used in the attribute-based indexing scheme of the layout. The strings "Image", "Overlay", "HLine" and "Curve" are default group names, but you can supply your own names to define semantic groupings for your data. To illustrate this system, you can access the sampled data (a NumPy array) in Figure 4 using `content.Curve.Cross_Section.data`.

With the ability to overlay or concatenate any element with any other, there is great flexibility to declare complex relationships between elements. Whereas a single element primitive holds semantic information about a particular piece of data, trees encode semantic information between elements. The composition of visual elements into a single visual representation expresses some underlying semantic value in grouping these particular chunks of data together. This is what composite trees capture; they represent the overall *semantic content* of a figure in a highly composable and flexible way that always preserves both the raw data and associated metadata for further interactive analysis and reproduction.

### *Spaces*

A single plot can represent at most a few dimensions before it becomes visually cluttered. Since real-world datasets often have higher dimensionality, we face a tradeoff between representing the full dimensionality of our data, and keeping the visual representation intelligible and therefore effective. In practice we are limited to two or at most three spatial axes, in addition to attributes such as the color, angle, and size of the visual elements. To effectively explore higher dimensional spaces we therefore have to find other solutions.

One way of dealing with this problem is to lay out multiple plots spatially. Plotting packages like ggplot [Wic09] and seaborn [Was14] have shown how this can be done easily using various grid-based layouts. Another solution is to present the data sequentially over time as an animation. A third solution is to provide interactive control, allowing the user to reveal

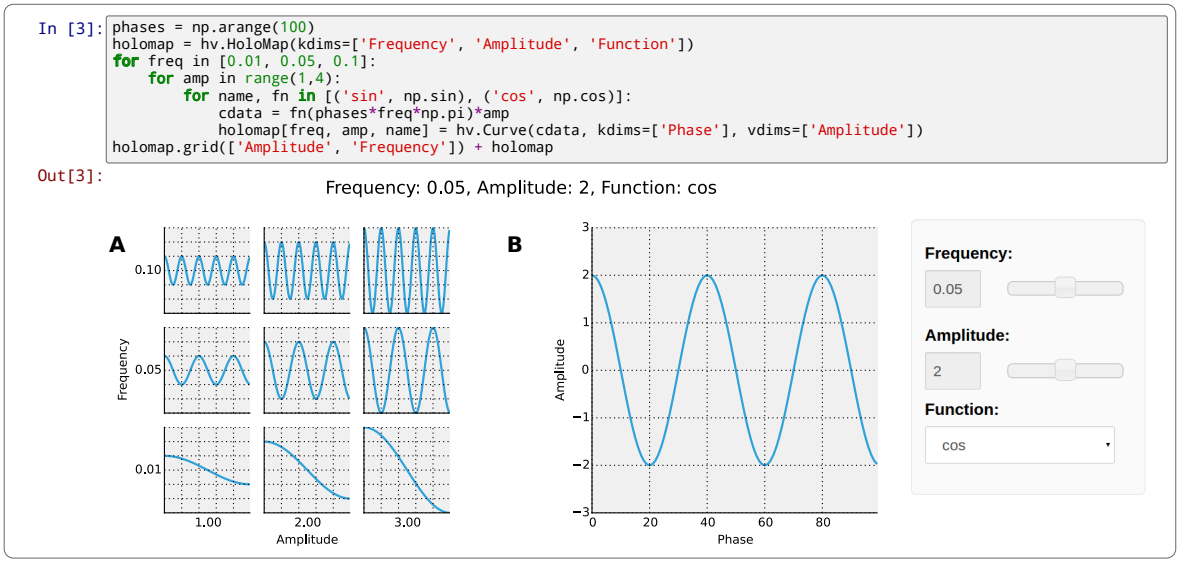


***Fig. 2:*** *Example of a Layout object containing two different representations of a multi-dimensional space. Both representations contain* `Curve` *objects embedded in three dimensions (*`Frequency`*,* `Amplitude`*,* `Function`*), but not all of these dimensions can be visualized at once. In* ***A****, two of the dimensions are mapped onto the rows and columns of a grid, and the remaining* `Function` *dimension can be selected using the widget at the right. In* ***B****, only a single curve is shown, with the three sliders at the right together selecting the appropriate curve from the 3D HoloMap space. When two* `HoloMaps` *are joined in a* `Layout` *like this, it will automatically find the joint set of dimensions the HoloMaps can be varied over. In this way HoloMaps allow users to explore data naturally and conveniently even when its dimensionality exceeds what can be sensibly displayed on the screen at once.*

further dimensionality by interacting with the plots using various widgets.

HoloViews provides support for all three of these approaches, via composable data structures that embed collections of `Element` objects in any arbitrarily dimensioned space. Fundamentally, this set of data structures (subclasses of `NdMapping`) are multi-dimensional dictionaries that allow the user to declare the dimensionality of the space via a list of key dimensions (`kdims`).

The list of supported `NdMapping` classes includes:

- `HoloMaps`: The most flexible high-dimensional data structure in HoloViews, allowing `Element` instances to be embedded in an arbitrarily high-dimensional space, to be rendered either as a video animation or as an interactive plot that allows exploration via a set of widgets.
- `GridSpaces`: A data structure for generating spatial layouts with either a single row (1D) or a two-dimensional grid. Each overall grid axis corresponds to a key dimension.
- `NdLayouts`/`NdOverlays`: Similar to `Layout` or `Overlay` objects, where the contained objects vary over one or more dimensions.

To explore a high-dimensional space of height as a function of age across different countries and years, you could declare `space=HoloMap(kdims=['Country', 'Year'])`. Now we can treat `space` as a dictionary and insert instances of classes such as `Curve` or `Scatter` with the appropriate `(country, year)` keys. For instance, the age and height `Curve` for the USA in 1988 (`usa`) can be inserted using `space['USA', 1988] = usa`. Note that the order of the indexing corresponds to the order of the declared key dimensions.

All of the above classes are simply different ways to package and view a high-dimensional dataset. Just as with `Elements`, it is possible to cast between these different spaces via the constructor. In addition, they can all be tabularized into a HoloViews `Table` element or a pandas `DataFrame` [McK10], a feature that is also supported by the `Element` primitives.

To get a sense of how composing data and generating complex figures works within this framework, we explore some artificial data in Figure 2. Here we vary the frequency and amplitude of sine and cosine waves, demonstrating how we can quickly embed this data into a multi-dimensional space. First, we declare the dimensions of the space we want to explore as the key dimensions (`kdims`) of the HoloMap. Next, we populate the space iterating over the frequencies, amplitudes, and the two trigonometric functions, generating each `Curve` element individually and assigning to the HoloMap at the correct position in the space.

We can immediately go ahead and display this HoloMap either as an animation or using the default widgets, as in Figure 2 **B**. Visualizing individual curves in isolation is not very useful, of course; instead we probably want to see how the curves vary across `Frequency` and `Amplitude` in a single plot. A `GridSpace` provides such a representation and by using the space conversion method `.grid()` we can easily transform our three-dimensional HoloMap into a two-dimensional GridSpace (which then allows the remaining dimension, the choice of trigonometric function, to be varied via the drop-down menu). Finally, after composing a `Layout` together with the original `HoloMap`, we let the display system handle the plotting and rendering.

If we decide that a different representation of the data would

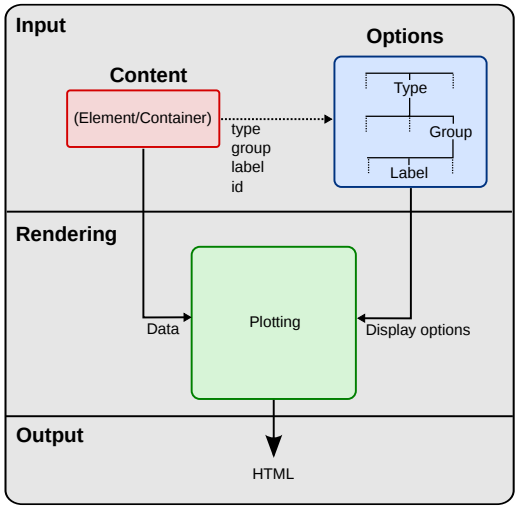


***Fig. 3:*** *This view of the HoloViews display and customization systems illustrates the complete separation between the content (data) to be displayed, the display options, and the rendering/plotting system. The display options are stored entirely separately from the content as a tree structure, with the appropriate options being selected with user-controllable levels of specificity: general options for all objects of a given type, more specific options controlled by user-definable* `group` *and* `label` *strings, or arbitrarily specific options based on the integer* `id` *assigned to each content object. Plotting and rendering happens automatically through the use of IPython display formatters. These combine the content with the specified display options, call an external plotting library, which returns an HTML representation that can then be rendered in the notebook.*

be more appropriate, it is trivial to rearrange the dimensions without needing to write new plotting code. Even very high-dimensional spaces can be condensed into an individual plot or expressed as an interactive plot or animation, by simply specifying which part of the data we are interested in rather than writing new brittle and error-prone custom plotting code.

### Customizing the visual representation

In this section we show how HoloViews achieves a total separation of concerns, keeping the composable data structures introduced above completely separate from both customization options and the plotting code. This design is much like the separation of content and presentation in HTML and CSS, and provides the same benefits of making the content easily maintainable while the presentation is easily controllable.

The only required connection between the above data structures and the custom display options is a single, automatically managed integer. Using this integer attribute we can make the data structures behave as if they were rich, stateful, and individually customizable objects, without actually storing anything to do with visualization on the objects. We will show how this separation is useful and extensible so that the user can quickly and easily customize almost every aspect of their plot. For instance, it is easy to change the font size of text, change the subfigure label format, change the output format (e.g. switch from PNG to SVG) and even alter the plotting backend (currently defaulting to Matplotlib) without changing any part of the underlying object being rendered.

Figure 3 provides an overall summary of how the different components in the display system interact. The declarative data structures define what will be plotted, specifying the arrangements of the plots, via Layouts, Overlays, and spaces. The connection between the data structure and the rendered representation is made according to the object type, the aforementioned integer attribute, and optionally specified group and label strings. By collecting the display options together and associating them with particular objects via these attributes, the visual representation of the content may be easily customized, e.g. to tweak aesthetic details such as tick marks, colors and normalization options. Once the user has specified both content and optionally customized the display the rendering system looks up the appropriate plot type for the object in a global registry, which then processes the object and looks up the specified options in order to display it appropriately. This happens transparently without any input from the user. Once the plotting backend has rendered the plot in the appropriate format, it will be wrapped in HTML for display in the notebook.

The default display options are held on a global tree structure similar in structure to the composite trees described in the previous section, but with nodes holding custom display options in the form of arbitrary keywords. In fact, these option trees also use labels and groups the same way as composite trees except they additionally support type-specific customization. For instance, you may specify colormap options on the `Image` node of the tree that will then be applied to all `Images`. If this chosen colormap is not always suitable, you can declare that all `Image` elements belonging to a group (e.g. `group='Fractal'`) should use a different colormap by overriding it on the `Image.Fractal` node of the tree. This form of inheritance allow you to specify complex yet succinct style specifications, applying to all objects of a particular type or just to specific subsets of them.

To explore how option setting works in practice, Figure 4 shows an example of customizing Figure 1 with some basic display options. Here we use an optional but highly succinct method for setting the options, an IPython cell magic `%%opts`, to specify aspect ratios, line widths, colormaps, and sublabel formats. By printing the string representation of the content (`Out[6]`) and the options (`Out[7]`), we can see immediately that each entry in the options tree matches a corresponding object type. Finally, in the actual rendered output, we can see that all these display options have taken effect, even though the actual data structure differs from the object rendered in Figure 1 only by a single integer attribute.

A major benefit of separating data and customization options in this way is that all the options can be gathered in one place. There is no longer any need to dig deep into the documentation of a particular plotting package for a particular option, as all the options are easily accessible via a tab-completable IPython magic and are documented via the `help` function. This ease of discovery enables a workflow where the visualization details of a plot can be easily and quickly iteratively refined once the user has found data of interest.

The options system is also inherently extendable. New options may be added at any time, and will immediately

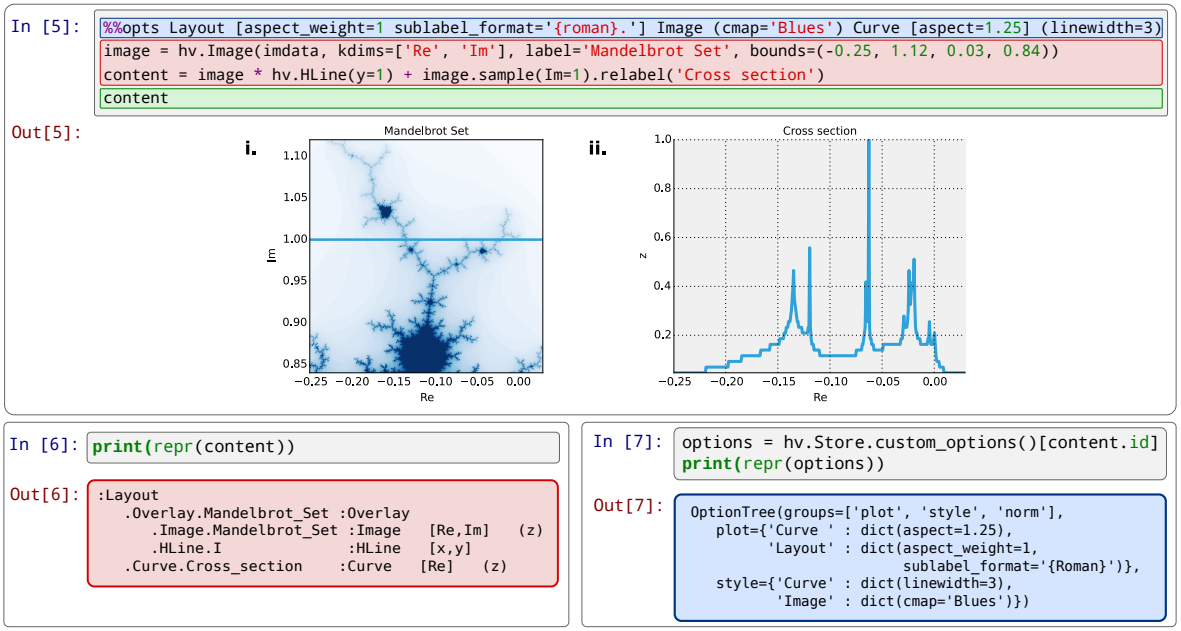


***Fig. 4:*** *An example of customizing the display of Figure 1's data using the default Matplotlib backend.* `In[5]` *is color coded according to the components in Figure 3, where red is the content, blue is the display options (using an optional IPython-specific succinct syntax), and green is what triggers the the rendering.* `Out[5]` *shows how the supplied options have affected the final plots, compared to Figure 1. Finally,* `Out[6]` *and* `Out[7]` *show the textual representations of the content and the style specification respectively, demonstrating how the two are separate yet linked.*

become available for tab-completion. In fact, the plotting code for each element and container type may be switched out completely and independently, and the options system will automatically reflect the changes in the available customization options. This approach lets the user work with a variety of plotting backends at the same time, without even having to worry about the different plotting APIs.

The separation between content, options and plotting explicitly supports the workflows that are common in science, repeatedly switching between phases of exploration and periods of writing up. Interesting data can be collected and curated over time, where each step is instantly and transparently visualizable without any custom code cluttering up the notebook. Visualizations of data that are worth keeping can be customized through an interactive and iterative process, and the final set of plotting options can then be expressed as a single data structure separate from the actual displayed data, ready to be applied to the next batch of data from a subsequent measurement or experiment. Throughout, the scientist curates the data of interest, as revealed in associated visual representations, along with the visualization options and a separate codebase of general-purpose plots (mostly included in HoloViews, but potentially extended locally for specific domains). Each of these three aspects of the process (data, options, and code) can be developed, maintained, archived, and improved independently, providing comprehensive support for the natural process of exploration and dissemination common to all scientific disciplines.

## Discussion

This paper demonstrates a succinct, flexible, and interactive approach for data exploration, analysis, and visualization. HoloViews restores the immediate feedback cycle that is characteristic of working with simple data in an interpreter. This is achieved by having declarative objects display themselves with good defaults allowing the user to immediately understand their data. In the majority of cases this eliminates the need to write plotting code and allows the user to keep a concise and reproducible recipe of their work, from exploration to the final publication. HoloViews thus allows scientists to capture the entire workflow involved in a research project.

Without a strictly enforced separation of concerns, workflow stages often end up mixing both data processing and visualization. Although a displayed representation is always necessary for understanding, it has been a dead end for further data processing. Because HoloViews objects represent themselves visually but also contain the raw data, the ability to continue processing is never terminated and exploration can continue. Furthermore, the chosen representation can easily be changed, turning what used to be a highly disjointed workflow into a open-ended process concerned with the semantics of the data. Only once results worth disseminating are attained does it become necessary to consider the details of visualization.

The compositionality of HoloViews is superficially reminiscent of systems such as the Grammar of Graphics [Wil05] for the R language, but the aim of HoloViews is quite different. Instead of expressing all the complexities of graphics, the declarative data structures in HoloViews define a language for the semantics of the actual data. This language focuses on how the researcher conceptualizes it, *independent* of the exact

details of plotting. The need for an automatic and useful visual representation is driven by the need to immediately present the data in a meaningful format.

HoloViews is one of many packages designed for working with large, multidimensional datasets, but it differs from each of these in important ways. For instance, Python's `seaborn` [Was14] and R's `ggplot2` [Wic09] library support laying out high-dimensional data into subplots and grids, while Python's Bokeh library and R's shiny [shiny] web application framework provide widgets for interactive data exploration. While each of these packages can provide extremely polished interactive graphics, getting them set up for specific sets of data requires significant additional effort and custom code, placing a barrier to their primary use case, the interactive exploration of data. HoloViews instead tries to avoid custom coding altogether as far as possible, with users instead supplying metadata to declare the properties of the data and option settings to control its visual appearance.

Although HoloViews is a general purpose library for working with data at every stage, it actually represents a significant advance over previous approaches focused only on achieving reproducibility of the final result. Simply by keeping specifications for figures succinct, HoloViews allows the entire recipe to be preserved in the notebook, not scattered over separately imported plotting code files. Secondly, because HoloViews can directly express the complex relationships between different bits of data as subfigures, it can capture entire figures within notebooks that would previously have required unreproducible work in external drawing programs. Lastly, HoloViews exports the actual data alongside published figures, allowing it to be tested automatically (as is done for the project web site) without conflating it with arbitrary display choices. HoloViews makes it possible to reproduce results from every step of the project, up to and including the final published figures, in a way that has not previously been practical.

Although HoloViews aims to provide good default behavior, scientific work often requires highly specialized visualizations. For that reason we have made it easy to extend the defaults and integrate new visualizations. Firstly, as many plotting and styling options as possible are exposed in an easily accessible manner, while providing a powerful, inheritance-based system for changing these options when required. Secondly, the options system has been designed to work well with the compositional data structures provided by HoloViews. Thirdly, HoloViews makes it trivial to add completely novel types of Elements with corresponding plots (or to override specific code in existing plots) using custom code when needed, and these custom plots will then combine seamlessly with other objects to make composite figures. Finally, not only is it possibly to implement new plot classes but entire plotting backends may be added and exposed to the user, such as the prototype Bokeh backend, which is well suited to live interaction and large datasets. Thus default plots are simple and straightforward, but even complex figures are easily achievable. Many such examples, ranging from simple to complex, can be found in the Tutorials and Examples sections of holoviews.org.

In this paper, we have focused on how a user can quickly build data structures for their content of interest. An even more powerful approach is for a developer to integrate HoloViews directly into a library, analysis tool, or simulator. By returning HoloViews objects (which do not depend on any plotting library), any Python package can immediately have access to flexible, compositional data structures that automatically double as a visualization system. This is exactly the approach taken by the ImaGen image generation library and the Topographica neural simulator, two very different projects that both output data wrapped in HoloViews data structures.

## Conclusion

Based on the key principles of: (1) making data immediately and transparently visualizable, (2) associating data directly with its semantic description, (3) keeping display option settings separate from the data, (4) keeping display code separate from both data and display options, (5) explicitly expressing the relationships between data elements compositionally, and (6) keeping the original data accessible even in complex visualizations, Holoviews supports the entire life cycle of scientific research, from initial exploration, to dissemination and publication, to eventual reproduction of the work and new extensions. Existing approaches for achieving some of these goals individually have been very limiting and only partially successful, each adding significant new costs along with the benefits they offer. HoloViews instead addresses the underlying problems fundamental to current methods for scientific research, solving seemingly intractable issues like reproducibility almost as a side effect of properly supporting the basic process of doing science.

## Acknowledgments

This work was funded in part by grant 1R01-MH66991 to the University of Texas at Austin from the USA National Institute of Mental Health, by grant EP/F500385/1 from the UK EPSRC and MRC research councils, and by the Institute for Adaptive and Neural Computation at the University of Edinburgh.

## References

[Cro13] Crook et al., "Learning from the Past: Approaches for Reproducibility in Computational Neuroscience", *20 Years of Computational Neuroscience*, J.M. Bower, ed., Springer, 9:73-102, 2013.

[Wol03] Stephen Wolfram, *The Mathematica Book*, Fifth Edition, Wolfram Media/Cambridge University Press, 2003.

[knitr] Foundation for Open Access Statistics, *knitr*, http://yihui.name/knitr, 2015.

[Per07] Fernando Perez and Brian E. Granger, IPython: a System for Interactive Scientific Computing, *Computing in Science and Engineering*, 9:21-19, 2007.

[Ste05] William Stein and David Joyner. SAGE: System for Algebra and Geometry Experimentation. *ACM SIGSAM Bulletin*, 39:61-64, 2005.

[Hun07] John D. Hunter, *Matplotlib: A 2D graphics environment*, Computing In Science & Engineering, 9(3):90-95, 2007.

[Wic09] Hadley Wickham, *ggplot2: elegant graphics for data analysis*, Springer New York, 2009.

[Was14] Michael Waskom et al.. *seaborn: v0.5.0*, Zenodo. 10.5281/zenodo.12710, November 2014.

[McK10] Wes McKinney, *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56, 2010.

[Wil05] Leland Wilkinson, *The Grammar of Graphics*, Springer-Verlag New York, 2005.

[shiny] RStudio, Inc, *shiny: Easy web applications in R.*, http://shiny.rstudio.com, 2014.

# Bibliography


Albrecht, D. G., Geisler, W. S., Frazor, R. A., and Crane, A. M. (2002). Visual Cortex Neurons of Monkeys and Cats: Temporal Dynamics of the Contrast Response Function. *Journal of Neurophysiology*, 88(2):888–913.

Antolik, J. and Bednar, J. A. (2011). Development of maps of simple and complex cells in the primary visual cortex. *Frontiers in Computational Neuroscience*, 5(17).

Bednar, J. A. (2008). Understanding Neural Maps with Topographica. *Brains, Minds, and Media*, 3:bmm1402.

Bednar, J. A. (2012). Building a mechanistic model of the development and function of the primary visual cortex. *Journal of Physiology-Paris*, 106(5-6):194–211.

Bednar, J. A. and Williams, C. K. I. (2016). *Neural Maps: Their Function and Development*, volume From Neuron to Cognition of *Computational Neuroscience*. MIT Press. In press.

Blasdel, G. and Salama, G. (1986). Voltage-sensitive dyes reveal a modular organization in monkey striate cortex. *Nature*, 321(6070):579–585.

Blasdel, G. G. (1992a). Differential imaging of ocular dominance and orientation selectivity in monkey striate cortex. *The Journal of Neuroscience*, 12(8):3115–3138.

Blasdel, G. G. (1992b). Orientation Selectivity, Preference, and Continuity in Monkey Striate Cortex. *The Journal of Neuroscience*, 12:3139–3161.

Bokeh Development Team (2014). *Bokeh: Python library for interactive visualization*.

Brette, R. and others, . (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23:349–398.

Bringuier, V., Chavane, F., Glaeser, L., and Fr/'egnac, Y. (1999). Horizontal Propagation of Visual Activity in the Synaptic Integration Field of Area 17 Neurons. *Science*, 283:695–699.

Brown, J. W. (2014). The tale of the neuroscientists and the computer: why mechanistic theory matters. *Frontiers in Neuroscience*, 8(349).

Buzás, P., Eysel, U. T., Adorján, P., and Kisvárday, Z. F. (2001). Axonal topography of cortical basket cells in relation to orientation, direction, and ocular dominance maps. *The Journal of Comparative Neurology*, 437(3):259–258.

Buzás, P., Kovács, K., Ferecskó, A. S., Budd, J. M. L., Eysel, U. T., and Kisvárday, Z. F. (2006). Model-Based Analysis of Excitatory Lateral Connections in the Visual Cortex. *The Journal of Comparative Neurology*, 499(6):861–881.

Carandini, M. and Heeger, D. J. (2012). Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13:51–62.

Chapman, B., Stryker, M. P., and Bonhoeffer, T. (1996). Development of orientation preference maps in ferret primary visual cortex. *The Journal of Neuroscience*, 16(20):6443–6453.

Chemla, S. and Chavane, F. (2010a). A biophysical cortical column model to study the multi-component origin of the VSDI signal. *NeuroImage*, 53(2):420 – 438.

Chemla, S. and Chavane, F. (2010b). Voltage-sensitive dye imaging: Technique review and models. *Journal of Physiology-Paris*, 104(1-2):40–50.

Coppola, D. M., White, L. E., Fitzpatrick, D., and Purves, D. (1998). Unequal Representation of Cardinal and Oblique Contours in Ferret Visual Cortex. *Proceedings of the National Academy of Sciences*, 95(5):2621–2623.

Crair, M. C., Gillespie, D. C., and Stryker, M. P. (1998). The Role of Visual Experience in the Development of Columns in Cat Visual Cortex. *Science*, 279(5350):566–570.

DeAngelis, G. C., Ohzawa, I., and Freeman, R. D. (1995). Receptive-field dynamics in the central visual pathways. *Trends in Neurosciences*, 18(10):451 – 458.

Drummond, C. (2009). Replicability in not Reproducibility: Nor is it Good Science. In *The 4th workshop on Evaluation Methods for Machine Learning*, Montreal, Canada.

Elver, M. and Nagarajan, V. (2014). TSO-CC: Consistency directed cache coherence for TSO. In *HPCA*, pages 165–176.

Erwin, E., Obermayer, K., and Schulten, K. (1995). Models of orientation and ocular dominance columns in the visual cortex: a critical comparison. *Neural Computation*, 7(3):425–468.

Gallant, J. L., Connor, C. E., and Essen, D. C. V. (1998). Neural activity in areas V1, V2 and V4 during free viewing of natural scenes compared to controlled viewing. *Neuroreport*, 9(1):2153–2158.

Gent, I. P. (2013). The Recomputation Manifesto. *Computing Reasearch Repository*, abs/1304.3674:1–6.

Grinvald, A. and Hildesheim, R. (2004). VSDI: a new era in functional imaging of cortical dynamics. *Nature Reviews Neuroscience*, 5(11):874–885.

Gur, M., Kagan, I., and Snodderly, D. M. (2005). Orientation and Direction Selectivity of Neurons in V1 of Alert Monkeys: Functional Relationships and Laminar Distributions. *Cerebral Cortex*, 15(8):1207–1221.

Hebb, D. O. (1949). *The Organization of Behavior*. John Wiley, New York.

Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Hyvärinen, A., Hurri, J., and Hoyer, P. O. (2009). *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Springer Publishing Company, Incorporated, 1st edition.

Jonas, E. and Kording, K. (2016). Could a neuroscientist understand a microprocessor? *bioRxiv*.

Kang, K., Shelley, M., and Sompolinsky, H. (2003). Mexican hats and pinwheels in visual cortex. *Proceedings of the National Academy of Sciences*, 100(5):2848–2853.

Karns, C. M., Dow, M. W., and Neville, H. J. (2012). Altered Cross-Modal Processing in the Primary Auditory Cortex of Congenitally Deaf Adults: A Visual-Somatosensory fMRI Study with a Double-Flash Illusion. *Journal of Neuroscience*, 32(28):9626–9638.

Kaschube, M., Schnabel, M., Löwel, S., Coppola, D. M., White, L. E., and Wolf, F. (2010). Universality in the evolution of orientation columns in the visual cortex. *Science (New York, N.Y.)*, 330(6007):1113–1116.

Keil, W., Kaschube, M., Schnabel, M., Kisvarday, Z. F., Löwel, S., Coppola, D. M., White, L. E., and Wolf, F. (2012). Response to Comment on "Universality in the Evolution of Orientation Columns in the Visual Cortex". *Science*, 336(6080):413.

Keil, W. and Wolf, F. (2011). Coverage, continuity, and visual cortical architecture. *Neural Systems & Circuits*.

Knuth, D. E. (1984). Literate programming. *The Computer Journal*, 27:97–111.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Law, J. S. (2009). *Modeling the development of organization for orientation preference in primary visual cortex*. PhD thesis, The University of Edinburgh.

Löwel, S., Schmidt, K. E., Kim, D.-S., Wolf, F., Hoffsümmer, F., Singer, W., and Bonhoeffer, T. (1998). The Layout of Orientation and Ocular Dominance Domains in Area 17 of Strabismic Cats. *European Journal of Neuroscience*, 10(8):2629–2643.

Maunsell, J. H., Ghose, G. M., Assad, J. A., McAdams, C. J., Boudreau, C. E., and Noerager, B. D. (1999). Visual response latencies of magnocellular and parvocellular LGN neurons in macaque monkeys. *Visual Neuroscience*, 16:1–14.

Maunsell, J. H. and Gibson, J. R. (1992). Visual response latencies in striate cortex of the macaque monkey. *Journal of Neurophysiology*, 68(4):1332–1344.

Miikkulainen, R., Bednar, J., Choe, Y., and Sirosh, J. (2005). *Computational Maps in the Visual Cortex*. Springer.

Muller, L., Reynaud, A., Chavane, F., and Destexhe, A. (2014). The stimulus-evoked population response in visual cortex of awake monkey is a propagating wave. *Nature Communications*, 5.

Müller, T., Stetter, M., Hubener, M., Sengpiel, F., Bonhoeffer, T., Gödecke, I., Chapman, B., Löwel, S., and Obermayer, K. (2000). An Analysis of Orientation and Ocular Dominance Patterns in the Visual Cortex of Cats and Ferrets. *Neural Computation*, 12(11):2573–2595.

Nauhaus, I., Benucci, A., Carandini, M., and Ringach, D. L. (2008). Neuronal selectivity and local map structure in visual cortex. *Neuron*, 57(5):673–679.

Nijholt, B. and Akhmerov, A. R. (2015). Orbital effect of magnetic field on Majorana phase diagram (theory). *arXiv*.

Nowak, L. G., Munk, H. H. J., Girad, P., and Bullier, J. (1995). Visual latencies in areas V1 and V2 of the macaque monkey. *Visual Neuroscience*, 12:371–384.

Obermayer, K., Ritter, H., and Schulten, K. (1990). A principle for the formation of the spatial structure of cortical feature maps. *Proceedings of the National Academy of Sciences*, 87(21):8345–8349.

Ohki, K., Chung, S., Ch'ng, Y. H., Kara, P., and Reid, R. C. (2005). Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex. *Nature*, 433(7026):597–603.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609.

Orbach, H. S. and Cohen, L. B. (1983). Optical monitoring of activity from many areas of the in vitro salamander olfactory bulb: A new method for studying functional organization in the vertebrate central nervous system. *Journal of Neuroscience*, 3(11):2251–2262.

Paik, S.-B. and Ringach, D. L. (2011). Retinal origin of orientation maps in visual cortex. *Nature Neuroscience*, 14(7):919–925.

Pérez, F. and Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing in Science and Engineering*, 9(3):21–29.

Reynaud, A., Masson, G. S., and Chavane, F. (2012). Dynamics of Local Input Normalization Result from Balanced Short- and Long-Range Intracortical Interactions in Area V1. *The Journal of Neuroscience*, 32(36):12558 12569.

Ringach, D. L. (2002). Spatial Structure and Symmetry of Simple-Cell Receptive Fields in Macaque Primary Visual Cortex. *Journal of Neurophysiology*, 88(1):455–463.

Ringach, D. L., Shapley, R. M., and Hawken, M. J. (2002). Orientation Selectivity in Macaque V1: Diversity and Laminar Dependence. *Journal of Neuroscience*, 22(13):5639–5651.

Ritter, H., Martinetz, T., and Schulten, K. J. (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley.

Roe, A. W., Pallas, S. L., Kwon, Y. H., and Sur, M. (1992). Visual Projections Routed to the Auditory Pathway in Ferrets: Receptive Fields of Visual Neurons in Primary Auditory Cortex. *Journal of Neuroscience*, 12(9):3651–3664.

Rudiger, P. (2016). *Development and encoding of visual statistics in the primary visual cortex*. PhD thesis, The University of Edinburgh. In press.

Sato, T. K., Nauhaus, I., and Carandini, M. (2012). Traveling Waves in Visual Cortex. *Neuron*, 75(2):218–229.

Schottdorf, M., Keil, W., Coppola, D., White, L. E., and Wolf, F. (2015). Random Wiring, Ganglion Cell Mosaics, and the Functional Architecture of the Visual Cortex. *PLoS Computational Biology*, 11(11):1–40.

Schummers, J., Mariño, J., and Sur, M. (2004). Local networks in visual cortex and their influence on neuronal responses and dynamics. *Journal of Physiology-Paris*, 98(4-6):429–441.

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3:417–424.

Sirosh, J. and Miikkulainen, R. (1994). Cooperative Self-Organization Of Afferent And Lateral Connections In Cortical Maps. *Biological Cybernetics*, pages 66–78.

Sit, Y. F., Chen, Y., Geisler, W. S., Miikkulainen, R., and Seidemann, E. (2009). Complex Dynamics of V1 Population Responses Explained by a Simple Gain-Control Model. *Neuron*, 64:943956.

Snodderly, D. M. and Gur, M. (1995). Organization of striate cortex of alert, trained monkeys (Macaca fascicularis): ongoing activity, stimulus selectivity, and widths of receptive field activating regions. *Journal of Neurophysiology*, 74(5):2100–2125.

Solomon, S. G. and Lennie, P. (2007). The machinery of colour vision. *Nature Reviews. Neuroscience*, 8(4):276–286.

Stevens, C. F. (2000). Models are common; good theories are scarce. *Nature Neuroscience*, 3:1177.

Stevens, J.-L., Elver, M., and Bednar, J. A. (2013a). An automated and reproducible workflow for running and analysing neural simulations using Lancet and IPython Notebook. *Frontiers in Neuroinformatics*, 7:44.

Stevens, J.-L. R. (2011). A temporal model of neural activity and vsd response in the primary visual cortex. Master's thesis, School of Informatics.

Stevens, J.-L. R., Law, J. S., Antolik, J., and Bednar, J. A. (2013b). Mechanisms for Stable, Robust, and Adaptive Development of Orientation Maps in the Primary Visual Cortex. *Journal of Neuroscience*, 33:15747–15766.

Stevens, J.-L. R., Rudiger, P., and Bednar, J. A. (2015). HoloViews: Building Complex Visualizations Easily for Reproducible Science. In *Proc. of the 14th Python in Science Conference*.

Sur, M., Garraghty, P., and Roe, A. (1988). Experimentally induced visual projections into auditory thalamus and cortex. *Science*, 242(4884):1437–1441.

Tanaka, S., Tani, T., Ribot, J., O'Hashi, K., and Imamura, K. (2009). A Postnatal Critical Period for Orientation Plasticity in the Cat Visual Cortex. *PLoS ONE*, 4(4).

Tenner, V. T., de Dood, M. J. A., and van Exter, M. P. (2016). Measurement of the Phase and Intensity Profile of Surface Plasmon Laser Emission. *ACS Photonics*.

von der Malsburg, C. (1973). Self-Organization of Orientation-Sensitive Cells in the Striate Cortex. *Kybernetik*, 15:85–100.

Wolfram, S. (2003). *The Mathematica Book, Fifth Edition*. Wolfram Media, 5th edition.