Abteilung Neuroinformatik
Universität Ulm

Synchronization and pattern separation in spiking associative memories and visual cortical areas.

Dissertation zur Erlangung des Doktorgrades
Dr.rer.nat. der Fakultät für Informatik der Universität Ulm

von

Andreas Knoblauch

aus Geislingen an der Steige

2003

Amtierender Dekan : Prof. Dr. Friedrich von Henke
1. Gutachter : Prof. Dr. Günther Palm
2. Gutachter : Priv.-Doz. Dr. Friedrich Sommer
3. Gutachter : Prof. Dr. Ulrich Rückert
Tag der Promotion : 29. Oktober 2003

meiner Familie

Danksagung

An dieser Stelle möchte ich allen jenen danken, die zum Entstehen dieser Arbeit beigetragen haben.

Besonders verbunden fühle ich mich hierbei meinem Betreuer und langjährigen Lehrer Professor Dr. Günther Palm. Ihm danke ich für das Vertrauen und die freie Hand, die ich bei der Auswahl und Bearbeitung der behandelten Themen hatte, sowie für das große Interesse, das er meiner Arbeit entgegengebracht hat. Seine Kompetenz, sein hintergründiger Humor, seine Freundschaft und die stetige Bereitschaft zu anregenden Diskussionen waren eine reiche Quelle aus der ich Wissen und Motivation schöpfen konnte.

Dann möchte ich ganz herzlich Priv.-Doz. Dr. Fritz Sommer danken für die Freundschaft, produktive Zusammenarbeit und die vielen hintergründigen Ratschläge und Diskussionen welche mir den Einstieg in den Wissenschaftsbetrieb enorm erleichtert haben. Außerdem danke ich ihm für viele konstruktive Kritik und die Bereitschaft meine Arbeit zu begutachten.

Ein herzliches Dankeschön für die jahrelange Zusammenarbeit und Freundschaft möchte ich auch Professor Dr. Thomas Wennekers aussprechen. Ein wesentlicher Anteil der dieser Arbeit zugrundeliegenden Veröffentlichungen profitierte von seiner scharfsinnigen und kompetenten Kritik. Tatsächlich formten seine Vorarbeiten und seine Eigenschaft als Betreuer meiner Diplomarbeit wesentlich die Thematik der vorliegenden Dissertation. Insbesondere danke ich ihm für das von ihm entwickelte und mir zur Verfügung gestellte Simulationswerkzeug Felix welches die Grundlage der in dieser Arbeit durchgeführten Simulationen war. Besonders danken möchte ich ihm auch für das akribische Korrekturlesen dieser Arbeit und die konstruktive Kritik an Stil und einigen spekulativen Behauptungen. Die eventuelle Nicht-Beseitigung dieser Mängel gehen natürlich weiterhin zu Lasten der Sturheit des Autors.

Den Professoren Dr. Ulrich Rückert, Dr. Heiko Neumann, Dr. Friedrich von Henke und Dr. Enno Ohlebusch danke ich für ihre Bereitschaft zur Begutachtung meiner Arbeit und/oder für ihre Mitwirkung als Prüfer in meinem Promotionskolloquium. Ihre sehr positiven Urteile haben mir eine große Freude bereitet.

Für das Zustandekommen dieser Arbeit hat auch die freundliche und konstruktive Atmosphäre an der Abteilung Neuroinformatik beigetragen. Allen Kolleginnen und Kollegen während dieser Zeit sei herzlich für ihren Anteil daran gedankt, besonders meinen Mit-DoktorandInnen Marcus Borst, Michael Borth, Christian Dietrich und Rebecca Fay, den Assistenten und ehemaligen MitarbeiterInnen Pierre Bayerl, Dr. Andrea Bibbig, Dr. Friedhelm Schwenker, Priv.-Doz. Dr. Alfred Strey und Dr. Andreas Wichert, sowie unserer Sekretärin Birgit Lonsinger-Miller.

Die vorliegende Arbeit enstand im Rahmen verschiedener Gemeinschaftsprojekte zwischen der Abteilung Neuroinformatik der Universität Ulm unter Leitung von Professor Dr. Günther Palm, dem Centre for Hybrid Intelligent Systems der University of Sunderland unter Leitung von Professor Dr. Stefan Wermter, der Speech and Language Group am Medical Research Council Cambridge unter Leitung von Professor Dr. Friedemann Pulvermüller, dem Dipartimento di Neurofisiologia der Universita degli Studi di Parma unter Leitung von Professor Dr. Giacomo Rizzolatti, dem CORTEX-Team am INRIA Lorraine/LORIA-CNRS Nancy unter Leitung von Dr. Frederic Alexandre, der Arbeitsgruppe Neuroakustik der Technischen Universität Darmstadt unter der Leitung von Professor Dr. Gerald Langner, dem Speziallabor Nicht-Invasive Bildgebung am Leibniz-Institut für Neurobiologie in Magdeburg unter der Leitung von Professor Dr.

Henning Scheich, Dr. Birgit Gaschler-Markefski und Priv.-Doz. Dr. Peter Heil, und der Abteilung Akustik, Lernen, Sprache am Leibniz-Institut für Neurobiologie in Magdeburg unter der Leitung von Dr. Holger Schulze. Ich danke den jeweiligen Leitern für die Initiierung der Projekte und die einmalige Gelegenheit an den Projekten mitwirken zu dürfen. Insbesondere danke ich Professor Dr. Friedemann Pulvermüller, Dr. Olaf Hauck, Dr. Vittorio Gallese, Dr. Cornelius Weber, Dr. Christo Panchev, Wolfgang Schmidle, Mark Elshaw, Professor Dr. Gerald Langner und Michael Ochse für viele anregenden Debatten über neurobiologische und linguistische Sachverhalte sowie ihre Modellierung. Der Europäischen Union, der Stiftung Volkswagenwerk, sowie der DFG und dem Land Baden-Württemberg danke ich für die Finanzierung der Projekte und meiner Arbeit.

Zuletzt möchte ich meiner Familie und allen meinen Freunden für ihre emotionale Unterstützung danken. Besonders danke ich meiner Frau Kattia für ihre Liebe und Geduld trotz der vielen Arbeit. Meiner Schwester Stefanie danke ich weil ich ohne sie vielleicht nicht den Mut zu dieser Arbeit gehabt hätte. Schließlich empfinde ich meinen Eltern gegenüber tiefe Dankbarkeit für ihre kompromisslose Liebe und Unterstützung über all die Jahre hinweg.

Contents

1 Overview 1
1.1 Motivation and classification . . . 1
1.2 Organization . . . 2

2 Introduction 5
2.1 Neurobiology of the brain . . . 7
2.1.1 Anatomy of the brain . . . 7
2.1.2 Neurons, synapses, and Hebbian learning . . . 8
2.1.3 Architecture of the cerebral cortex . . . 9
2.1.4 Visual system . . . 11
2.2 Distributed representations, neural assemblies, and binding . . . 13
2.2.1 Distributed representations . . . 13
2.2.2 Neural assemblies . . . 14
2.2.3 The binding problem . . . 14
2.2.4 The temporal correlation hypothesis . . . 16
2.3 Associative memory . . . 19
2.3.1 The Willshaw model . . . 20
2.3.2 Neural associative memory and technical applications . . . 21
2.3.3 Willshaw associative memory as a model for a cortical column . . . 22
2.3.4 Alternative models of associative memory . . . 24
2.4 Simulation tools and models of neurons and synapses . . . 24

3 Willshaw associative memory 27
3.1 Classical analysis of the Willshaw model . . . 28
3.1.1 Storing and retrieving patterns . . . 28
3.1.2 Classical analysis of one-step retrieval . . . 29
3.2 Matrix compression yields storage capacity 1 . . . 30
3.2.1 Ultra-sparse patterns . . . 33
3.2.2 Moderately-sparse patterns . . . 34
3.2.3 Non-sparse patterns . . . 34
3.3 Fault tolerance . . . 35
3.3.1 Missing ones in the address patterns . . . 35
3.3.2 False ones in the address patterns . . . 38
3.4 Retrieval efficiency for sequential implementations . . . 39
3.4.1 Look-up-tables with $M = n^d$ . . . 39
3.4.2 Classical Willshaw model . . . 40
3.4.3 Compressed Willshaw model with ultra-sparse patterns . . . 40
3.4.4 Compressed Willshaw model with moderately-sparse patterns . . . 41
3.5 Retrieval efficiency for parallel implementations . . . 41
3.5.1 Classical Willshaw model . . . 42
3.5.2 Parallel implementation of the compressed Willshaw model . . . 42
3.5.3 Compressed Willshaw model with ultra-sparse patterns . . . 43

3.5.4 Compressed Willshaw model with moderately-sparse patterns 44
3.6 A critique of the classical binomial analysis 45
3.6.1 The binomial approximation of the Willshaw distribution can be bad 45
3.6.2 Oscillatory modulations in the membrane potential distribution 47
3.6.3 Underestimation of the variance of membrane potentials 48
3.6.4 Consequences 50
3.6.5 An experiment to test the hypothesis of cell assemblies in the brain 51
3.7 Implications 53
3.7.1 Implications for technical applications 53
3.7.2 Implications for biological models 54

4 Spiking associative memory **57**
4.1 Addressing with superpositions 57
4.2 Pattern separation 59
4.3 Spike counter model 60
4.4 Analysis of the spike counter model 63
4.4.1 Single noisy address patterns 63
4.4.2 Asymptotic analysis 65
4.4.3 Superpositions of several noisy address patterns 67
4.4.4 Relevance and applicability of the analysis 69
4.4.5 Refined analysis 71
4.5 Technical implementations of the spike counter model 72
4.5.1 Model extension 73
4.5.2 Sequential implementations 75
4.5.3 Parallel implementations 76
4.5.4 Simulation experiments 77
4.5.5 Summary 84
4.6 Biological implementations of the spike counter model 84
4.7 Implications 88
4.7.1 Summary of the chapter 88
4.7.2 Why spikes? 90

5 Model of visual cortical areas **95**
5.1 Modeling 96
5.1.1 Model of the primary visual cortex 96
5.1.2 Reciprocal connection to a central cortical area 97
5.2 Interaction between two cortical areas 98
5.2.1 Single stimulus 98
5.2.2 Multiple stimuli 99
5.3 Relation to neurophysiological results 102
5.3.1 Two-bars experiment 102
5.3.2 Inter-areal synchronization 103
5.3.3 Synchronization on larger time scales 105
5.3.4 Attention and biased competition 107
5.3.5 Synchronization on larger space scales 108
5.4 Summary of the mechanisms in the model 111
5.4.1 Local synchronization 111
5.4.2 Fast oscillations 111
5.4.3 Intra-areal long-range synchronization 111
5.4.4 Inter-areal long-range synchronization 112
5.4.5 Attentional switching between two activation states 112
5.4.6 Binding 113

6 Binding in the brain **115**
- 6.1 Is there a binding problem in the brain? . . . 115
- 6.2 Critique of the strong Temporal Correlation Hypothesis . . . 116
  - 6.2.1 Postulations of the strong TCH and phase coding . . . 116
  - 6.2.2 Critique of phase coding in the gamma range . . . 116
  - 6.2.3 Functional relevance of fine timing . . . 118
  - 6.2.4 Different time and space scales for binding by synchronization . . . 119
- 6.3 Sketch of a more global model of binding in cortical areas . . . 120
  - 6.3.1 Classification of different binding problems . . . 120
  - 6.3.2 Possible solutions . . . 120

7 Conclusions **123**
- 7.1 Contributions to the theory of Willshaw associative memory . . . 123
- 7.2 Contributions to the theory of cell assemblies . . . 125

8 Zusammenfassung (in German) **129**

A Information theory **135**
- A.1 Basic information theory . . . 135
  - A.1.1 Information of random variables . . . 135
  - A.1.2 Transinformation and transinformation rate . . . 135
  - A.1.3 Channels and channel capacity . . . 136
- A.2 Binary channels . . . 136
- A.3 Optimal compression of sparse bit streams . . . 137
  - A.3.1 Huffman codes . . . 137
  - A.3.2 Golomb codes . . . 137

B Analysis of associative memory **141**
- B.1 Information-theoretical storage capacity . . . 141
- B.2 Auto association and hetero association . . . 142
- B.3 Retrieval and separation quality . . . 143
  - B.3.1 Retrieval quality . . . 143
  - B.3.2 Separation quality . . . 143
- B.4 Further analysis of Willshaw associative memory . . . 144
  - B.4.1 Derivatives of memory load, storable patterns, and storage capacity . . . 144
  - B.4.2 Optimal storage capacity for moderately sparse patterns . . . 144
  - B.4.3 For non-sparse patterns asymptotic storage capacity is generally zero . . . 147

C Simulation tool Felix++ **149**
- C.1 Basic architecture of Felix++ . . . 149
  - C.1.1 The core modules of Felix++ . . . 149
  - C.1.2 Auxiliary modules of Felix++ . . . 151
  - C.1.3 Component classes of Felix++ . . . 152
- C.2 Simulation environment and components: Code examples . . . 154
  - C.2.1 Simulation environment: Class TSimulationEnvironment . . . 155
  - C.2.2 Components: Class TComponent . . . 156
  - C.2.3 Class TSSNeuron: a simple spiking neuron model . . . 157
- C.3 Structure of a Felix++ simulation . . . 159
  - C.3.1 A skeleton simulation program . . . 159
  - C.3.2 The parameter file . . . 161
  - C.3.3 Compiling and running simulations . . . 162

D Symbol reference **165**

Chapter 1

Overview

1.1 Motivation and classification

Throughout the last decades many representatives of computer science and in particular artificial intelligence promised that to the end of the twentieth century at the latest we will have computers and robots taking from us many of the so-called *primitive* cognitive works such as responding to telephone calls, driving a car, or cleaning the house. Actually up to date there are at best some quite primitive solutions to the primitive problems, such as artificial telephone-agents understanding “yes” or “no”, artificial car drivers that react reliably in about ninety percent of situations occurring when driving on the highway (but would cause accidents in the remaining ten percent), and service robots making every effort to navigate in a simple empty environment (such as the floors of a building) but have great difficulties in finding a cup of coffee on an office desk crowded with other distracting objects. The apparent underestimation of these primitive cognitive abilities of humans and animals is one motivation (among others) to investigate the basics of biological cognitive systems such as the neural networks in the brain.

This interdisciplinary work intends to contribute to several fields in computer science, computational neuroscience, neurophysiology and neuropsychology using methods of computer science, computational neuroscience, biomathematics, and neurophysics. The basic assumption is that the cognitive abilities of humans and animals result from information processing in the neural networks of the nervous system, and that main aspects of this process can be modeled and simulated using computers.

In close relation to the classical theory of Hebbian cell assemblies [64, 26, 130, 133] the concept of neural associative memory [204, 129, 69, 134, 29, 160, 171] is used to model local networks of the cerebral cortex. A large part of this work contributes to the theory of neural associative memory. In particular information theoretical methods are used in order to analyze neural associative memory with respect to storage capacity, fault tolerance, and retrieval efficiency. The analysis of storage capacity deals with the questions of how much information can be stored in the synaptic connectivity of a local cortical neuron population; analysis of fault tolerance deals with the decrease of storable information if the address patterns used for information retrieval deviate from the original ones used for storing the information; and analysis of retrieval efficiency deals which the questions of how fast information can be retrieved.

From the perspective of computer science neural associative memories are simply devices for rapid access of stored information, for example, the association of pattern vectors. There exist several alternative algorithms in computer science that can be used in principle instead of neural associative memory. For example, a look-up table could be used for storing in each line a pattern vector or the association of two pattern vectors. Alternatively, one could also use indexing and binary search if the address vector is scalar, or hash-tables if the address vectors used for retrieval are exactly the same as those used for storing, i.e., if no fault-tolerance is required. While the classical algorithms are superior in many of the classical applications, the use of neural associative memory can be advantageous if a large number of patterns have to be stored in a fault tolerant way.

One focus in this work is on the use of *spikes* for information processing in local and global neural networks. Common neuron models often use gradual (even linear) or sigmoidal output units. This is in contrast to the biological reality where neurons mainly communicate by pulse-like action potentials or *spikes*. While many classical models assume that the spike *rate* is sufficient for describing interactions between neurons, more recent results rather suggest that also the precise fine timing of individual spikes is important. Neurophysiological recordings indicate that *synchronization* of spike activity within the time window of a few milliseconds is involved in binding distributed neurons representing one common entity [41, 61, 168]. Another phenomenon observed in neurophysiological experiments are spatio-temporal spike patterns within a neuron group, so-called synfire-chains [2, 5], which can last tens to hundreds of milliseconds, and which repeat with a precision corresponding to the millisecond time window.

In this work the function of spikes are investigated on a local and a global level. The local level corresponds to local cortical networks or neural associative memory, and the focus is on the role of spikes in the separation of overlaying local representations. On the more global level of the visual system a model of reciprocally connected cortical areas is designed in order to investigate aspects of binding by spike-synchronization. In particular the representation, competition, and separation of multiple objects is investigated.

1.2 Organization

This work can roughly be divided into three parts: The first part (chapter 2) gives an introduction into the relevant fields, like neurobiology, assembly theory, and the theory of associative memory. The second part (chapters 3 and 4) contributes to the theory of neural associative memory. In the third part (chapters 5 and 6) a biologically more realistic model of interacting visual cortical areas is developed in order to investigate scene segmentation, pattern separation, and binding between distributed representations, and to compare the simulation results with neurophysiological observations. Finally chapter 7 summarizes and discusses the main results of this work.

Chapter 2 introduces the disciplines relevant for this work, like neurobiology, assembly theory, theory of neural associative memory, and the modeling and simulation of neurons, and discusses briefly related work.

Chapter 3 reviews and extends the classical theory of binary Willshaw associative memory [204, 129, 134]. It is shown that optimal compression of the binary memory matrix can yield a storage capacity of full 100% (normalized to the required physical memory), and that this can be achieved for a large range of coding parameters. Then aspects of fault tolerance and efficient sequential and parallel implementations are analyzed. Finally the validity of the classical binomial approximation of the Willshaw model is discussed using a refined analysis technique [29, 171]. It turns out that the binomial approximation can be qualitatively different to the exact Willshaw distribution even for very large networks. This difference suggests a neurophysiological experiment to test the hypothesis of local cell assemblies in the brain.

Chapter 4 investigates an extension of binary Willshaw associative memory using spiking neurons. From simple considerations of the separation of overlaying patterns the so-called spike counter model is derived as a continuous version of the Willshaw model. The theory of chapter 3 is extended for the spike counter model in order to analyze fault tolerance against false one-entries in the address patterns, or superpositions of several patterns, and the results of the analysis are compared to simulation experiments. Then a biologically more realistic model for a local cortical network is derived from the spike counter model using a more realistic neuron model, and different implementations of the model are compared in simulation experiments.

Chapter 5 develops a biologically realistic model of two reciprocally coupled visual cortical areas. The lower area P is modeled as the orientation selective subsystem of primary visual cortex V1, while the more central area C is modeled as the biological version of spiking associative memory as described in chapter 4 containing Hebbian learned representations of objects in the visual field.

In simulation experiments the model is tested using a single stimulus or a superposition of multiple stimuli, and the results are compared to various neurophysiological observations involving attentional effects and stimulus-specific spike synchronization in relation to visual feature binding [168]. In particular the results contradict common phase-coding models [150].

Chapter 6 discusses more generally possible solutions to the binding problem. A classification of different binding problems is given. For some of them the visual model of chapter 5 suggests already reasonable solutions contradicting commonly used phase-coding models. For other more difficult problems possible solutions are discussed in the context of the modeling in this work.

Chapter 7 summarizes and discusses the main results of this work.

Appendix A gives a brief introduction to the main concepts of information theory necessary for analyzing associative memory. In particular discussed are binary channels for the definition of storage capacity, and Golomb codes for efficient compression of the memory matrix.

Appendix B explains how information theory can be applied in order to analyze the storage capacity of Willshaw associative memory. Some quality measures are defined to judge performance of retrievals. And finally some proofs for propositions in chapter 3 are given.

Appendix C gives an overview over the simulation tool Felix++ used for the simulations in this work.

Appendix D defines some notational conventions for the use of symbols.

Some parts of the presented work have already been published in journal papers, book chapters, conference proceedings, or in other form. Results from chapter 3 have been published partially in [87], results from chapter 4 in [89, 90, 86, 95], and results from chapters 5 and 6 in [91, 92, 96, 89, 90, 86, 97, 98, 101, 93, 94] (in order of relevance).

How to read this work

Depending on background some readers may be interested only in parts of this interdisciplinary work. For the *computer scientist* mainly interested in analysis and efficient implementations of associative memory for technical applications it may be sufficient to read section 2.3, chapter 3, chapter 4, and section 7.1.

For the *biologist* interested in neural associative memory as a model for a local cortical network, assembly theory, and binding mechanisms in the brain, but not so much in the theoretical analysis of associative memory, it is recommended to read chapter 2, section 3.1.1, section 3.6.5, section 3.7.2, chapter 4 without section 4.4, and chapters 5 - 7.

Chapter 2

Introduction

Computer scientists contribute to society by creating machines that are supposed to relieve the burdens of our life and work. These burdens include stereotypic procedures like computing, accounting, and the administration of large data bases. Many of these problems are well-understood, and it is possible today to implement well-suited and cheap solutions.

However, there are still some problems apparently resisting appropriate algorithmic solutions. On the one hand one can formally prove for some of these problems that we cannot expect efficient solutions (e.g. the so-called traveling-salesman-problem, and other instances from the class of NP-hard problems; e.g. [159]). On the other hand there are many problems where we do not know if efficient solutions exist, and in many instances we are not even in the position to define the problem appropriately. Ironically many of these problems can be solved easily by humans, e.g. driving a car, or understanding natural language.

In contrast to the problems where efficient algorithmic solutions have already been found, many of the hard problems involve a much broader *context*. For example driving a car involves a myriad of different situations that have to be identified and that require special behaviors to cope with. And even worse, understanding natural language would practically involve the whole context of human life. In contrast, problems where efficient algorithms have been developed usually require only a very *narrow* context (e.g., playing chess, or administrating a data base), and can easily be formalized.

The fact that humans can tackle many of the hard problems (as language understanding and vision) together with the philosophical conviction that humans are biological organisms subjugated to the physical laws (and therefore potentially describable by formal models) has tempted many computer scientists to ally with biologists, psychologists, and other members of the cognitive neuroscience community to understand how the brain of humans (and animals) accomplishes its fabulous performance.

Apart from building intelligent machines a second and probably deeper motivation to investigate the brain origins from the philosophical questions who we are, where we have come from, and where we will leave to. Indeed, the question on the relation of the physical body and the mind occupied many scientists throughout the history of civilization (e.g., [49]). Almost three thousand years ago old Egyptian writers still favored the heart to be the organ responsible for sensation and thought. In the fifth century B.C. the Greeks first determined that the brain was the physical seat of the mind by observations of human patients with brain damage.

For another thousand years the brain itself was considered by many to be a mere package for the real substance of thought, the cerebrospinal fluid located in the ventricles (the present day use of the word *spirit* origins from this idea). This accorded quite well with the dualism of Christian theology, as the ventricles could be said to contain the soul without hypothesizing an identity between mind and the physical substrate of brain tissue.

In the seventeenth century, Rene Descartes proposed a simple sensory-motor loop, in which stimulated nerves caused release of spirits in the ventricles, which, in turn, caused efferent nerves and muscles to act. For intelligent human action, this loop was modulated by the soul via its effects on the pineal gland.

In the eighteenth and nineteenth centuries anatomical knowledge increased. From investigation of hundreds of skulls of humans (among them sadists and murderers) and beasts Franz Josef Gall developed his theory of cerebral localization or phrenology stating that basic human faculties (e.g., aggression, memory, vanity, religion, love, etc.) are located at separate skull regions. This more materialistic view was opposed by holistic theories denying the parcelation of the mind and the brain into distinct parts and instead emphasizing a unitary soul (e.g., Jean-Pierre Flourens).

In 1861 Paul Broca described a brain lesion of a late patient who had suffered from speech loss confirming earlier suggestions for the location of a speech center in the frontal lobe. In the following decade these results encouraged further investigations of the relation of cognitive deficits to brain lesions. Carl Wernicke discovered a second language area and proposed a model of representations in different sensory and language centers connected by specific pathways. This type of theorizing came to be known as *associationism* or *connectionism*, and was quickly extended to explain other disorders.

With the advent of computers in the 1940s it became possible to implement ideas of logicians and mathematicians how the mind may work. The new field now generally recognized as Artificial Intelligence (AI; e.g., [156]) developed from three sources: Knowledge of basic physiology and function of neurons in the brain; the formal analysis of propositional logic; and Turing's theory of computation. In particular the work of McCulloch and Pitts [117] who proposed a model of two-states (on/off) neurons can be seen as the common forerunner of both the logicist tradition in AI and the connectionist tradition. In 1943 they proposed a model of two-state neurons (on/off or 1/0) where a neuron enters the on-state (or 1-state) if it obtains sufficient input from neighboring neurons. They showed that any computable function could be computed by some network of connected neurons. In 1949 Donald Hebb contributed a neural learning rule that explained how such networks could be learned, and even more important, founded the theory of cell assemblies [64, 26, 130] intended to explain how thoughts are represented in the brain.

After Minsky's proof in 1969 that a single layer neural network cannot compute certain logical functions [119] the connectionist AI practically died out in the 1970s, while the early success of logicist AI led to enthusiasm and great expectations. However, the promises of logicist AI were not fulfilled in the following decades. While many ideas worked quite well for small closed worlds (e.g., block-world), the implemented systems bumped into complexity problems when faced with real-world-problems since, for example, no efficient algorithms are known to derive efficiently the truth of a proposition given a large knowledge base of facts and rules. Other severe problems are learning and the handling of uncertainty. Even for problems where nowadays computers are better than most humans, e.g., playing chess, it turned out that the strategies of computers are quite different from those of the humans. To find out the best move a chess program has to generate millions of alternatives most of which are completely irrelevant. In contrast, human players choose from only very few but very relevant alternatives by conscious and logical reasoning, where the set of alternatives seems to be the result of unconscious, intuitive, or associational processes.

After the return of neural networks in the 1980s the two disciplines, logicist and connectionist AI, have been seen each other rather as rivals, unfortunately. Representatives of the logicist (or *classical cognitive*) AI proposed that "... mind/brain architecture is not connectionist at the cognitive level [...] (but) may provide an account of the neural [...] structures in which classical cognitive architecture is implemented." [51], which states essentially that the neural machinery of the brain serves rather exclusively to implement conscious logical reasoning, and thus that human cognition can be described completely on the logical level of classical AI. This argumentation is based on the assumption that connectionist architecture cannot adequately implement composite structures in a dynamic fashion facing for example the problem of dynamic variable-binding.

In the 1980s and 1990s refined connectionist models were suggested that can handle the representation of rules, variables, and dynamical bindings, typically using temporal synchrony (e.g., [191, 165]) inspired by recent neurobiological findings of synchronized neural activity [41, 61, 168]. These models intend to be more than the neural substrate of the classical cognitive ideas. For example Shastri and Ajjanagadde [165] distinguish between *reflexive* and *reflective* reasoning. While the latter corresponds to the classical cognitive ideas requiring " reflection, conscious deliberation, and often an overt consideration of alternatives and weighing of possibilities ", the model intends to implement rather *reflexive* reasoning accounting for the human ability to " draw a variety of inferences rapidly, spontaneously, and without conscious effort " (similarly to the human chess player obtaining the relevant alternatives for a move, see above). Although inspired by neurophysiological observations these models are typically far from the reality of the brain, however.

In this work we will discuss the relation between neural architectures and different versions of the binding problem [192, 168]. Starting from simple abstract models for neural associative memory [204, 129, 69, 134] and neural assemblies [64, 26, 130, 133] we will develop a biological realistic model using spiking neurons that reproduces many neurophysiological observations, and that solves very naturally some instances of the binding problem. In chapter 6 binding is discussed more generally in relation to the ongoing debate between connectionist tradition and the tradition of classical cognitive logicists AI. The rest of this chapter introduces into the basics of the relevant fields like neurobiology, assembly theory, and the theory of associative memory, necessary for understanding this work.

2.1 Neurobiology of the brain

Today it is commonly believed that the main purpose of the *nervous system* is the information processing and communication between different parts of the human body. We can distinguish the *vegetative* and *animalic* nervous systems. While the *vegetative nervous system* regulates the internal milieu of the organism via the two antagonists sympathicus and parasympathicus, the *animalic nervous system* serves to coordinate interactions with the external environment. The latter means generation of sensory information, integration to conscious percepts, and generation of adequate motor output. The animalic nervous system can be further divided into *central nervous system* (brain and spinal cord) and *peripheral nervous system* (nerves of head, trunk, and extremities).

It is relatively easy to investigate the peripheral nervous system in physiological experiments. For example, the response measured from a sensory nerve will be closely related to a stimulus in the environment which can be easily controlled in an experiment. Similarly the reaction of a muscle will be closely related to the stimulation of a motor nerve.

In contrast it is generally much more difficult to investigate functional aspects of the central nervous system since here responses will not only depend on the stimulus but also on *internal states* (e.g., states of memory, expectation, attention, motivation, etc.) which can be very hard to control.

This work intends to make a contribution to a reasonable theory of the organization of internal states of the brain [64, 26, 130].

2.1.1 Anatomy of the brain

The brain can be roughly divided into six parts (see Figure 2.1). The (1) *medulla oblongata* is the transition region to the spinal cord and is dedicated to elementary vital functions such as control of digestion, breathing, and control of the heart beat. The (2) *pons* deals with interchange of motion-related information between the cerebral cortex and the cerebellum. The (3) *cerebellum* regulates strength and extension of motor actions, and plays a role for learning of motor capabilities. The (4) *mesencephalon* controls elementary sensory and motor functions, such as eye-movements, and the coordination of visual and auditory reflexes. The (5) *diencephalon* lies between the two hemispheres of the cerebrum and contains the thalamus and the hypothalamus. The thalamus constitutes the interface between the cerebral cortex and the remaining parts of the central nervous system, and the hypothalamus regulates autonomous, endocrine and viscerale functions. Finally the (6) *cerebrum* consists of the cerebral cortex and below three additional structures: The basal ganglia (control of motor activity), the hippocampus (memory), and the amygdala (emotions, motivation and reward).

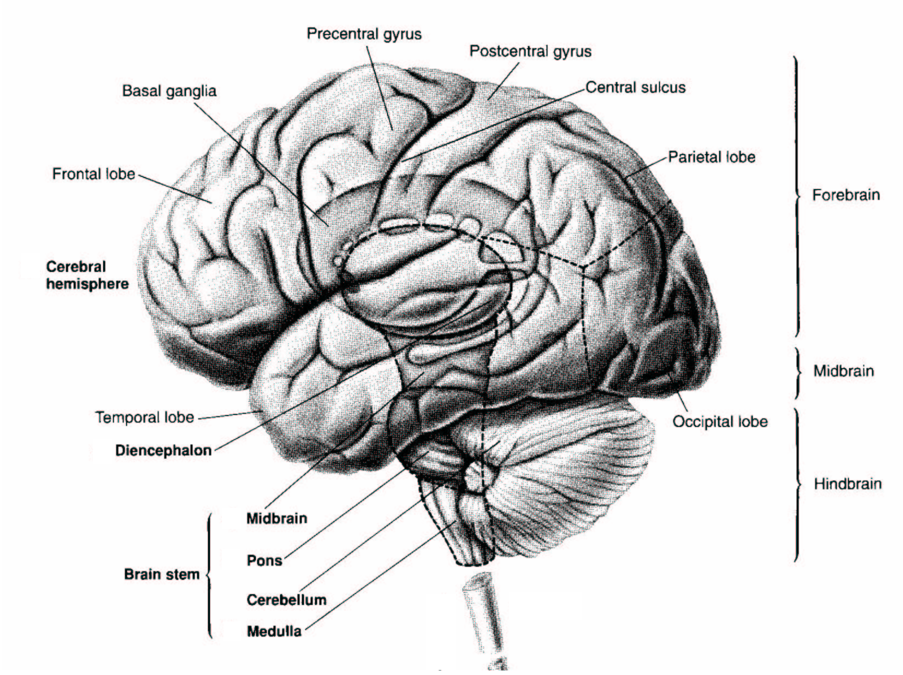


Figure 2.1: Illustration of the human brain (Modified from [79].)

2.1.2 Neurons, synapses, and Hebbian learning

There are two cell types in the nervous system: Neurons and glia. A typical *neuron* can be divided into four morphologically defined regions (see Fig.2.2): (1) The *soma* is the metabolic center of the cell. It gives rise to two types of processes called dendrites and axons. The (2) *dendrites* are the 'sensors' of the neuron branching out in a tree-like fashion. They serve to receive signals from other neurons. While a neuron usually has many dendrites, only a single (3) *axon* origins from the *axon hillock*, a specialized region of the cell soma. It serves to convey information from the neuron to other distant cells. The axon can make complex arborizations and run to different distant locations where it makes (4) *synapses* on the target cells.

In this work we will simulate the electrophysiology of a large number of neurons. For this we use simple mathematical models for the electrical properties of individual neurons (see section 2.4). They can be explained as follows. Between the soma and the extra-cellular space there is an electrical potential, the so-called *transmembrane potential*: In the resting state of the neuron the voltage difference between the inner and outer medium is about -60mV (i.e., negative inside the cell). Synaptic input from other cells generate *postsynaptic potentials* (PSPs) in the dendrite at the synaptic location which are electrotonically transmitted to the soma. If the cell receives many excitatory PSPs within a small time window (e.g., 10msec) then they will superpose additively. If the depolarization at the axon hillock exceeds a critical value (firing threshold, roughly -50mV) then a *spike* is generated and transmitted along the axon to the synapses on target neurons.

It is believed that synapses play an important role for long-term memory. In 1949 Donald Hebb proposed his theory of neural assemblies [64] postulating the strengthening of synapses between coactivated neurons (see also [76]). Since then it has been confirmed in many neurophysiological experiments that simultaneous activation of two neurons can strengthen the efficacy (i.e., the PSP amplitudes) of the synapses connecting the two neurons [19, 1]. Recent results

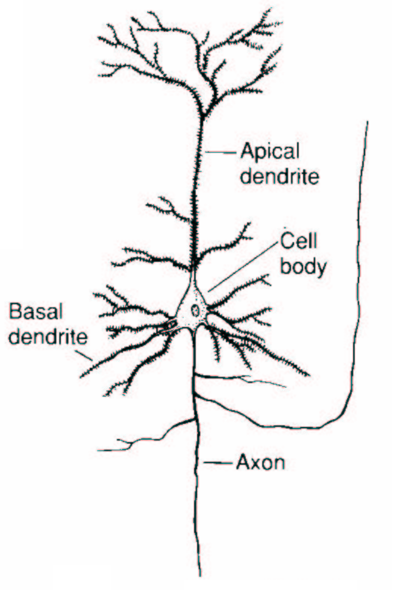


Figure 2.2: Illustration of a cortical pyramid neuron. (Modified from [80].)

even suggest that a synapse is specifically strengthened if the presynaptic spike precedes the postsynaptic spike [112, 15, 53].

In addition to the neurons there are about 10-50 times more *glial cells* [80]. Despite their large number it is believed that glial cells play no direct role in the fast (electrical) transmission of signals. This seems to be exclusively the domain of the neurons. Instead it is believed that glial cells serve rather as supporting elements, for example in order to provide firmness and structure to the brain, regulate the concentration of ions, isolate axons to accelerate transmission of signals (myelin sheath), or remove debris after injury or neuronal death.

2.1.3 Architecture of the cerebral cortex

The cerebral cortex is the *gray matter* below the surface of the hemispheres of the cerebrum. It contains the cell bodies, the dendrites, and local axons of the neurons. Below the cortex we find the *white matter* constituted by the myelinated axons connecting cortical areas with distant cortical or sub-cortical regions.

The architecture of cortex is remarkably constant between different species of different size all sharing the same *layered* architecture and pattern of *afferents* (input fibers) and *efferents* (output fibers). Most amazing is the constance of *synaptic density* ($\approx 0.5 - 1 \cdot 10^9$ synapses/mm^3) over different animals and brain regions (cf. Table 1.5.4. in [3]) suggesting that the limiting factor for local cortical connectivity is the packing density of the synapses. Also the *thickness* of cortex is remarkably constant between different species. For example, the mouse cortex has a thickness of about 0.8mm while the thickness of human or elephant cortex is about 2.5mm or 4mm (although the weight of a human or an elephant brain is about 3400 or 13000 times the weight of a mouse brain; cf. Fig. 1.1.3 in [3]; cf. [83]). There is a tendency for higher *neuron densities* in smaller animals. For example the cell density in mouse, human, or elephant motor cortex has been estimated to be about 143, 10.5, or 7 $\cdot 10^3/\text{mm}^3$ where the data varies significantly between investigators (factor 1-3), cortical regions (factor 3 for visual cortex), and cortical layers (cf. Table 1.5.1 in [3]). It has been claimed that the *number of cells* below a small patch of cortex surface is quite constant (e.g. 10^5 per 1mm^2) in all mammals where the lower cell density would compensate for the thicker cortices in larger animals [152, 27, 3]. Another functional relevant parameter is the *speed of axonal transmission* which depends mainly on the diameter of the axon and if the axonal sheath is myelinated (by glia cells) or not. The axonal transmission speed can vary over several orders of magnitude for different neurons even within one local cortical region of the same species [179]. Since larger cells (corresponding to a lower cell density) are assumed to have also thicker axons this may also compensate to a certain degree for the larger distances in larger animals. Despite the described differences it is reasonable to assume that all cortical regions in all animals process information locally according to the same principles.

It is believed that all the synapses of a given cortical neuron release the same neurotransmitters (Dale's principle) [36, 39]. In particular a given neuron can make either excitatory or inhibitory connections, but not both types. This seems to be a fact not considered by certain cortex models [69, 150] (see also sections 2.3.3 and 2.3.4). It is commonly accepted that about $80-90\%$ of the cortical neurons are excitatory, and only $10-20\%$ are inhibitory. Most of the excitatory cells are pyramidal neurons characterized by the typical shape of the soma and the two types of basal and apical dendrites (Fig. 2.2). The apical dendrites ascend perpendicular to the cortex surface towards the upper layers, while the basal dendrites arborize in the local vicinity of the soma. Indeed it is due to the parallel arrangement of thousands of apical dendrites that we can measure electrical potentials (e.g., EEG) above the cortical surface [120] (in less regular arrangement the individual electrical fields would rather average out). Similar to the apical dendrite the axon of a pyramidal cell typically descends perpendicular to the cortex surface down to deep layers and then leaves cortex into the white matter while also collaterals of the axon branch in the local vicinity of the neuron. An important consequence of the apical architecture is that a pyramid cell can be connected to all other neurons above and below, i.e., within the so-called *cortical column*. Moving in parallel to the cortex surface to other neurons the probability of a connection to the first neuron decreases continually resulting from the limited extension of dendritic and axonal collaterals reaching only a few millimeters [65, 27].

Along its depth perpendicular to the surface the cortex shows a layered architecture. Usually six layers can be distinguished in cytological preparations which show differences in cell density, cell and fiber types, etc. Although the layered structure is evident there is no complete isolation of the neurons in the individual layers. Rather the axons and dendrites of individual cells make rich arborizations and connections throughout the local cortex. At least two observations support the view that the separation of the neuron populations in the different layers should not be overestimated, i.e., that the layers are rather symmetrically interconnected. First, by local electric stimulation usually the *whole* cortical column responds [181, 78]. Second, for fast cortical oscillations (e.g. at 50Hz) spikes seem to be synchronized over the whole cortical depth [41, 175, 176]. If the inter-layer connections were asymmetric or even uni-directional one would rather expect systematic shifts in spike correlograms given the significant transmission delays between upper and lower layers [179].

A local cortical neuron population receives afferents from subcortical (e.g. thalamic) regions and also from other cortical locations of the ipsilateral hemisphere or via the corpus callosum from the contralateral hemisphere. It is known that most of the cortico-cortical connections are reciprocal or bidirectional. I.e., if there is a connection from area A to area B, then it is very likely that there exists also a connection from area B to area A. Tracer studies revealed that these connections are rather patchy where one local cortical patch (diameter about 1mm) is connected to $5-10$ other distant cortical patches [182, 27]. An important observation is that almost exclusively excitatory pyramidal cells contribute to the long-range cortico-cortical connections while inhibitory neurons make only local connections [182, 27].

With the argumentation in this section it is justified to think of a prototypical essentially two-dimensional cortex architecture as a first approximation. We can argue that local connections between any two neurons are possible if they are below the same square millimeter of cortical surface. Below a square millimeter cortex we can assume about 100000 neurons where most of them are pyramidal cells each receiving about 10000 synapses from other local neurons, and another 10000 synapses from 5-10 more distant cortical or sub-cortical locations [3, 27]. In section 2.3.3 we will introduce neural associative memory as a model for a small patch of cortex based on the parameters and arguments derived in this section.

2.1.4 Visual system

Figure 2.3 illustrates the visual system in humans. In the eye sensory cells of the *retina* (the so-called rods and cones) translate brightness and color information into electric potentials. From the retinal ganglion cells the information is conducted through the optical nerve to the chiasm where the nerves from the two eyes cross. Here axons representing information from the left and right visual hemifield separate: Information about the left visual hemifield (from the right parts of the retinae of the two eyes) is processed in the right hemisphere of the brain, whereas information about the right visual hemifield (from the left parts of the retinae) is processed in the left hemisphere. From the chiasm the optical nerves run via the optical tract to the LGNs (lateral geniculate nucleus) in the thalamus, and from there via the optical radiation to the primary visual cortex V1.

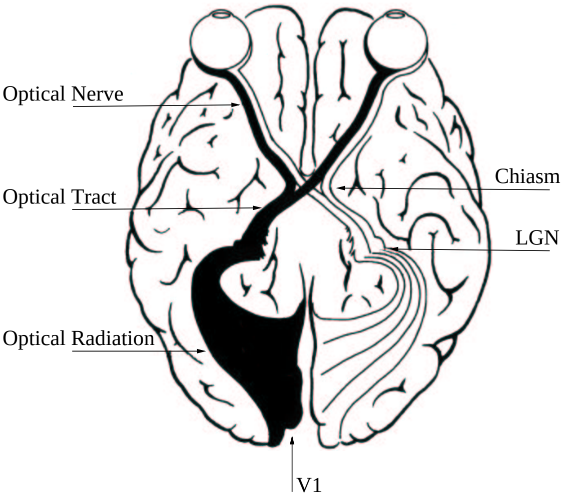


Figure 2.3: Diagram of the retino-geniculo-cortical pathway in the visual system of a higher mammal. The brain is viewed from below. (Modified from [184].)

Figure 2.4 shows schematically the architecture of primary visual cortex. In cell preparations of slices orthogonal to the cortex surface one can distinguish the six cortical layers (see section 2.1.3). Layer IV is much thicker than in other cortical areas and can be subdivided into further sublayers. Here the fibers from the LGN make *topographically* ordered synaptic connections onto cortical pyramids and spiny stellates (and also inhibitory cells). The input fibers from the left and right eyes are separated from each other which results in the *ocular dominance bands* of layer IV (width about 0.4mm) [73].

Of particular importance for the modeling in this work are the modularly organized orientation columns: Many cells in primary visual cortex show a preference for oriented stimuli (for example bars). All the cells in an orientation column or microcolumn (i.e., the volume below a small patch of cortex surface with perhaps diameter 20-50μm), have preference for the same orientation. For tangential penetration through the primary visual cortex (in parallel to the cortex surface) the orientation preference changes in possibly discrete but small and mostly ordered steps (e.g. 0°, 10°, 20°, ...) [73, 184]. Imaging studies of larger (two-dimensional) parts of primary visual cortex revealed a pinwheel-like architecture, where the orientation preference changes circularly around discontinuities, the so-called *pinwheels* (see Fig.5.2; [73, 20, 21, 13]).

A larger patch of primary visual cortex comprising all possible orientations is also called a *hypercolumn* (diameter about 0.4-0.5mm; corresponds to the width of the ocular dominance band) [73, 116, 184, 68]. The hypercolumns are arranged modularly and within a module there are the so-called *blobs* (diameter about 0.2mm) which can be seen in cytochrome oxidase preparations indicating regions with strong metabolic activity. The blobs are related to *three processing streams* which already originate in the retina and LGN, and enter the primary cortex in separate

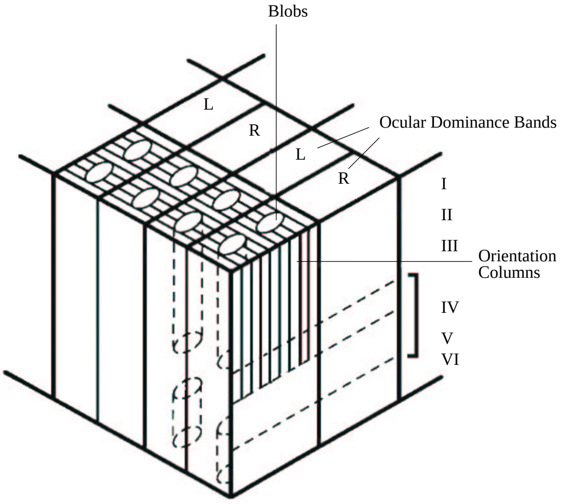


Figure 2.4: Architecture of the primary visual cortex, the so-called ice-cube model. (Modified from [184].)

sub-layers of layer IV [81, 58]. The so-called *magno-cellular system* (M-System) corresponds to relatively large cells and is specialized for the identification of movement and spatial relations. The so-called *parvo-cellular system* (P-System) consists of relatively small cells and can be further sub-divided into two sub-systems: The *P-blob-system* lies within the blobs of V1, and deals mainly with the processing of color information, whereas the cells of the *P-interblob-system* lie between the blobs and deal mainly with the processing of form. It is important to note that the described separation of the three streams is rather functional than anatomical. In particular cells from one stream can still make many synaptic connections onto cells of the other streams.

There exists a large number of further visual cortical areas, and attempts have been made to find a hierarchical order [50, 116, 81]. Many visual cortical areas show more or less a topographical organization similar to primary visual cortex. The ascent in the hierarchy is usually accompanied by increased *receptive field sizes*. The classical *receptive field* (RF) of a cell is the area in the visual field where stimulation can evoke a response in the cell. Using simple stimuli (e.g., small filled circles) one can determine the RF of a cell. It is known that cells in primary visual cortex have relatively small RFs and respond to simple stimuli like oriented bars. There is a precise topographical relation between location of the RF and location of the cell in V1, where a large area of V1 processes the central part of the visual field at a high resolution, while the peripheral parts are processed at lower resolution [162]. For higher visual cortical areas the topographical relation is less precise, the RF size is larger, and the cells have often a preference for more complex stimuli. For example there are cells in IT (infero-temporal cortex, see Fig.2.1) which respond selectively to faces independently of the location of the face in the visual field [37].

Each of the diverse visual cortical areas in the hierarchy seems to process a special kind of information. Thus the separation of different processing streams already present in V1 (M-, P-blob, P-inter-blob systems) is a general principle of the visual system. Best known is the separation of a dorsal ’where’-stream (actually originating from the M-system in V1) dealing with perception of motion and location, and a ventral ’what’-stream (originating from the P-systems) dealing rather with object recognition and visual memory.

Thus visual information is processed in a distributed and hierarchically organized manner. Information related to the same entity can therefore be distributed over different distant cortical areas, but also distributed in the same topographically organized area (like V1). The questions arises how these distributed pieces of information can constitute our apparently unified perceptual and conscious states, i.e., how the brain can solve the so-called *binding problem* of visual features [191, 192, 168].

The idea of distributed neural representations in the brain, and the complex of the binding problem was one of the main motivations for this work. The notion of binding in the brain and different variants of the binding problem are introduced in the next section.

2.2 Distributed representations, neural assemblies, and binding

2.2.1 Distributed representations

In the previous section we have seen that the visual system is roughly organized in a hierarchical and distributed manner. Primary visual cortical areas are large, organized retinotopically or topographically (with respect to the visual field), and contain maps of elementary features (like orientations, contours, local motion direction and speed, etc.). The individual neurons exhibit only strong spike responses if a matching stimulus is present within a small area in the visual field, i.e., in the receptive field (RF) of the cell. In contrast, neurons in higher visual cortical areas have larger RFs, and can respond to more complex stimuli (e.g., faces in IT) where the response is rather independent from the exact stimulus location [37]. This corresponds to a separation of 'what' and 'where' information (see section 2.1.4) and leads to the question how the distributed pieces of information could be put together to a unified percept.

Neuropsychological as well as anatomical and neurophysiological observations suggest that a distributed and hierarchical organization is not restricted to the visual system, but constitutes rather a general organization principle of the brain (cf. [50, 47]; see also section 6.1).

Neuronal representations are distributed in a trivial sense because the elements of the representations, i.e., the neurons, are distributed over the brain. If we define, for example, the representation of an object as the set of neurons that are activated when the object is present in the visual field, then of course more than one neuron will get activated along the visual path from the retina via LGN to the different areas of the cortex.

A popular notion of neural representation that is not really distributed in a non-trivial sense is that of a *feature pyramid* (cf. [197]). With the example of a visually perceived object elementary features (e.g., pixels) of the object would be represented in a large map at the base of the pyramid. In several processing steps the base map would be transformed to further maps representing features of increasing complexity (e.g., contours, edges, complex configurations of lines, etc.). One can have the notion of a pyramid because the number of *active* neurons in the maps decreases with increasing transformation level. In the extreme case a single cell (often called *cardinal* or *grandmother neuron*; cf. [12, 158, 128]) would be activated at the top of the pyramid. This kind of representation is not really distributed because the representation at each level contains all relevant information of the levels below, i.e., if level i gets activated level $i-1$ (and below) becomes irrelevant. This way of representing is exceedingly inefficient and therefore improbable to occur in the real brain because the same information would be unnecessarily represented redundantly at many different levels. Although the number of *active* cells may decrease with increasing level of the pyramid, the *total* number of cells necessary to represent all special configurations explodes in a combinatorial catastrophe.

In contrast, a really distributed representation would process different *feature dimensions* (like form, color, motion, depth, and location) in different streams, similar to the 'where' and 'what' streams of the visual system, and therefore avoid the combinatory explosion. For example, we can recognize objects in the visual field with arbitrary colors (we would have no problem with recognizing a blue cow), and while combinatory coding each of n forms and m colors would require $n \times m$ representations, we require only $n + m$ representations if the feature dimensions form and color are processed in separate streams (but also an additional mechanism for binding if more than one object is represented at the same time; see sections 2.2.3 and 6.1). Such an architecture would exhibit a separate pyramid for each feature dimension.

2.2.2 Neural assemblies

The described observations and arguments support ideas initially postulated by Donald Hebb [64] (but see also [76]) that objects and thoughts are represented in the cortex of complex animals such as monkeys and humans by the coincident activation of a group of distributed neurons called a cell assembly [26, 130, 192, 133, 3]. The idea is appealing because the formation of a cell assembly can easily be explained by the assumption of a simple synaptic learning rule: the Hebbian rule (cf. section 2.1.2) postulates that a synaptic connection between two neurons is strengthened if both neurons are active at the same time. Thus, if a group of neurons gets repeatedly activated by a specific stimulus, all the synapses connecting the neurons in the group will be strengthened, and thereby a representation of the stimulus (i.e., the cell assembly) evolves.

A cell assembly representing, for example, a cow will presumably be distributed over many cortical areas [136, 135, 143] corresponding to the visual, auditory, language (in humans), olfactory, and further modalities, and would also be distributed over the postulated different processing streams within each modality. On the one hand the particular postulate of assembly theory is the representation of objects and thoughts in *discrete* activation states (i.e., the coincident activation of the neurons constituting the cell assembly), on the other hand it is important to note that there is probably not a single cow assembly in the brain that gets *completely* activated whenever the subject sees, hears, thinks of, or smells a cow, but that "the" cow assembly is rather a *collection* of stable activation states that have probably large overlaps but that differentiate between different occurrences of a cow, for example in the visual modality to account for different views of a cow (frontal, from behind, etc.).

This corresponds also to the fact that it is anatomically impossible to have synaptic connections between all neurons of a global assembly (see section 2.1.3). Even for two local neurons (not necessarily belonging to the same assembly) below the same square millimeter of cortex surface the chance of a connection is only about ten percent [27, 65]. And for more distant neurons synaptic links can only exist along the patchy cortico-cortical connections. This means that a global assembly extending over two anatomically separated processing streams (e.g., 'what' vs. 'where') is divided into two rather independent parts in the two streams that may only be connected with each other at the locations of convergence or divergence of the streams. In contrast, neighboring neurons of a global assembly (e.g., below the same square millimeter) are very likely to be *directly* connected by synapses.

Thus it is important to differentiate between *global* and *local* cell assemblies. Local cell assemblies are neuron groups in the same cortical column that are highly interconnected. In particular we can assume that the chance of a synapse between two neurons of a local assembly is much higher than the average connection probability of ten percent, for example fifty percent. We can use neural associative memory as a model for local cell assemblies (see section 2.3.3) where for analysis usually a *complete* connection within local assemblies is assumed, i.e., a neuron of an assembly makes synapses on all other neurons of the same assembly. From the analysis of neural associative memory we know another argument for *locally* distributed representations. By using n neurons we can represent almost n^2 independent entities if the representations are sparse, i.e., if the cell assemblies consist only of few of the n neurons. This is analyzed and discussed in chapter 3. In the following we will address a severe problem occurring for distributed representations.

2.2.3 The binding problem

One immediate argument against the idea of distributed representations and cell assemblies which has been put forward many times is the *binding* or *superposition problem*: if two or more assemblies are activated at the same time, how can they be segmented into the individual assemblies? One solution to this problem is provided by the strong mutual excitation between the neurons belonging to the same *local* assembly, probably acquired by Hebbian learning or auto-association. This has been worked out in several rather theoretical papers [204, 26, 129, 130, 69, 131, 133] which essentially show that by controlling the total activity within a certain region of the cortex, for example by unspecific inhibition, the superposition problem, i.e. the activation of two assemblies at the same time, can be avoided. Furthermore this can be achieved even when a very large number of assemblies (sparse activation patterns) are stored by auto-association.

The term "binding problem" as used in the literature (e.g., [154, 188, 191, 192, 51, 165, 150, 193, 168, 199, 187, 194, 148, 56, 163, 167, 59, 147, 205]) actually corresponds to a whole family of related problems. The solution described above guarantees that local superpositions of activation patterns can be segregated into its components, i.e., into single local assemblies. On a more global level another question is how local assemblies (e.g., representing features in a large topographical cortical area) are coordinated to a whole. If we look at Figure 2.5 for the first time the picture contains apparently nothing than a chaos of unrelated black patches. However, after looking at the picture for some time (e.g., two minutes or so) our brain can generate an interpretation of the apparent chaos. Suddenly we can recognize the dalmatian dog hidden in the chaos of black patches. This corresponds to the binding problem of perception or Gestalt psychology [151]: How are the local features bound to a coherent percept?

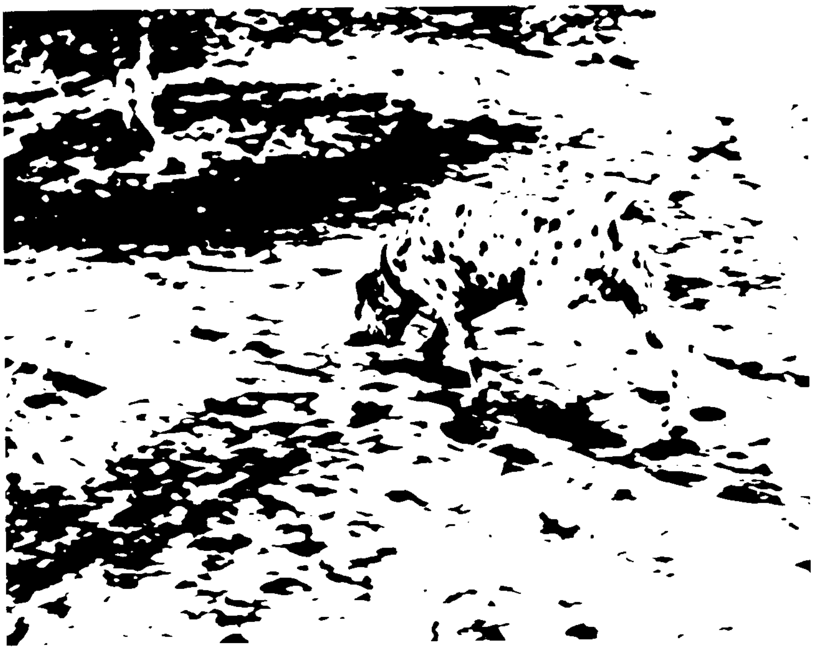


Figure 2.5: Illustration of the feature binding problem. There is a dalmatian dog hidden in the chaos of black patches. (Modified from [75].)

A related version of the feature binding problem is the question what happens if a picture allows more than one interpretation. For example, Figure 2.6 can be interpreted either as an old or a young woman. Interestingly, we cannot see both interpretations at the same time. Instead, the interpretation switches on a time scale of several seconds [139], presumably due to neural habituation mechanisms [46]. This instructive example gives us two hints: First, binding is a dynamical process: the *same* stimulus can cause binding of *different* feature configurations. Second, representation of multiple interpretations (a special case of the superposition problem described above) avoids incongruent superpositions by using the temporal dimension. This leads to the idea that the temporal domain plays a general role in solving the different binding problems. Below we will discuss the so-called *temporal correlation hypothesis* postulating that binding in the brain is accomplished by synchronization of neural activity [191, 192].

Figure 2.7 is a more extreme example for multiple interpretations. It is composed of small black dots arranged in such a way that we can perceive a very large number of different circles of different sizes. If we let our glance glide from reading-distance over the picture we are attracted by a vast number of overlapping circles popping out of the figure. If we fixate one of the points

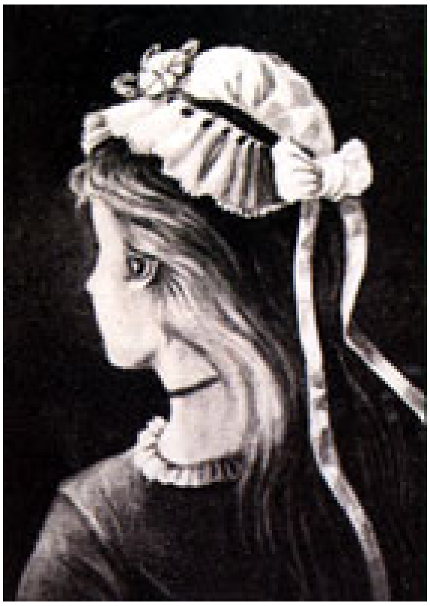


Figure 2.6: A variant of the "Young-Girl-Old Woman" illusion also known as "My Wife and My Mother-In-Law". (See [22].)

our attention is distracted quickly (perhaps every tenth of a second) from one bubbling circle to the next. This points to the involvement of active (i.e., top-down) grouping processes in the brain. I.e., the binding of local features represented in primary cortical areas is actively guided by expectations or Gestalts presumably represented in higher cortical areas. You can experience top-down grouping also when looking to Figure 2.5 for a second time: now you can probably recognize the dalmatian dog almost immediately. Obviously a new representation has been allocated somewhere in our brain when recognizing the dalmatian dog for the first time, which is used as a pattern to accelerate binding of the features when looking again at the picture. These observations motivate the design of a neural model of two reciprocally connected visual cortical areas described in chapter 5 which is used to investigate feature binding (cf. also [89, 90, 91, 92]).

Note that so far we have not addressed problems that will occur for really distributed representations as discussed in section 2.2.1. We argued that independent processing streams for different feature dimensions avoid a combinatory explosion of the number of necessary representations. For example only n+m instead of $n \cdot m$ representations are necessary for representing n forms and m colors. However, as soon as we represent multiple objects at the same time, we have the problem of binding across different feature dimensions. For example if a scene contains a green hat and a red book the coordination of which form belongs to which color would be lost if form and color are processed independently of each other. Thus we need an additional mechanism to bind across different feature dimensions.

An even more severe problem occurs if *dynamic binding* is required between different entities. This occurs for example if one tries to understand the sentence "Tom beats Fritz". This involves the representation of a 'beater' and a 'victim', but no prejudicial bias of the role of Tom and Fritz. Here the binding problem is to assign these roles dynamically to Tom and Fritz. This kind of binding problem is closely related to language and logic, and can occur in arbitrarily nasty forms, involving for example n-ary predicates and multiple instantiations [51, 165].

2.2.4 The temporal correlation hypothesis

As a possible solution to the binding problem the *temporal correlation hypothesis* (TCH) has been suggested [191, 192]. The TCH postulates that binding is accomplished by synchronization of

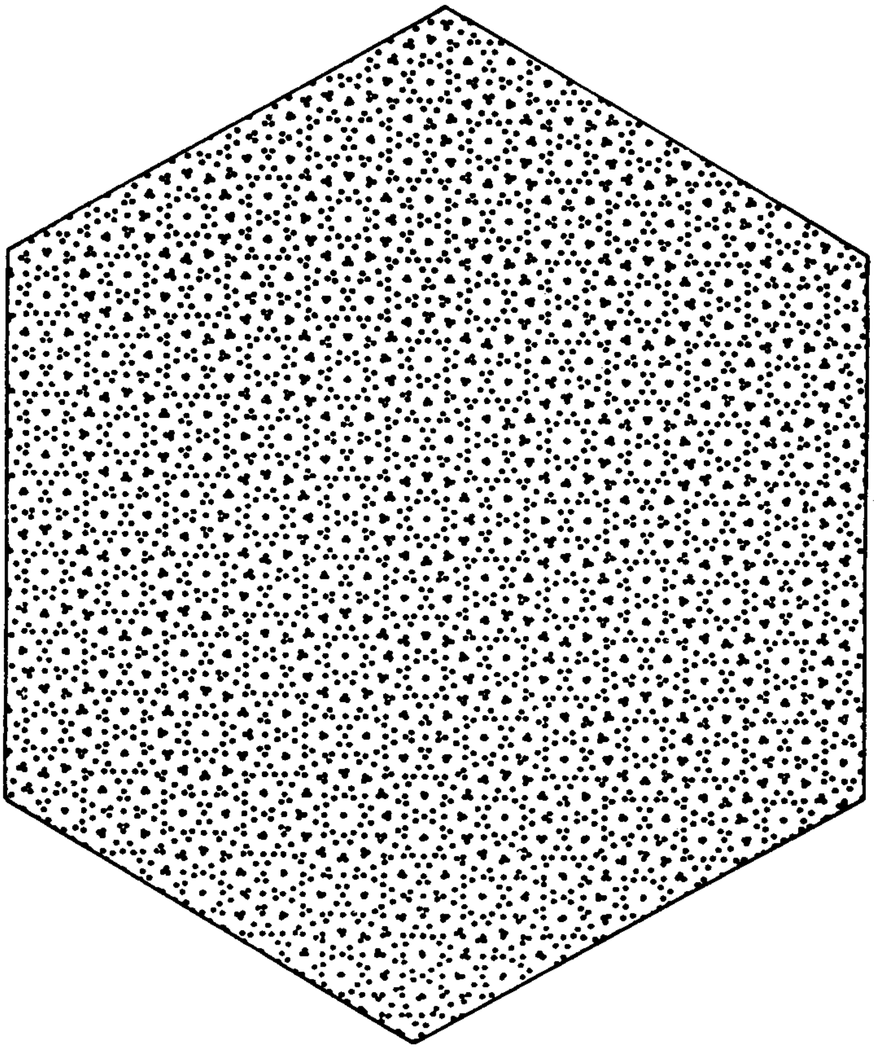


Figure 2.7: Evidence for the existence of active grouping processes. (From [114, 113].)

neural activity. For the example with the green hat and the red book the neurons representing the color 'green' and the form 'hat' would fire in synchrony, and also the representations for 'red' and 'book' would be synchronized, while 'red' and 'book' would not be synchronized neither with 'green' nor with 'hat'. Thus the basic idea is that time can be used as an additional dimension to represent dynamic relations between the spatial (i.e., neural) representations of entities.

From Figures 2.6 and 2.7 we already know that the temporal dimension is actually used to express relations between features of a picture. We can see consciously only one interpretation of a picture *at a time*. Thus at any given time the features constituting Fig.2.6 are bound either to the representation of a young or and old woman. There can be no dispute about the truth of this very weak version of the TCH involving the time window of conscious experience (about 3sec; see [139]).

However, binding may also be necessary *within one* conscious interpretation of a picture, e.g. to resolve which lines of the picture constitute the head of the old woman. Thus the real dispute is about the representation of *one* coherent percept, and about the *time scale* on which we interpret coincidence of neural activity. The *strong version* of the TCH postulates that the relevant time scale for binding is the time scale of a few milliseconds corresponding to single spikes [191, 192].

As a consequence the strong TCH predicts that spikes of neuron groups distributed over the cortex representing entities to be bound should be globally synchronized within a time window of perhaps $1-10$msec, whereas *all* neurons representing the whole scene should be activated perhaps within $10-50$msec, e.g., in phase-coding models within one period of an underlying fast oscillation (e.g., [150]).

Actually the TCH was further supported by the observation of high-frequency (40-60 Hz) oscillations in the visual cortex apparently reflecting global stimulus properties (e.g., the two-bars-experiments, see Fig.2.8; see also [41, 61, 168]). Evidence for this oscillatory activity was found in EEG or local field potential recordings and also in single-unit auto- or cross-correlograms. While the general idea of interpreting coincidence in assemblies on the timescale of milliseconds or spikes has been pursued by many if not most researchers elaborating the Hebbian assembly concept in the temporal domain [2, 133, 3, 136, 55, 6, 7, 198, 201], the special use of fine-timing made in the context of the visual 'binding problem', also related to the idea of 'phase-coding' (activating different assemblies at different phases of a single underlying oscillation), has been subject to considerable controversies (e.g. [196, 72, 195, 150, 106, 199, 200]).

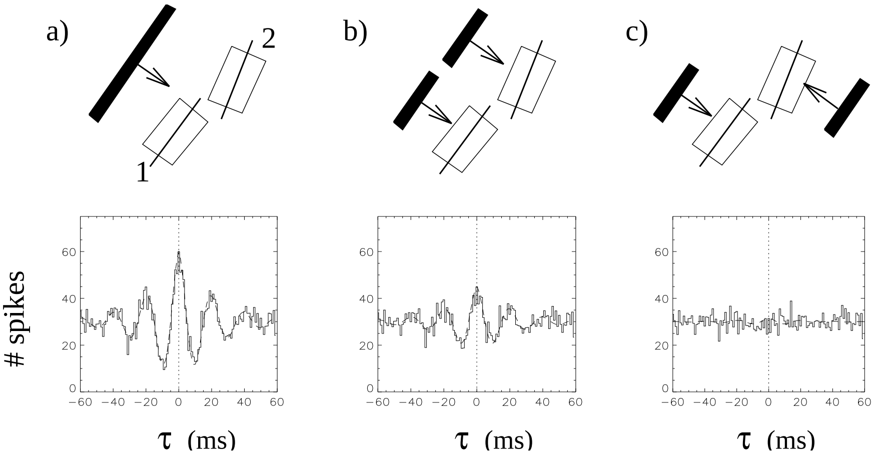


Figure 2.8: The so-called 'two bars experiment'. Two neurons are stimulated under three different stimulus configurations (boxes correspond to the receptive fields). From the corresponding spike activity cross-correlograms are shown. **a:** A single bar moves over the receptive fields (RFs) of the two neurons. The correlogram exhibits modulations with a central peak and additional sidepeaks which indicates synchronized oscillatory activity. **b:** Two smaller collinearly aligned bars move across the RFs. Now the modulations in the correlogram are much weaker. **c:** Two bars move in opposite direction over the RFs of the neurons. The correlogram is essentially flat, i.e., activity of the two neurons is uncorrelated. (Modified from [197].)

In chapter 5 we will explore the TCH by the simulation of two reciprocally connected visual cortical areas using spiking neurons. In contrast to other investigators implementing phase-coding-models we have difficulties to find arguments supporting the strong version of the TCH. We can obtain synchronization on the spike time scale only locally within the range of local or cortico-cortical connections, but not globally. Our results support rather a weaker version of the TCH, where global binding of corresponding features occurs by synchronization of neural activity within a time window of perhaps $30-300$msec whereas the representation of a whole scene can occur on the time scale of spike *rates* or of the 3sec window of a coherent conscious experience [139]. In contrast to the phase-coding models supporting the strong version of the TCH (usually based on a Hopfield architecture, see section 2.3.4) our model is consistent with many neuroanatomical and neurophysiological results (see chapter 5; cf. section 2.3.3).

The current implementation of our base model of two connected cortical areas accounts only for some of the described binding problems. In chapter 6 we will discuss the relation to the more challenging binding problems. In the next section we will introduce neural associative memory as a model for local cell assemblies in the brain that can also be useful for technical applications in computer science.

2.3 Associative memory

An **associative memory** is a system that can be used to store patterns or associations between patterns (Fig. 2.9), where the patterns are usually vectors of a fixed dimension.

In the case of **auto-association** patterns u^1, u^2, ..., u^M are stored during the learning phase. After learning the stored patterns can be retrieved by addressing the associative memory by noisy versions of the previously stored patterns. In the most general case one addresses with a noisy superposition of many stored patterns $\tilde{u}^{\mu_1}$, $\tilde{u}^{\mu_2}$, ..., $\tilde{u}^{\mu_N}$ plus additional noise yielding as retrieval output patterns $\hat{u}^{\mu_1}$, $\hat{u}^{\mu_2}$, ..., $\hat{u}^{\mu_M}$. In the ideal case those output pattern coincide with the original patterns u^{μ_1}, u^{μ_2}, ..., u^{μ_N} in order of relevance that made up the addressing pattern. However, in general the retrieval output will deviate from the ideal case and one can characterize the retrieval by a **retrieval quality** (with respect to some similarity measure on the set of possible patterns; see appendices A and B).

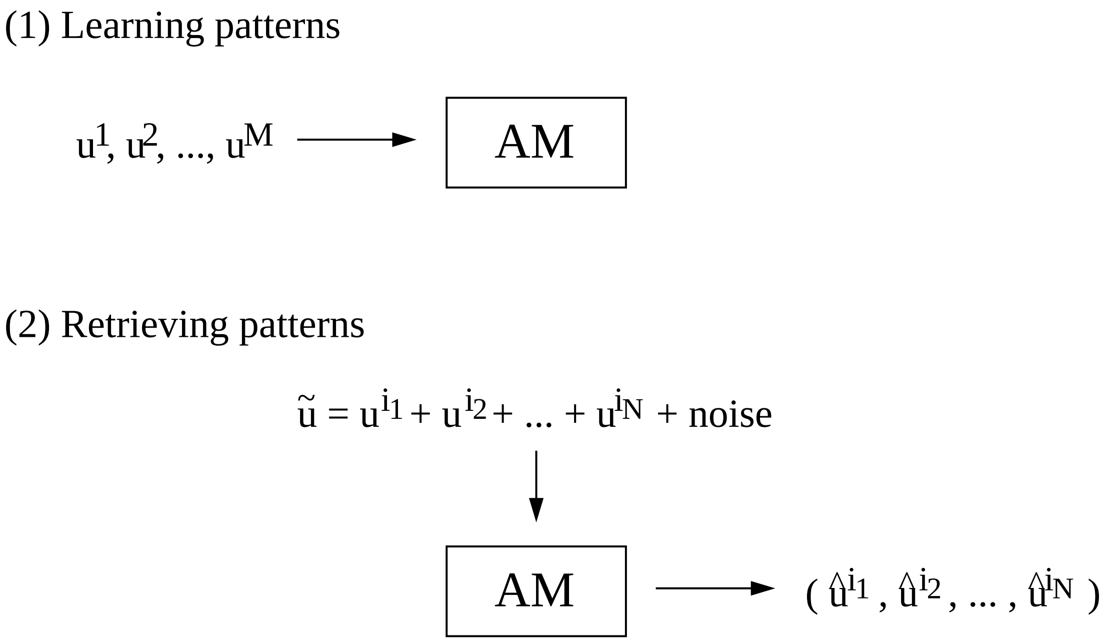


Figure 2.9: An associative memory (AM) can be used for learning and retrieving patterns. During the learning phase (top) M patterns (e.g., feature vectors) $u^1, u^2, ..., u^M$ are stored. For a retrieval (bottom) an address pattern $\tilde{u}$ is used which may consist of the superposition of N previously stored patterns $u^{i_1}, u^{i_2}, ..., u^{i_N}$ plus noise. As retrieval result the associative memory produces patterns $\hat{u}^{i_1}, \hat{u}^{i_2}, ..., \hat{u}^{i_N}$ which should equal the addressed patterns $u^{i_1}, u^{i_2}, ..., u^{i_N}$.

In the case of **hetero-association** we can store associations $u^\mu \to v^\mu$ between two sets of patterns U and V. This can be seen as a special case of auto-association where the stored patterns are tuples (u^μ, v^μ).

Associative memories can be implemented in different ways [204, 103, 104, 105, 33]. For example one could simply use a look-up table containing all the stored patterns. This would certainly yield the best possible results with respect to retrieval quality. However, the time necessary to conduct a retrieval and/or the memory capacity would not be optimal then. In contrast, neural implementations of associative memory [174, 204, 129, 69] can compress the necessary physical memory and retrieval time while preserving fault tolerance and the possibility of parallel implementation. In addition, **neural associative memory** can be used to model neural assemblies and cortical areas of the brain. The patterns (or assemblies) are subsets of a neuron population, and the information about which neurons belong to one pattern is stored in the synaptic weights by a Hebbian learning rule, i.e. synapses between neurons of a single pattern are stronger than other synapses, assuming the coactivation of the pattern neurons.

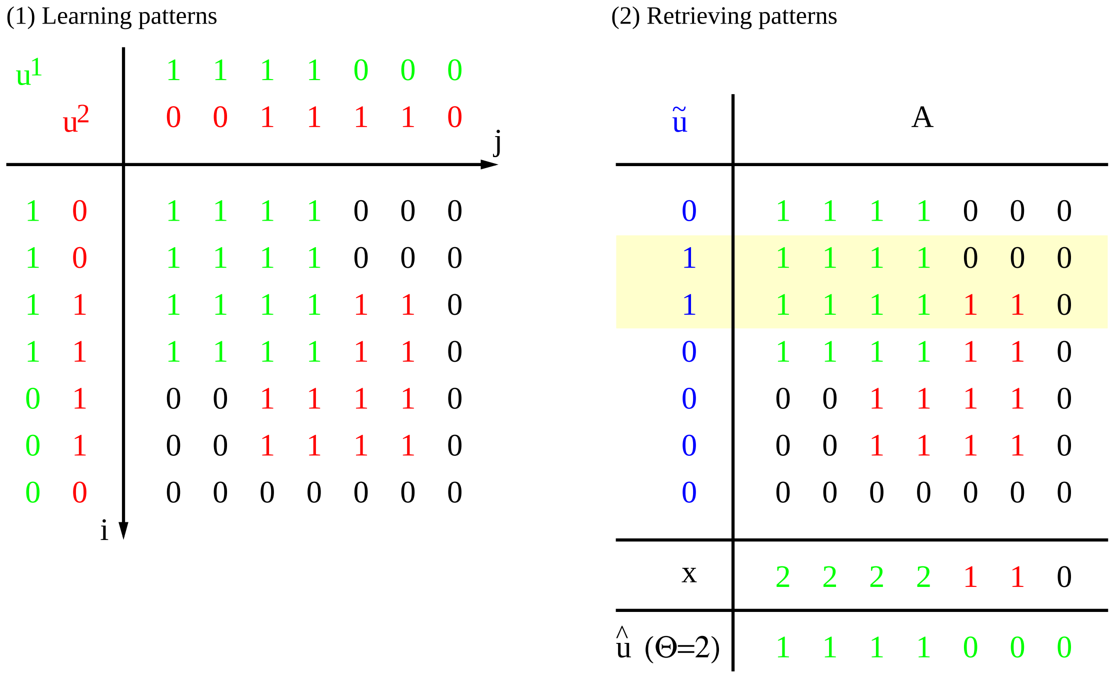
2.3.1 The Willshaw model

In the following chapters we will study variants of the binary **Willshaw model** of neural associative memory [174, 204, 129, 134, 29, 160, 171] which uses binary neurons and synapses. An illustration of the working principle of the binary Willshaw model is given in Fig. 2.10 for *auto-association*. The patterns are binary vectors u^μ of length n corresponding to a population of n neurons which can be active or silent. Similarly, each of the binary synapses can be either active (1) or inactive (0). Usually all the pattern vectors contain the same number k of active neurons, also called the *pattern activity*. Each neuron output is fed back to the other neurons where all the synapses are inactivated before learning begins. A pattern is stored by activating the pattern neurons and Hebbian learning which activates all synapses connecting two pattern neurons (Fig. 2.10, left part). In other words, patterns or assemblies are represented by *completely* connected neuron groups. After storing M patterns the **memory matrix** A of synaptic weights can be expressed as the superposition of outer products of the pattern vectors (see section 3.1.1). An important parameter of the memory matrix is the **memory load** or **matrix load** p_1 defined as the relative number of active synapses where p_1 is a function of n, k, and M. In general p_1 increases with the number of stored patterns M. In general the retrieval result will be better for smaller p_1 (i.e., smaller M). Analysis of the classical Willshaw model reveals that we typically can store patterns until $p_1 = 0.5$, i.e., until half of the synapses are active.

Retrieving is performed by activating the output variables of neurons corresponding to an address pattern $\tilde{u}$, which could be made up of one or more noisy versions of stored patterns. By summing synaptic input, the dendritic potentials $x = \tilde{u}A$ of the neurons are obtained by a vector matrix multiplication of the input vector $\tilde{u}$ and the memory matrix A. In a final step an appropriate threshold Θ is chosen such that each super-threshold neuron is activated. The **Willshaw threshold strategy** simply chooses the threshold equal to the number of one-entries in the address pattern. Note that if the address pattern $\tilde{u}$ is part of a stored pattern u^{μ_1}, i.e. $\tilde{u} \subseteq u^{\mu_1}$, the Willshaw strategy will activate all the neurons of this addressed pattern u^{μ_1}, but possibly also additional neurons. Thus the Willshaw strategy enables perfect pattern completion as shown in Fig. 2.10 (right part) if not too many patterns have been stored, and if the overlap $|\tilde{u} \cap u^{\mu_1}|$ is large enough (see section 3.1.2).

For *hetero-association* the Willshaw model can be applied analogously. For the association of pattern pairs $(u^\mu \rightarrow v^\mu)$ where the patterns u^μ and v^μ are binary vectors of lengths m and n, one can store auto-associatively the concatenated binary vectors (u^μ, v^μ) of length $m + n$ as described above. From the resulting $(m + n) \times (m + n)$-matrix A it is sufficient to use the $m \times n$ clip $H = (A_{ij})_{i \in \{1,2,\ldots,m\}, j \in \{m+1,m+2,\ldots,m+n\}}$ that contains information about the correlations of the two pattern sets (Fig. 2.11).

The information-theoretically analysis of the binary Willshaw-model has initially been conducted in 1969 by David Willshaw [204] and has been refined in 1980 by Günther Palm [129] (cf. appendix B.1). It has been found that high memory capacities can be obtained if the patterns are sparse, i.e. the number of one-entries k must be much smaller than the length n of the pattern vector (typically $k = \log n$). Using n neurons and **sparse coding** one can expect to store and correctly retrieve on the order of $n^2/\log^2 n$ patterns. This yields an asymptotic memory capacity of $\ln 2 \approx 0.7$ bit per synapse for $n \rightarrow \infty$, i.e., the classical Willshaw model can make use of about 70 percent of the allocated physical memory.


(1) Learning patterns

u^1	u^2							
		1	1	1	1	0	0	0
		0	0	1	1	1	1	0
1	0	1	1	1	1	0	0	0
1	0	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	0
0	1	0	0	1	1	1	1	0
0	1	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0

(2) Retrieving patterns

$\tilde{u}$	A						
0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0
x	2	2	2	2	1	1	0
$\hat{u}$ (Θ=2)	1	1	1	1	0	0	0

Figure 2.10: Working principle of the binary Willshaw model (for auto-association). During the learning phase (left) M binary vectors $u^1, u^2, ..., u^M$ are stored sequentially in the binary memory matrix A representing connections (synapses) between neurons. While initially all synapses are inactive each stored vector activates the synapses corresponding to its outer product, i.e., after learning of all patterns $A_{ij} = \min(1, \sum_{\mu=1,...,M} u_i^\mu \cdot u_j^\mu) \in \{0, 1\}$. For a retrieval (right) an address pattern $\tilde{u}$ is used for vector-matrix-multiplication resulting in the membrane potentials $x = \tilde{u}A$. To obtain the retrieval result $\hat{u}$ (=u^2) a threshold Θ is applied.

2.3.2 Neural associative memory and technical applications

In the recent past associative memory has been used in various technical applications like pattern recognition, language processing, computer vision, and word recognition (cf. [153, 206, 157]). The Willshaw model of associative memory turned out to be the most efficient model variant with respect to storage capacity, fault tolerance, and retrieval efficiency [134].

Instead of an associative memory one could use in principle simply look-up-tables where each pattern or pattern pair would be explicitely represented in a list or table. This would even be advantageous in many cases because in a look-up-table we can store an arbitrary number of patterns, we can have an arbitrary degree of fault tolerance (defined by a distance measure), and we can also have a large storage capacity (if we compress the patterns). One drawback of simple look-up-tables is that the access to the patterns is quite slow if many patterns are stored. Given an address pattern we have to compare it to any one of the stored patterns in order to find the most similar pattern. If we have stored n^2 patterns we need also n^2 times the number of steps necessary for comparing two patterns. If the address patterns would be scalars we could decrease the time by using binary search [33], but for general vectors this is not possible. Another possibility would be to use a hash-table [33] where a hash-function could compute in perhaps constant time access to the desired piece of information. However, hash-tables cannot easily implement fault tolerance.

In contrast, the Willshaw model can store a large number of pattern vectors (almost n^2 for vector size n) in a fault-tolerant way, where a sequential implementation of a retrieval requires only $\sim n \log n$ steps. A parallel implementation using n processors would even decrease the

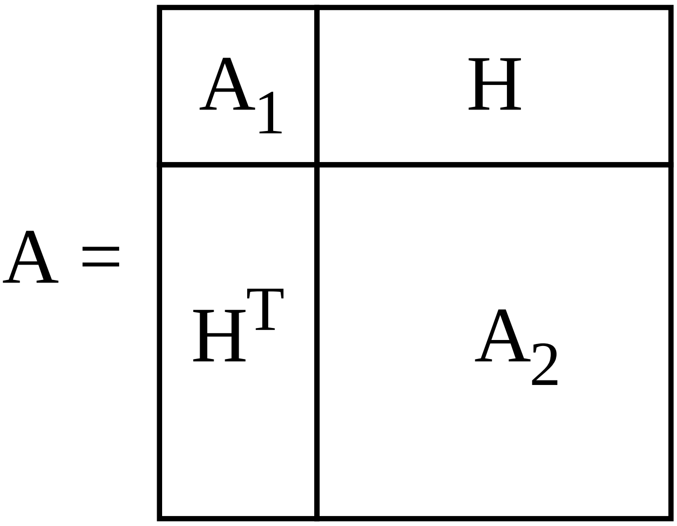


Figure 2.11: Hetero-association is a special case of auto-association, and vice versa. For example an auto-associative memory matrix can be divided in two quadratic auto-associative submatrices A_1 and A_2 and two hetero-associative matrices H and H^T.

number of necessary steps to $\sim \log n$ for sparse coding.

However, the usefulness of the Willshaw model (and also other models) of associative memory for technical applications has been severely limited by a number of facts. (1) High storage capacity near the asymptotic maximum ($\ln 2 \approx 0.7$) occur only for a very large vector size n, e.g., a storage capacity of 0.5 requires more than 10^6 neurons. (2) For given vector size n high storage capacities occur only if the number k of ones in the binary pattern vectors is in a very narrow range around $k = \log n$. For smaller or larger pattern activity k the storage capacity decreases rapidly (see section 3.1.2). (3) For relatively small n the performance of associative memory seems to be much lower than predicted by the classical analysis which relies on some assumptions which turn out to be invalid in some conditions even for very large n (see section 3.6). (4) We can store a large number of patterns only if the patterns are uncorrelated, e.g., random patterns. But in most applications we typically do not have random patterns.

In summary, current implementations of Willshaw associative memory are superior to standard algorithms of computer science [33] only for special parameter ranges, e.g, if a very large number of uncorrelated (random) pattern vectors have to be stored requiring fault tolerance. In all other cases we should prefer the usage of look-up-tables, search indices, or hashing.

A considerable part of this work contributes to the theory of Willshaw associative memory. In chapter 3 we find a simple solution to problems (1) and (2). It turns out that optimal compression of the memory matrix leads to much higher storage capacities for finite n and a very large range of k. For $n \to \infty$ we can even obtain storage capacity 1 for almost all sub-linear pattern activities $k(n)$. In section 3.6 a refined analysis of Willshaw associative memory [29, 171] reveals the origin for problem (3) and suggests some relief. However, problem (4) turns out to be a principle problem. The role of correlated patterns is not well understood so far and depends certainly on the domain of the application. In the next section we describe why the Willshaw model can be interpreted as a reasonable model for local cortical networks.

2.3.3 Willshaw associative memory as a model for a cortical column

Neural associative memory models such as the Willshaw model represent information about previously stored patterns in the synaptic connections between the neurons. This happens by application of a Hebbian learning rule on a *completely* connected network architecture, where a neuron typically is (or can be) connected to all other neurons. The assumption of a complete connection scheme is certainly *not* valid for the brain on the *global* level, but possibly at the *local* level of a cortical column. As discussed in section 2.1.3 it is reasonable to assume that any two neurons below the same square millimeter of cortex surface containing rawly about $n = 100000$ pyramidal cells can make connections on each other.

The Willshaw model assumes that synapses are binary corresponding to only two possible strengths or weights: zero or one. Despite such simple synapses the amount of information that can be stored *per synapse* ($\ln 2 \approx 0.7$) is quite near the theoretical optimum. It has been argued that the packing density of synapses in the cortex is the limiting factor for cortical connectivity (section 2.1.3). Therefore it is reasonable to assume that misplaced synapses that are depressed to a strength near zero by a correlational learning rule (see section 2.1.2; cf. [19, 1]; cf. [204, 64, 103, 69]) finally could be removed and possibly replaced by newly generated synapses at other locations. This would be an argument for identifying the *activated* synapses in the Willshaw model with the synapses anatomically present in the cerebral cortex. Since it is known that a cortical neuron is connected to about ten percent of the other local neurons in the same cortical column (see section 2.1.3) we can assume a matrix or memory load $p_1 = 0.1$ for a typical local cortical network.

If the hypothesis of local cortical cell assemblies (see section 2.2.2) is true, there is still the question for the size of a typical local assembly corresponding to the number k of ones in the binary pattern vectors of the Willshaw model. Theoretical considerations suggest that local assemblies may comprise a few hundred neurons (see [136, 203, 169]). Thus we may be not totally wrong when assuming $k = 100$.

Cortical tissue contains about $80 - 90\%$ excitatory cells (mostly pyramids) and only $10 - 20\%$ inhibitory cells (see section 2.1.3). Further only excitatory cells can make long-range cortico-cortical connections, and it has long been assumed that only excitatory cells are involved in learning. From these facts and assumptions it has been suggested that information processing is performed mostly by the excitatory cells ("skeleton cortex"), whereas the role of inhibitory cells would rather be the regulation of the total amount of neural activity [26, 130, 27]. This means for the Willshaw model that the memory matrix can be identified with the synaptic connections between the excitatory cells, while the inhibitory cells would implement an appropriate threshold control ([199, 89]; see section 4.6).

It must be stressed that Willshaw associative memory should be seen only as a very abstract model of a local cortical network. It does not account for a number of anatomical and neurophysiological facts or plausible assumptions: (1) Synaptic strengths as measured in neurophysiological experiments exhibit standard deviations on the order of the mean value (see Fig.2 in [115], Fig. 52 in [27]; cf. [137]). (2) In its basic form the Willshaw model does not reflect the layered structure of real cortex. (3) Even if the assembly hypothesis is true it is unlikely that a neuron of a local assembly is connected to *all* other neurons of the assembly. It would rather be plausible to assume that the connectivity within an assembly is increased to perhaps fifty percent compared to the average connectivity of ten percent. (4) It is also likely that not all the assemblies have the same size k. It is even possible that the distribution of assembly size has multiple peaks or that a large cell assembly could be structured in a larger number of smaller assemblies whereas the level of inhibition determines the current number of active neurons. (5) It is also probable that local cortical connectivity is not perfectly symmetric corresponding to stable activation states in auto-associative networks [129, 69, 70]. In addition to cell assemblies a second likely organization principle of cortical networks may be synfire chains [2, 5, 197] corresponding to asymmetric hetero-associative connections. Apart from cell assemblies and synfire chains there may also be other subsystems in the same cortical column performing processing of gradual information. (6) The postulated small patch of (a square millimeter) cortex is not isolated from neighboring columns, and the probability of a synaptic connection is not homogeneously $p_1 = 0.1$. Actually the probability of a connection between two neurons decreases rather continuously with the distance of the neurons [3, 27, 65].

Despite all these limitations we find it reasonable to assume as a working hypothesis that the Willshaw model with parameters $n = 100000$, $p_1 = 0.1$, and $k = 100$ reflects rawly the true situation of cerebral cortex if the assembly hypothesis is true. In the following we will refer to this parameter set as the *cortical parameters* of the Willshaw model.

2.3.4 Alternative models of associative memory

Besides the Willshaw model there exist a number of alternative models for associative memory, for example linear associative memory [103, 104], or the Hopfield model [69, 70].

While linear associative memory can be used to store a relatively small number of continuous-valued vectors, the Hopfield model is usually used for storing binary vectors similarly as the Willshaw model (although the Hopfield model uses -1/1 instead of 0/1 binary patterns). Although the Hopfield model became very popular in the last two decades, it is argued that the Willshaw model has a number of advantages over Hopfield-like models both for technical applications and brain modeling.

For technical applications the main advantages of the binary Willshaw model using n neurons are the following: (1) In the Willshaw model we can store a much larger number of patterns (typically $\sim (n/\log n)^2$) than in the Hopfield model (typically $\ll n$). Although the patterns in the Willshaw model have to be sparse (with typically $k = \log n$) while the patterns in the Hopfield model have to be non-sparse (typically $k = n/2$) this (2) results in a much higher storage capacity *per synapse* (asymptotically $\ln 2 \approx 0.7$) for the Willshaw model than in the Hopfield model. (3) Due to the binary synapses in the classical Willshaw model the storage capacity normalized to the required physical memory is equivalent to the storage capacity per synapse. In contrast the representation of each synapse in the Hopfield model requires a whole integer (corresponding to typically $\sim \log n$ bits of physical memory) such that the storage capacity per physical bit approaches zero for large n. As we will see in chapter 3 the Willshaw model can even reach storage capacity 1 if the memory matrix is optimally compressed. (4) Another consequence of the binary synapses and of the sparse patterns is that for the Willshaw model retrievals can be performed very quickly in $n \log n$ (sequential) or even $\log n$ (parallel) steps.

For biological models of the cerebral cortex there are additional advantages using the Willshaw model: (5) The binary pattern vector (0,1) reflects the binary nature of spikes. (6) The binary synaptic matrix (0,1) reflects the fact that a cortical neuron can make only one type of connection: either excitatory or inhibitory (see section 2.1.3). In contrast, the Hopfield model, for example, requires synapses with both signs on the same axon and even a possible change of sign of the same synapse. Therefore the Hopfield model can be adequate only for modeling at a lower resolution interpreting one Hopfield node as a mixed group of excitatory and inhibitory neurons. Even then it is especially difficult to interpret models using spiking neurons (e.g., [150]). Moreover, it is not yet clearly verified by neurophysiological experiments that learning of inhibitory synapses (which would be required by Hopfield nets) follows the rule postulated by Hopfield. (7) The Willshaw model requires sparse activation patterns as they seem to be present in the real brain. In contrast Hopfield nets rather need very large pattern activities.

2.4 Simulation tools and models of neurons and synapses

There exists a whole zoo of different neuron models at any level of complexity. The simplest models are memory-free *threshold elements* instantaneously summing the dendritic inputs, and subsequently applying a threshold operation. If the dendritic inputs exceed the threshold the output variable is 1 (corresponding to a spike), and otherwise 0. This model was already introduced in 1943 by McCulloch and Pitts [117] and it has been shown that networks of these units can compute any boolean function (and therefore also any discrete function). Actually, the neurons used for Willshaw and Hopfield associative memory as described in section 2.3 are also McCulloch-Pitts threshold units (although for the Hopfield model we have output values -1/+1 instead of 0/1).

For other neuron models used for approximation of continuous functions (e.g., by multi-layer-perceptrons using the backpropagation algorithm; [153, 155]) the output variable is a continuous sigmoid function of the dendritic inputs, whereas the biological interpretation of the output variable would rather be that of a spike *rate* instead of an individual weighed spike.

Since Hodgkin and Huxley [67] there have also been developed detailed biophysical models of neurons including detailed electrical and biochemical properties on the level of ion channels. Usually each anatomical or functional segment of the neuron's dendrite and soma is modeled by a separate compartment (e.g., [185, 138, 161]). Although these models can very precisely reproduce recordings from isolated neurons in brain slices, their implementation is very expensive in terms of computation time. Therefore it is currently possible to simulate only a relatively small number of such neurons on standard computers (perhaps 100-1000). Furthermore, it is difficult to analyze the behavior of such neurons mathematically, and sometimes using multi-compartment models appears to contradict the rule to prefer the simplest possible model that adequately reproduce a phenomenon under consideration (Occam's razor; e.g. [34]).

There have been derived simplified spiking models not requiring the costly spike generation of the original Hodgkin/Huxley model. The so-called integrate-and-fire-neurons integrate presynaptic input by a low-pass-equation with corresponding membrane time constant to the trans-membrane potential. If the membrane potential exceeds the threshold the spike variable is activated for one time step. After a spike additional mechanisms (e.g., reset of the membrane potential, or temporary increase of the threshold) prevent the neuron from emitting a further spike at the next time step. These simplified models have the advantage that a relatively large number of neurons can be efficiently simulated while they still produce spike data which can be compared to physiological recordings. Furthermore, even the simplified spiking models show a qualitatively different behavior than gradual rate-function models in otherwise comparable parameter configurations [202, 197].

An even more simplified model of a spiking neuron in an oscillatory regime are *phasors* (e .g., [8]) which can be analyzed and simulated with minimal costs. On the other side they are difficult to interpret with respect to biology.

In this work we use *three* different neuron models. In chapter 3 we use simple binary McCulloch/Pitts neurons for the analysis of Willshaw associative memory. For a spiking variant of the Willshaw model in chapter 4 we introduce the so-called counter model [86, 89, 90] which can be seen as the abstract version of a more realistic neuron model used for the biological models in chapter 4 and 5.

The biological model used in this work is a one-point (i.e., one-compartment) integrate-and-fire neuron model with an intermediate level of complexity - comparable with the so-called PTNR10 model of MacGregor [111] - that produces relatively realistic input-output dynamics. The model has two states g_{ex} and g_{in} corresponding to excitatory and inhibitory *synaptic conductances*. Presynaptic spikes cause an instantaneous increase of the conductances followed by an exponential decay with time constants τ_{ex} and τ_{in} according to the following low-pass equations,

$$\tau_{\text{ex}} \frac{d}{dt} g_{\text{ex}}(t) = -g_{\text{ex}}(t) + \sum_s w_s \sum_i \delta(t - t_{s,i}) + \text{noise} \tag{2.1}$$

$$\tau_{\text{in}} \frac{d}{dt} g_{\text{in}}(t) = -g_{\text{in}}(t) + \sum_s w_s \sum_i \delta(t - t_{s,i}) + \text{noise}, \tag{2.2}$$

where the sums are over the synapses s and the corresponding time points of presynaptic spikes $t_{s,i}$. Similarly each spike emitted by the neuron (at time t_i) increases the habituation variable h instantaneously by H followed by an exponential decay with time constant τ_h,

$$\tau_h \frac{d}{dt} h(t) = -h(t) + H \sum_i \delta(t - t_i) \tag{2.3}$$

The synaptic conductances cause excitatory and inhibitory postsynaptic currents which drive the transmembrane potential x towards the excitatory and inhibitory equilibrium potentials E_{ex} and E_{in}. The low-pass equation for x can be written as

$$\tau_x \frac{d}{dt} x(t) = -x(t) + g_{\text{ex}}(t)(E_{\text{ex}} - x(t)) + g_{\text{in}}(t)(E_{\text{in}} - x(t)) + \text{noise} \tag{2.4}$$

The dynamic firing threshold Θ can be expressed as

$$\Theta(t-) = \begin{cases} \infty & , \quad t - s(t-) \leq R_a \\ \Theta_\infty \cdot (1 + \frac{R_r}{t - s(t-) - R_a}) + h(t-) & , \quad \text{otherwise.} \end{cases} \tag{2.5}$$

where R_a and R_r are parameters for absolute and relative refractoriness, and $s(t)$ is the time point of the latest spike before time t. The neuron emits a spike (variable y) as soon as the membrane potential x exceeds Θ.

$$y(t) = \mathbf{1}_{[x(t) \geq \Theta(t-)]}. \tag{2.6}$$

Typical parameters used for the simulations described in this work are summarized in table 2.1.

type	τ_{ex} [ms]	τ_{in} [ms]	τ_x [ms]	E_{ex} [mV]	E_{in} [mV]	Θ_∞ [mV]	R_a [ms]	R_r [ms]	τ_h [ms]	H [mV]
e.s.	5	7	10	80	-10	10	2	3.0	150	0.6
i.s.	5	7	10	80	-10	10	2	3.5	-	-
i.g.	5	7	5	80	-10	-	-	-	-	-

Table 2.1: standard neuron parameters (averages) for the three neuron types: e.s.=excitatory spiking, i.s.=inhibitory spiking, i.g.=inhibitory gradual

Similar as for the neuron models there are also many different *simulation* tools which can be used in principle for the implementation. Each tool has its strengths and weaknesses with respect to features like detailed modeling of single neurons (e.g., [66, 25]) or support of parallel computer architectures (e.g., [177]). The implementation of the network models described in this work required in particular efficient implementation of special topographic network mappings, efficient implementations of delayed connections and binary associative memories on relatively small computers, at least in the early phase of this work [86, 89]. To achieve this the Felix simulator for neural networks of Thomas Wennekers [197] has been extended to an object-oriented C++ based simulation tool Felix++ [88]. All simulations described in this work have been implemented using Felix or Felix++. In appendix C a brief overview of the architecture of Felix++ is given.

Chapter 3

Willshaw associative memory

Associative memories are systems that contain information about a finite set of associations between pattern pairs $\{(u^\mu \to v^\mu) : \mu = 1, ..., M\}$. A possibly noisy address pattern $\tilde{u}^\mu$ can be used to retrieve an associated pattern $\hat{v}^\mu$ that ideally will equal v^μ.

In *neural implementations* the information about the associations is stored in the synaptic connectivity of one or more neuron populations. Neural implementations can be advantageous over hash-tables or simple look-up-tables if the number of patterns is large, if parallel implementation is possible, or if fault-tolerance is required, i.e. if the address patterns $\tilde{u}^\mu$ may differ from the original patterns u^μ used for storing the associations.

The so-called *storage capacity* of an associative memory can be measured information-theoretically by maximizing the transinformation between the stored patterns v^μ and the retrieved patterns $\hat{v}^\mu$. A valid quality measure is the storage capacity *normalized* to the required *physical memory* (or, less valid, normalized to the number of required synapses).

In 1969 Willshaw et al. [204] discovered that a high (normalized) storage capacity of $\ln 2 \approx 0.7$ is possible for Steinbuch's neural implementation of associative memory using binary patterns and synapses [174]. These results were refined by Palm in 1980 [129]. The analysis showed that the upper bound $\ln 2$ is only approached asymptotically for very large (i.e., not practical) numbers n of neurons (but see [160]). Even worse, high capacities are only possible for sparse patterns, i.e., if the number k of one-entries is much smaller than the pattern size n, typically $k = c \log n$ for a constant c (cf. Fig. 3.1). For k smaller or larger than required for the optimum, the capacity usually decreases rapidly to small values. These requirements imposed a severe limitation to the practical use of the *Willshaw (or Steinbuch) model* since it turned out to be difficult to find adequate codes that match the strict requirements for the pattern activity k. Another severe limitation is that high capacities can only be achieved for random patterns, i.e., if the stored patterns are uncorrelated.

In the mid eighties alternative neural implementations were proposed [69, 70, 71]. But it turned out that the capacity of the Willshaw model exceeds the capacities of other models by far [134]. From the late eighties on methods were developed to improve the retrieval results under noisy conditions, e.g. by bidirectional, iterative, or spiking retrievals [107, 160, 171, 89], and an early attempt to use sparse matrices efficiently was not very promising [14].

Although it was possible to slightly extend the upper theoretical bound for the storage capacity (per synapse) to $1/(2 \ln 2) \approx 0.72$ for non-binary synapses [189, 134], the bound of $\ln 2$ for the storage capacity normalized to the required physical memory was never exceeded for non-trivial models. It was even believed that it would be impossible to reach capacity 1 (or at least exceed $\ln 2$) for really distributed storage [134].

In this chapter we will see that it is indeed possible to reach capacity 1 asymptotically for really distributed storage (see also [87]). This can be achieved for the Willshaw model if the patterns are *not* optimally sparse in the sense of the classical analysis ($k \neq c \log n$). Then the memory matrix contains sparsely either one-entries (for $k < c \log n$) or zero-entries (for $k > c \log n$). It turns out that optimal compression of the memory matrix (e.g. by a Huffman or Golomb code [74, 57]) leads to capacities exceeding the classical optimum. Although capacity 1 is approached only for unrealistically large n, values in the range of $\ln 2$ (the classical upper bound) can already be reached for practical n. Even more relevant for real applications with finite n is the fact that high capacities on the order of or above the classical optimum are obtained for a very broad range of k, including ultra-sparse patterns (e.g. k constant) and moderately-sparse patterns (e.g. $k = \sqrt{n}$).

In the next sections we will investigate the Willshaw model along the dimensions storage capacity, retrieval efficiency, and fault tolerance. In section 3.1 we will briefly review the classical analysis of the basic storing and retrieval algorithms used in the Willshaw model. These classical results are applied in section 3.2 to analyze optimal compression of the memory matrix. Sections 3.4 and 3.5 deal with efficient implementations of the Willshaw model with compressed memory matrix on sequential and parallel computer architectures. In section 3.3 we discuss the fault tolerance of the Willshaw model, where we will focus on missing ones in the address patterns (for false ones see section 4.4). The results in this chapter (as well as in much of the classical literature) concerning storage capacity and fault tolerance are based on the binomial approximation of the true distribution of dendritic potentials (the so-called *Willshaw distribution*). In section 3.6 we will work out why the binomial approximation can be very bad for finite number of neurons. In the concluding section 3.7 we will discuss the implications of the results obtained in this chapter for technical applications and biological modeling.

3.1 Classical analysis of the Willshaw model

In this section we will briefly review the classical analysis of the basic storage and retrieval procedures used in the Willshaw model [174, 204, 129, 134, 171].

3.1.1 Storing and retrieving patterns

Consider the set of *associations* between M pattern pairs $\{(u^\mu \to v^\mu) : \mu = 1, ..., M\}$ where all patterns are *binary vectors* of length n containing exactly k ones.

We *store* the patterns hetero-associatively in the *binary memory matrix* $A \in \{0,1\}^{n \times n}$ corresponding to synaptic connections between two neuron populations, the *address population* corresponding to the patterns u^μ, and the *addressed* or *retrieval population* corresponding to the patterns v^μ. Matrix entry A_{ij} corresponds to the synaptic weight of the connection from neuron i in the address population to neuron j in the retrieval population, and is obtained from the superposition of outer products of corresponding pattern vectors as

$$A_{ij} = \min\left(1, \sum_{\mu=1}^{M} u_i^\mu \cdot v_j^\mu\right) \in \{0,1\}\,. \tag{3.1}$$

For pattern *retrieval* we address the associative memory by an address pattern $\tilde{u}$ which may (for example) be a noisy version of one of the original address patterns u^μ. By vector-matrix-multiplication $\tilde{u}A$ we obtain neural potentials which can be transformed to the retrieval result $\hat{v}$ by applying a threshold Θ,

$$\hat{v}_j = \begin{cases} 1\,, & \left(\sum_{i=1}^{n} \tilde{u}_i A_{ij}\right) \geq \Theta \\ 0\,, & \text{otherwise} \end{cases}\,. \tag{3.2}$$

The choice of the threshold Θ is important for good retrieval results. One possibility which is referred to as the *Willshaw-retrieval-strategy* is simply setting the threshold equal to the number of one-entries in the address pattern $\tilde{u}$,

$$\Theta = \sum_{i=1}^{n} \tilde{u}_i \tag{3.3}$$

If all the one-entries in the address pattern $\tilde{u}$ occur also in one of the original patterns, u^μ, then the one-entries in the retrieval result $\hat{v}$ will be a superset of the ones in the associated pattern, v^μ. Indeed, this strategy is the only possible choice if one assumes that the address pattern contains no 'false alarms', and it plays also an important role for pattern separation in spiking associative memories with time-continuous retrievals (see chapter 4, cf. [89, 92]). In the following we will investigate the storage capacity of the Willshaw model for this simple one-step retrieval algorithm.

3.1.2 Classical analysis of one-step retrieval

Consider storing hetero-associatively M random pattern pairs (u^μ, v^μ) as described in section 3.1.1. All patterns have size n and contain exactly k ones. The probability that a given synapse is *not* set by the association of one pattern pair is approximately $1-k^2/n^2$. Therefore the probability p_1 of a one-entry in the resulting binary $n \times n$-matrix A after association of all M pattern pairs can be approximated by

$$p_1 \approx 1-(1-k^2/n^2)^M \approx 1-e^{-Mk^2/n^2}, \tag{3.4}$$

which can be resolved for M yielding

$$M \approx \frac{\ln(1-p_1)}{\ln(1-k^2/n^2)} \approx -\frac{n^2}{k^2}\ln(1-p_1). \tag{3.5}$$

Note that the first approximations would become exact if the ones in the binary patterns were chosen independently of each other with probability k/n. However, we can assume that this discrepancy is negligible for our purposes. The second approximations are valid for large M (eq.3.4) and $k \ll n$ (eq.3.5).

After storing the patterns we can use an address pattern containing $k_1 := \lambda \cdot k \leq k$ one-entries ($0 < \lambda \leq 1$) of pattern u^μ (but no false ones) to retrieve the associated patterns v^μ. Applying the Willshaw strategy $\Theta = k_1$ (see sect. 3.1.1) we obtain a retrieved pattern $\hat{v}^\mu$ that contains all the one-entries of v^μ, but possibly also false one-entries. The probability p_{01} of a false one can be approximated by

$$p_{01} \approx p_1^{k_1}, \tag{3.6}$$

assuming that the ones in the memory matrix A are generated *independently* of each other. This *binomial approximation* leads to eq.3.6 since a false one occurs if and only if *all* the k_1 relevant synapses of the corresponding non-addressed neuron are active. In section 3.6 we will work out why the binomial approximation can be very bad for finite n, k, and M.

To obtain good retrieval results we demand similarly as in [134] a *high-fidelity-requirement* $p_{01}/(k/n) \approx 0$ and $p_{01} \to 0$ for $n \to \infty$ which states that the relative number of false ones is near zero. *High-fidelity* can be obtained by requiring $p_{01} \leq \epsilon k/n$ for a small positive ϵ and $k/n \to 0$ which is true for (sublinearly) sparse patterns. From eq. 3.6 we obtain a minimal address pattern size of

$$k_{1,\mathrm{hifi}} := \frac{\ln(n/k)-\ln\epsilon}{-\ln p_1}. \tag{3.7}$$

The storage and retrieval of the M pattern pairs as described can approximately be thought of as the transmission of Mn binary digits (of the v^μ patterns) over a binary channel (cf. appendix B.1). For small k/n the information per digit is $I(k/n) \approx -(k/n)\log(k/n)$ and for small ϵ the totally stored (or transmitted) information (cf. eq.B.2) is about

$$C_A(n,k;M) \approx MnI(k/n) \approx Mk\,\mathrm{ld}\,(n/k) \tag{3.8}$$

To *maximize* C_A for given n and k we store as much patterns as possible so that we can still fulfill the *hifi-requirement*. This means we can increase the memory load p_1 by storing patterns until $k_{1,\mathrm{hifi}}(p_1) = k$. From the hifi-requirement and eq. 3.6 it follows the *maximal* matrix load

$$p_{1,\max} = (\epsilon k/n)^{1/k}, \tag{3.9}$$

and with eq. 3.5 we get the *maximal* number of stored patterns,

$$M_{\max} = \frac{\ln(1-p_{1,\max})}{\ln(1-k^2/n^2)} \approx -\frac{n^2}{k^2}\ln(1-p_{1,\max}) \tag{3.10}$$

$$\approx -(\mathrm{ld}\, p_{1,\max})^2 \cdot \ln(1-p_{1,\max}) \cdot \frac{n^2}{(\mathrm{ld}\, n)^2}. \tag{3.11}$$

For the second approximation we used $k \approx -\mathrm{ld}\,(n/k)/\mathrm{ld}\, p_{1,\max}$ from eq. 3.7. Thus we obtain for the (normalized) *storage capacity* $C(n,k)$ as the *maximal* stored information per physical memory unit (bit for ld $= \log_2$),

$$C(n,k) := \frac{C_A(n,k;M_{\max})}{n^2} \approx -\ln(1-p_{1,\max})\frac{\mathrm{ld}\,(n/k)}{k} \tag{3.12}$$

$$\approx \mathrm{ld}\, p_{1,\max} \ln(1-p_{1,\max}), \tag{3.13}$$

where for the second approximation we used again $k \approx -\mathrm{ld}\,(n/k)/\mathrm{ld}\, p_{1,\max}$ from eq. 3.7. To compute the *storage capacity* $C(n) := C(n,k_{\mathrm{opt}})$ of an associative memory with n neurons, we still have to *maximize* $C(n,k)$ with respect to k. From eq. 3.13 we see that the optimum (for large $n \to \infty$) occurs for $p_1 = 0.5$ corresponding to $k_{\mathrm{opt}} = \mathrm{ld}\, n$. Therefore $C(n) \to \ln 2$ for $n \to \infty$.

In the following we will simply write p_1 for $p_{1,\max}$, and M for $M_{\max}$. Figure 3.1 illustrates some results of the classical analysis. As can be seen in Fig.3.1d the convergence of $C(n)$ towards $\ln 2$ for $n \to \infty$ is quite slow. For $\epsilon = 0.01$ values $C(n) > 0.5$ are obtained only for $n > 3.7 \cdot 10^6$. Surprisingly, maximization for the number of stored patterns requires a smaller $p_1 \approx 0.16$ than the maximum of the storage capacity (Fig. 3.1c).

For a given n, high capacities C are obtained only for a very narrow range of k around an optimum k_{opt} ($\epsilon = 0.01$, $n = 1000000$, $k_{\mathrm{opt}} = 21$, and $C \approx 0.49$ in Fig.3.1a,b). For larger or smaller k the capacity decreases rapidly. The memory load p_1 increases with k monotonically from 0 to 1. For $k = k_{\mathrm{opt}}$ the memory matrix is not sparse with $p_1 \approx 0.5$. and compression is not possible. However, for k smaller and larger the memory load approaches 0 and 1, respectively, and we can hope to improve the storage capacity by compressing the memory matrix. This is investigated in the next section.

3.2 Matrix compression yields storage capacity 1

As a result from the analysis above, the storage capacity *per synapse* is limited to $\ln 2$, and even this value is achieved only for infinitely large numbers of neurons $n \to \infty$ and a certain $k(n)$. Nevertheless what we can do is making use of the *physical memory* in an optimal way by compressing the memory matrix A. However, compressing the matrix for the classical maximum C(n) is useless since it is obtained at a memory load $p_1 = 0.5$ where A is incompressible. On the other hand, for (classically) non-optimal k we will obtain $p_1 \neq 0.5$, and thus compression of the memory matrix will decrease the required physical memory.

A *naive approach* would simply be to store explicitely the indices of the sparse matrix elements (either ones for $p_1 < 0.5$ or zeros for $p_1 > 0.5$). Such a sparse matrix representation would require only $\min(p_1, 1-p_1)n^2 \mathrm{ld}\, n$ bits of physical memory. Thus the resulting storage capacity C^{sprs} can improve by a factor $\min(p_1, 1-p_1)\mathrm{ld}\, n$, and for large n we obtain from eq.3.13

$$C^{\mathrm{sprs}}(n,k) := \frac{C(n,k)}{\min(p_1,1-p_1)\mathrm{ld}\, n} \approx \frac{\mathrm{ld}\, p_1 \cdot \ln(1-p_1)}{\min(p_1,1-p_1)\mathrm{ld}\, n}. \tag{3.14}$$

Unfortunately, it turns out that storage capacity 1 cannot be achieved by this naive approach for pattern activities $k(n)$ where we have really distributed storage (i.e., $k > 1$; see eq.3.19; cf. Figure 3.2a,b,d; cf.[14]) or where we can store many patterns (i.e., $M > n$, see eq.3.25, eq.3.23, cf. table 3.1).

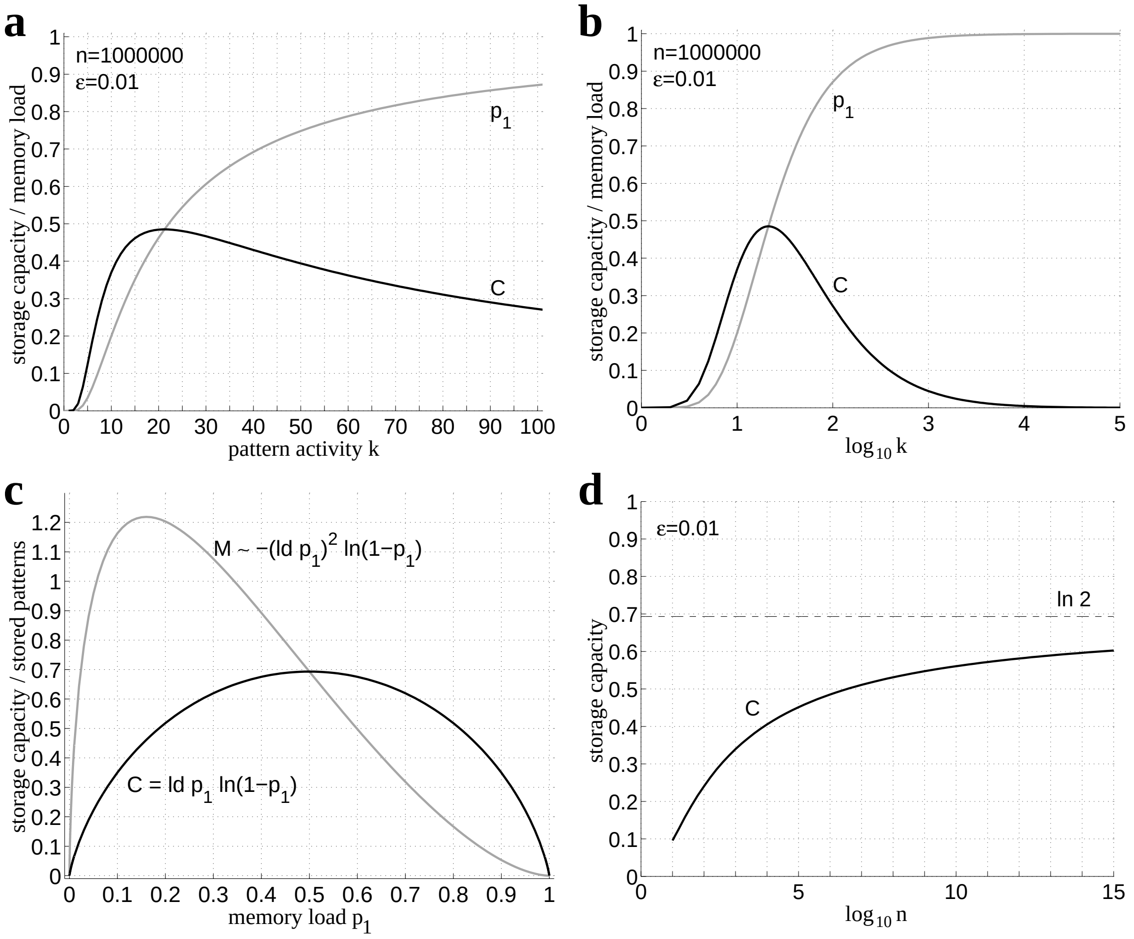


Figure 3.1: Some results from the classical analysis of the Willshaw model (hifi-parameter $\epsilon = 0.01$, see text). **a** : Dependence of the storage capacity $C(n,k)$ (black) and memory load $p_1(n,k)$ (gray) on the pattern activity k (number of one-entries in a pattern) for pattern size $n = 10^6$. The optimal capacity $C \approx 0.49$ is reached for $k = 21$. For larger or smaller k the capacity decreases rapidly. The memory load p_1 increases with k monotonically from 0 to 1. p_1 is near 0.5 for optimal k. **b** : same as (a), but logarithmic scale. **c** : For given large n we have $C \sim \mathrm{ld}\, p_1 \ln(1-p_1)$ (black, see eq.3.13) but $M \sim (\mathrm{ld}\, p_1)^2 \ln(1-p_1)$ (gray, see eq.3.11). While C is maximal for $p_1 = 0.5$, maximal M is obtained for moderately sparse memory matrix (matrix load $p_1 \approx 0.16$). **d** : The convergence of $C(n) \to \ln 2$ for $n \to \infty$ is rather slow, unfortunately.

However, *optimally* compressing the memory matrix by applying Huffman or Golomb coding [74, 57] (the latter requires $p_1 \to 0$ or $p_1 \to 1$, see appendix A.3.2) will decrease the required physical memory by a factor of $I(p_1) := -p_1 \mathrm{ld}\, p_1 - (1-p_1)\mathrm{ld}\,(1-p_1)$ ([164, 34], see also appendix A.1). Fig.3.2a,b shows the dependence of C and the improved storage capacity $C^{\mathrm{cmpr}} := C/I(p_1)$ on k for $n = 10^6$ and $\epsilon = 0.01$. While p_1 increases monotonically with k from 0 to 1 (cf. Fig.3.1a), surprisingly, for $k \neq k_{opt}$ we have not only $C^{\mathrm{cmpr}}(k) > C(k)$, but also $C^{\mathrm{cmpr}}(k) > C(k_{\mathrm{opt}})$. Indeed it turns out that $C(k_{\mathrm{opt}})$, the classical maximum of C, is rather a local minimum of C^{cmpr} (Fig.3.2b,c):

From eq. 3.13 we obtain for large n an improved storage capacity C^{cmpr} of

$$C^{\mathrm{cmpr}}(n,k) \quad := \quad \frac{C(n,k)}{I(p_1)} \quad \approx \quad \frac{\ln p_1 \ln(1-p_1)}{-p_1 \ln p_1 - (1-p_1)\ln(1-p_1)}. \tag{3.15}$$

For small $p_1 \to 0$ we have $I(p_1) \approx -p_1 \mathrm{ld}\, p_1$ and $\ln(1-p_1) \approx -p_1$, and therefore $C^{\mathrm{cmpr}} \to 1$.

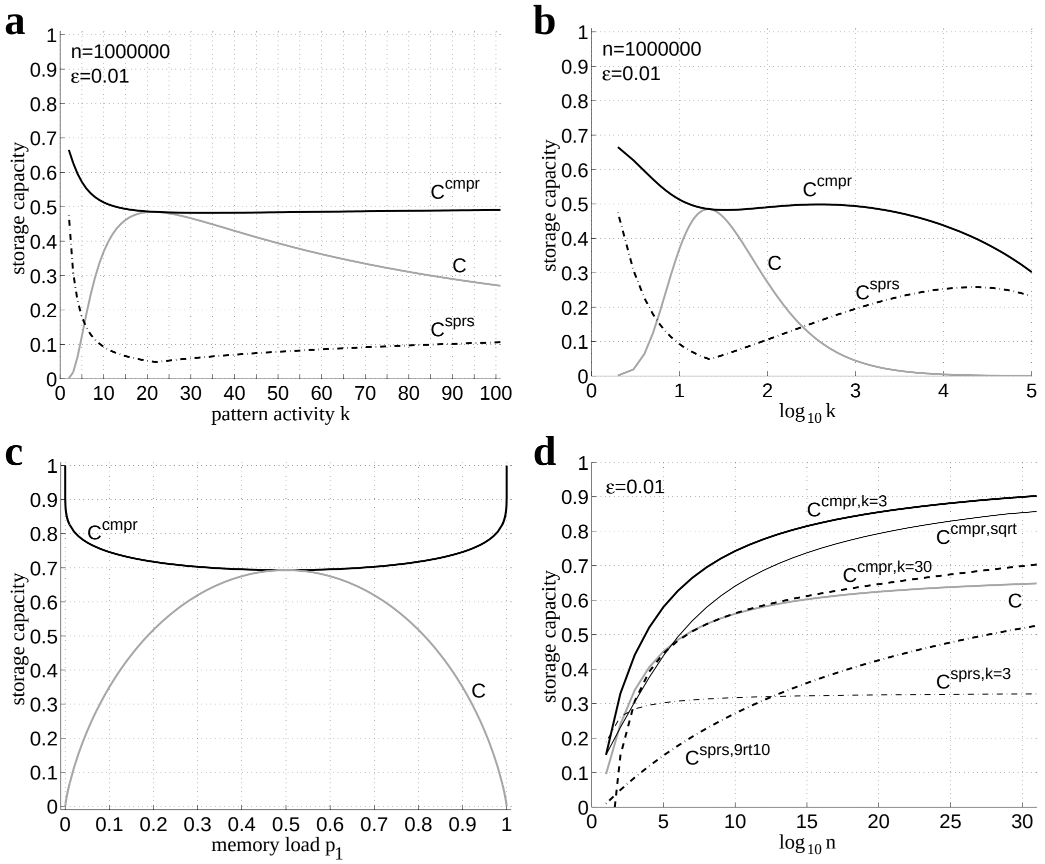


Figure 3.2: Comparison of storage capacities for the *classical* Willshaw model (C, gray), for sparse representation of the memory matrix (C^{sprs}, dash-dotted), and for optimally compressed memory matrix (C^{cmpr}, black). **a** : Comparison of $C(n,k)$, $C^{\text{sprs}}(n,k) := C/(\min(p_1, 1-p_1)\text{ld}\,n)$, and $C^{\text{cmpr}}(n,k) := C/I(p_1)$ for $n = 10^6$, $\epsilon = 0.01$ (cf. Fig.3.1a). **b** : Same as (a) but with logarithmic scale for k. While C is large only for a narrow range of k around the optimum $k_{\text{opt}} = 21$, $C^{\text{cmpr}}(n,k) > C(n,k_{\text{opt}})$ is possible for a very broad range of k. C^{sprs} achieves values around $C(n,k_{\text{opt}})$ only for very small k. **c** : Asymptotic storage capacities C and C^{cmpr} over memory load p_1 for $n \to \infty$. The maximum $\ln 2$ of C at $p_1 = 0.5$ turns out to be the minimum of C^{cmpr}. $C^{\text{cmpr}} \to 1$ for $p_1 \to 0$ or $p_1 \to 1$. **d** : $C(n)$, $C^{\text{sprs}}(n)$, and $C^{\text{cmpr}}(n)$ for different functions $k(n)$ and $\epsilon = 0.01$. C (gray) reaches $\ln 2 \approx 0.7$ asymptotically for $k = \text{ld}\,n$. C^{sprs} achieves $1/k$ for constant k (e.g., $C^{\text{sprs},k=3} \to 1/3$ for $k = 3$; thin dash-dotted) and d for $k = n^d$ (e.g., $C^{\text{sprs},9\text{rt}10} \to 0.9$ for $k = n^{0.9}$; thick dash-dotted). Optimal compression yields asymptotically storage capacity $C^{\text{cmpr}} \to 1$ for constant $k = 3$ ($C^{\text{cmpr},k=3}$, thick), constant $k = 30$ ($C^{\text{cmpr},k=30}$, dashed), and $k = \sqrt{n}$ ($C^{\text{cmpr,sqrt}}$, thin) as well as for all sublinear but non-logarithmic functions (see text).

Similarly for large $p_1 \to 1$ we have $I(p_1) \approx -(1-p_1)\text{ld}\,(1-p_1)$ and therefore also $C^{\text{cmpr}} \approx (\ln p_1)/(1-p_1) \to 1$.

Together, we obtain an asymptotic storage capacity of 1 as soon as $p_1 \to 0$ or $p_1 \to 1$ for $n \to \infty$. From eq. 3.9 we see that this can be fulfilled for almost all sublinear functions $k(n)$. For $c > 0$ and logarithmic $k = c \ln n$ we obtain from eq. 3.9

$$\ln p_1 = \frac{\ln(\epsilon c \ln n) - \ln n}{c \ln n} \to -\frac{1}{c}, \tag{3.16}$$

And therefore $p_1 \to \exp(-1/c)$ for $n \to \infty$. So varying c one can obtain all possible values in $(0;1)$ for asymptotic p_1 and correspondingly all values in $[\ln 2; 1)$ for asymptotic C^{cmpr} (see Fig.

3.2c). Since p_1 is monotonically increasing in k we conclude that $C^{\mathrm{cmpr}} \rightarrow 1$ for all sublinear, but non-logarithmic functions $k(n)$.

Fig.3.2d shows the convergence of C^{cmpr} for $k = 3$, $k = 30$, and $k = \sqrt{n}$ in comparison to the classical C. All the C^{cmpr} functions slowly approach 1. Values in the range of $\ln 2$, the classical maximum, are already approached for relatively small n at least for ultra-sparse patterns (e.g. $n = 10^8$ for $k = 3$).

Although the convergence of C^{cmpr} is fastest for ultra-sparse patterns ($k < k_{\mathrm{opt}}$), the results for the moderately-sparse patterns ($k > k_{\mathrm{opt}}$) are also relevant for applications because the broad range of k with $C^{\mathrm{cmpr}}(k) \geq C(k_{\mathrm{opt}})$ makes it much easier to find adequate codes.

In the following we derive some explicit formulas for the ultra-sparse and the moderately-sparse case which can be used to estimate storage capacities, storable patterns, and memory load for large values of n and k. Table 3.1 summarizes the asymptotic results for some example pattern activities k.

k	p_1	M	C	C^{sprs}	C^{cmpr}
c	0	$\sim n^{2-1/c}$	0	$1/c$	1
$c(\ln n)^d \quad , 0 < d < 1$	0	$\sim n^{2-1/(c(\ln n)^d)}/(\ln n)^{2d}$	0	0	1
$\mathrm{ld}\, n$	0.5	$(\ln 2) n^2/(\mathrm{ld}\, n)^2$	$\ln 2 \approx 0.69$	0	$\ln 2$
$c \ln n$	$\exp(-1/c)$	$\sim n^2/(\ln n)^2$	$\in (0; \ln 2)$	0	$\in (\ln 2; 1)$
$c(\ln n)^d \quad , 1 < d$	1	$\sim n^2 \ln \ln n/(\ln n)^{2d}$	0	0	1
$\sqrt{n}$	1	$0.5 n \ln n$	0	0.5	1
$cn^d \quad , 0 < d < 1$	1	$\sim n^{2-2d} \ln n$	0	d	1
$cn \quad , 0 < c < 1$	$\in (0; 1)$	$-\ln(1-p_1)/\ln(1-c^2)$	0	0	0

Table 3.1: Asymptotic results for memory load p_1, storable patterns M, and storage capacities C (classical), C^{sprs} (sparse matrix representation), and C^{cmpr} (optimal matrix compression) for different pattern activities $k(n)$ when requiring high-fidelity.

3.2.1 Ultra-sparse patterns

For sub-logarithmic pattern activity k with $k/\mathrm{ld}\, n \rightarrow 0$ we have $p_1 \rightarrow 0$, and with $-\ln(1-p_1) \approx p_1$ and $I(p_1) \approx -p_1 \mathrm{ld}\, p_1$ we obtain from eqs. 3.9, 3.10, 3.12, 3.14, and 3.15

$$M \approx \epsilon^{1/k} \cdot \frac{n^{2-1/k}}{k^{2-1/k}} \tag{3.17}$$

$$C \approx \epsilon^{1/k} \cdot \frac{\mathrm{ld}\,(n/k)}{n^{1/k} \cdot k^{1-1/k}} \rightarrow 0 \tag{3.18}$$

$$C^{\mathrm{sprs}} \approx \frac{1 - \log(k)/\log n}{k} \rightarrow 1/k \tag{3.19}$$

$$C^{\mathrm{cmpr}} \approx \frac{1}{1 - \frac{\ln \epsilon}{\ln(n/k)}} \rightarrow 1 \tag{3.20}$$

We can still store a very large number of ultra-sparse patterns with M being almost quadratic in n for large k. However, note that for a given n the maximal M is obtained for logarithmic k (cf. Fig.3.1c). The classical storage capacity C vanishes for large n, but for optimal compression we obtain $C^{\mathrm{cmpr}} \rightarrow 1$. For naive sparse matrix representation (see above) we obtain asymptotically $C^{\mathrm{sprs}} \rightarrow 1/k$. Thus we have non-zero values for C^{sprs} only for small constant k. For constant $k = 1$ we have trivially $C^{\mathrm{sprs}} \rightarrow 1$. However, this result is not very interesting since for $k = 1$ we have not really distributed storage. For $k = 1$ there are only n possible patterns to store, and the memory matrix degenerates to a look-up-table. In section 3.4.1 we will discuss that the Willshaw model of associative memory is more efficient than a look-up table only for $M/n \to \infty$.

3.2.2 Moderately-sparse patterns

For super-logarithmic but still sub-linear pattern activity k with $k/\mathrm{ld}\, n \to \infty$ and $k/n \to 0$ we have $p_1 \to 1$, and with $I(p_1) \approx -(1-p_1)\mathrm{ld}\,(1-p_1)$ and $1 - p_1 \approx -\ln p_1$ we obtain from eqs. 3.9, 3.10, 3.12, 3.14, and 3.15

$$\begin{aligned}
&(3.21) & p_1 &\approx n^{-1/k} \to 1 \\
&(3.22) & 1-p_1 &\approx \frac{1}{k}\ln\frac{n}{\epsilon k} \approx \frac{\ln(n/k)}{k} \to 0 \\
&(3.23) & M &\approx \frac{n^2}{k^2}(\ln k - \ln\ln\frac{n}{\epsilon k}) \\
&(3.24) & C &\approx (\ln k - \ln\ln\frac{n}{\epsilon k})\frac{\mathrm{ld}\,(n/k)}{k} \to 0 \\
&(3.25) & C^{\mathrm{sprs}} &\approx \frac{\ln k - \ln\ln\frac{n}{\epsilon k}}{(1-\frac{\ln\epsilon}{\ln(n/k)})\ln n} \approx \frac{\ln k}{\ln n} \\
&(3.26) & C^{\mathrm{cmpr}} &\approx \frac{\ln(n/k)}{k\cdot(1-p_1)} \approx \frac{1}{1-\frac{\ln\epsilon}{\ln(n/k)}} \to 1
\end{aligned}$$

Here we can still store many more moderately-sparse patterns than neurons ($M \gg n$) as long as $k \le \sqrt{n}$ (see eq.3.23, cf. table 3.4.1). The classical storage capacity C vanishes for large n, but for optimal compression we obtain $C^{\mathrm{cmpr}} \to 1$. Surprisingly, storage capacity 1 can be approached even for non-optimal matrix compression: For $k = n^d$ and $0 < d < 1$ we obtain from eq.3.25 $C^{\mathrm{sprs}} \to d$. However, the convergence is extremely slow (Fig.3.2d), and for $d > 0.5$ we obtain asymptotically only $M < n$ (see eq.3.23, cf. table 3.1) where simple look-up tables are more efficient than neural associative memories (see section 3.4.1).

As can be seen in Fig. 3.2b, both C^{cmpr} and C^{sprs} exhibit local maxima at $k_{\mathrm{opt}}^{\mathrm{cmpr}}$ and $k_{\mathrm{opt}}^{\mathrm{sprs}}$ for $k > \mathrm{ld}\, n$. In appendix B.4.2 these maxima are computed using C from eq.3.12 since the approximations of eq.3.25 and eq.3.26 exhibit no (or not the correct) local optimum. The asymptotic optima are approximately

$$\begin{aligned}
&(3.27) & k_{\mathrm{opt}}^{\mathrm{sprs}} &\sim n\cdot(e^{\sqrt{-\ln\epsilon}})^{-\sqrt{\ln n}} \\
&(3.28) & k_{\mathrm{opt}}^{\mathrm{cmpr}} &\sim n^{1-\frac{-\ln\epsilon-\sqrt{-\ln\epsilon}}{-\ln\epsilon-1}}.
\end{aligned}$$

Note that $k_{\mathrm{opt}}^{\mathrm{sprs}}$ grows faster than n^d for any $d < 1$, and therefore also $C^{\mathrm{sprs}}(k_{\mathrm{opt}}^{\mathrm{sprs}}) \to 1$.

3.2.3 Non-sparse patterns

The fact that matrix compression exhibit high storage capacities for all sub-linear pattern activities $k(n)$ leads to the question if we can also obtain a high storage capacity for non-sparse patterns, i.e., for linear $k(n)$. For a constant $0 < c < 1$ and $k = cn$ we have to demand $\epsilon(n) \to 0$ to still fulfill the hifi-requirement (see section 3.1.2). If we choose for example $\epsilon = 1/\ln n$ we can still use eqs.3.9 and 3.10 to estimate maximal matrix load p_1 and number M of storable patterns. If the binomial approximation of the error probability p_{01} (see eq.3.6) is still correct for linear $k = cn$

we would obtain with eqs. 3.8 and 3.22

$$M \approx \frac{\ln(1-p_1)}{\ln(1-c^2)} \approx \frac{\ln k}{-\ln(1-c^2)} \to \infty \tag{3.29}$$

$$C := \frac{C_A}{n^2} \approx \frac{MI(c)}{n} \approx \frac{I(c)}{-\ln(1-c^2)} \cdot \frac{\ln k}{n} \to 0 \tag{3.30}$$

$$C^{\mathrm{sprs}} := \frac{C}{(1-p_1)\mathrm{ld}\, n} \approx \frac{cI(c)}{\ln(1-c^2)\cdot \mathrm{ld}\,(\epsilon c)} \cdot \frac{\mathrm{ld}\, k}{\mathrm{ld}\, n} \to 0 \tag{3.31}$$

$$C^{\mathrm{cmpr}} \approx \frac{C}{-(1-p_1)\mathrm{ld}\,(1-p_1)} \approx \frac{cI(c)}{\ln(1-c^2)\cdot \mathrm{ld}\,(\epsilon c)} \to 0 \tag{3.32}$$

$$\tag{3.33}$$

Note that for small but constant ϵ both C^{sprs} and C^{cmpr} would converge towards a positive value $C_\infty(c) := cI(c)/(\ln(1-c^2)\mathrm{ld}\,(\epsilon c)) \in (0;1)$. While $C_\infty \to 0$ for $c \to 1$ we would still have $C_\infty \to 1$ for $c \to 0$.

Since we have chosen $\epsilon \to 0$ just for the sake of high-fidelity (see section 3.1.2) while the estimation of the capacity C_A should still be correct for small but constant ϵ (cf. appendix B.1), these results suggest that we could obtain a high asymptotic storage capacity also for small constant ϵ and linear k.

However, in section 3.6 it is shown that the classical binomial analysis is valid only for sublinear $k(n)$ and large n. For finite n the binomial approximation of eq.3.6 can be very bad even for logarithmic $k = \log(n)$. In appendix B.4.3 it is shown without using the binomial approximation that for linear $k = cn$ asymptotically indeed $C \to 0$, $C^{\mathrm{sprs}} \to 0$, and $C^{\mathrm{cmpr}} \to 0$ even for constant ϵ.

3.3 Fault tolerance

In the last two sections we have investigated the storage capacity for different variants of the Willshaw model. For this we assumed perfect address patterns. In this section we will investigate the decrease of storage capacity if the address patterns used for retrievals deviate from the original ones.

3.3.1 Missing ones in the address patterns

Here we will investigate the case of missing ones. Let us assume that an address pattern used for retrieval contains a fraction $\lambda \in (0;1]$ of the k ones of the original address pattern (but no false ones). To still achieve good retrieval results according to the hifi-requirement (see section 3.1.2) we have to store fewer patterns than before. With the same arguments as in section 3.1.2 we can increase the memory load p_1 by storing patterns until $k_{1,\mathrm{hifi}}(p_1) = \lambda \cdot k$ (cf. eq.3.7). Analogously to eq.3.9 we obtain a *maximal* memory load $p_{1,\max}(n,k,\epsilon,\lambda)$. In the following we will write simply $p_{1,\lambda}$ for $p_{1,\max}(n,k,\epsilon,\lambda)$, and p_1 for $p_{1,\max}(n,k,\epsilon,1)$. Thus

$$p_{1,\lambda} = (\epsilon k/n)^{1/(\lambda k)} = p_1^{1/\lambda}. \tag{3.34}$$

From eq.3.10 we see that now we can store only

$$M_\lambda \approx -\frac{n^2}{k^2}\ln(1-p_{1,\lambda}) \tag{3.35}$$

patterns (where we write simply M_λ for $M_{\max}(n,k,\epsilon,\lambda)$ and M for $M_{\max}(n,k,\epsilon,1)$). This is only the fraction

$$m_\lambda := \frac{M_\lambda}{M} \approx \frac{\ln(1-p_{1,\lambda})}{\ln(1-p_1)}. \tag{3.36}$$

of the number of storable patterns for perfect address patterns. Thus we see from eqs. 3.8 and 3.12 that also the classical storage capacity C is reduced to

$$C_\lambda \quad := \quad m_\lambda \cdot C \tag{3.37}$$

where we write simply C_λ for $C(n, k, \epsilon, \lambda)$ and C for $C(n, k, \epsilon, 1)$.

Maximal storage capacity for uncompressed memory matrix

One interesting question is about the maximal possible storage capacity for given n, ϵ, and λ, i.e., when maximizing over k, or equivalently, over $p_{1,\lambda}$ (or $p_1 = {p_{1,\lambda}}^{\lambda}$). Asymptotically for large n we obtain from eqs. 3.37, 3.36, and 3.13

$$C_\lambda \quad := \quad m_\lambda \cdot C \approx \mathrm{ld}\,(p_1) \cdot \ln(1 - p_{1,\lambda}) = \lambda \mathrm{ld}\,(p_{1,\lambda}) \cdot \ln(1 - p_{1,\lambda}). \tag{3.38}$$

We see that maximizing C_λ with respect to $p_{1,\lambda}$ for given λ is equivalent to maximizing C (see eq. 3.13). The maximum occurs for $p_{1,\lambda} = 0.5$ corresponding to $p_1 = 0.5^\lambda$, and we can write

$$C(n, \epsilon, \lambda) := \max_k C(n, k, \epsilon, \lambda) \quad \rightarrow \quad \lambda \cdot \ln 2 \tag{3.39}$$

If we denote by $C(q_1)$ the asymptotic storage capacity for asymptotic matrix load q_1 (cf. Fig.3.2c), and analogously $C_\lambda(q_1)$ when considering fault tolerance according to parameter λ we can write more generally

$$C_\lambda(q_1) = \lambda C(q_1). \tag{3.40}$$

Thus addressing using only a fraction λ of the ones in the original pattern decreases the classical asymptotic storage capacity by factor λ. In the following we will see that this is a very general result that is also true in a similar sense if the memory matrix is optimally compressed.

Storage capacity for optimal matrix compression

For given n, k, and ϵ, requiring fault tolerance according to parameter λ leads to a decrease in maximal memory load (see eq. 3.34), number of storable patterns (see eq. 3.35) and storage capacity (see eq. 3.37). The decrease in memory load will also alter the compressibility of the memory matrix. For optimal matrix compression (see section 3.2) we can write (cf. eq.3.15)

$$i_\lambda \quad := \quad \frac{I(p_{1,\lambda})}{I(p_1)}. \tag{3.41}$$

Note that $i_\lambda < 1$ for $p_1 < 0.5$, but usually $i_\lambda > 1$ for $p_1 > 0.5$. The latter occurs for moderately sparse patterns (with super-logarithmic k, see section 3.2.2) and implies that although fewer patterns are stored *more* physical memory is required. Now we obtain finally for the storage capacity $C_\lambda^{\mathrm{cmpr}}$ of the compressed Willshaw model (cf.eq. 3.15)

$$c^{\mathrm{cmpr}}(\lambda, p_1) \quad := \quad \frac{m_\lambda}{i_\lambda}, \tag{3.42}$$

$$C_\lambda^{\mathrm{cmpr}} \quad := \quad \frac{C_\lambda}{I(p_{1,\lambda})} = c^{\mathrm{cmpr}}(\lambda, p_1) \cdot C^{\mathrm{cmpr}}. \tag{3.43}$$

Thus the factor c^{cmpr} expresses the change of C^{cmpr} due to the requirement of fault tolerance.

Figure 3.3 illustrates the impact of requiring fault tolerance for ultra-sparse patterns ($p_1 < 0.5$), classically sparse patterns ($p_1 = 0.5$), and moderately sparse patterns ($p_1 > 0.5$). For the ultra-sparse and classically-sparse regime the number of storable patterns M_λ decreases very rapidly with decreasing λ, while the decrease is much slower for the moderately sparse patterns (Fig. 3.3a).

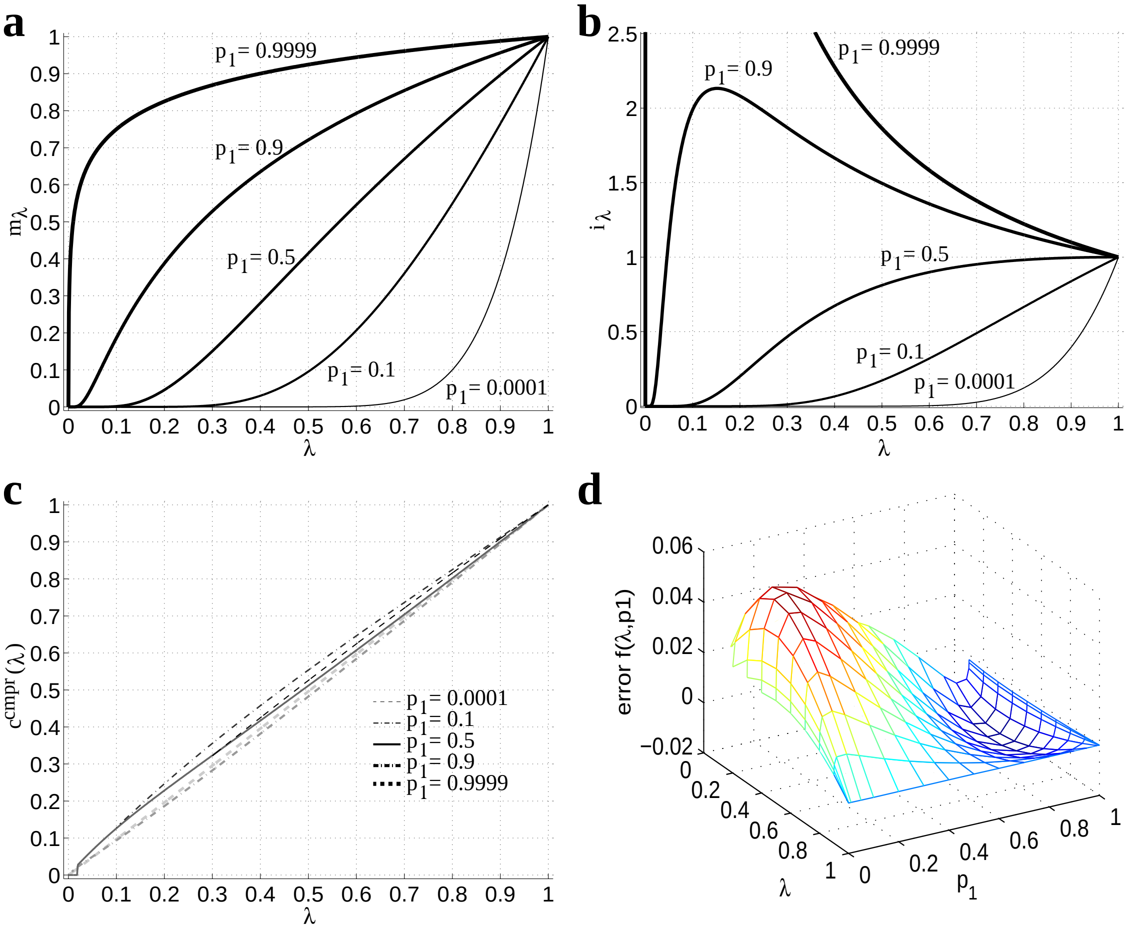


Figure 3.3: Impact of missing ones in the address pattern on the number of storable patterns, synaptic information, and storage capacity for different p_1. Address pattern are assumed to contain $\lambda \cdot k$ out of the k original ones, but no false ones. p_1 is the maximal matrix load for $\lambda = 1$. For $\lambda < 1$ the matrix load will be decreased to $p_{1,\lambda} = {p_1}^{1/\lambda}$. For $\epsilon = 0.01$ and $n = 10^6$ neurons we would obtain (approximately) $p_1 = 0.0001, 0.1, 0.5, 0.9, 0.9999$ for pattern activities $k = 2, 7, 22, 129, 70000$. **a** : The fraction $m_\lambda := M_\lambda / M$ vs. λ . While the number of storable patterns remains relatively constant for $p_1 > 0.5$, m_λ decreases rapidly with λ for $p_1 \leq 0.5$. **b** : The fraction $i_\lambda := I(p_{1,\lambda})/I(p_1)$ vs. λ exhibits the inverse behavior of m_λ. **c** : For all values of p_1 we have $c^{\mathrm{cmpr}}(\lambda, p_1) := m_\lambda / i_\lambda \approx \lambda$ (see text). **d** : The error $f(\lambda, p_1) := c^{\mathrm{cmpr}}(\lambda, p_1) - \lambda$ of approximating $c^{\mathrm{cmpr}}(\lambda, p_1)$ by λ is small ($-0.02 < f < 0.06$) and even vanishes for $p_1 \to 0$ and $p_1 \to 1$.

For the compressibility the situation is inverted (Fig. 3.3b). While for ultra-sparse and classically-sparse patterns the synapse information $I(p_{1,\lambda})$ decreases rapidly with decreasing λ (i.e., the compressibility increases rapidly), $I(p_{1,\lambda})$ even increases for the moderately-sparse patterns.

Most interestingly the two effects seem to compensate for each other quite well (Fig. 3.3c). Indeed, we can propose

$$c^{\mathrm{cmpr}}(\lambda, p_1) \quad = \quad \lambda + f(\lambda, p_1) \approx \lambda \tag{3.44}$$

for a small error function f. We can numerically verify $-0.02 < f(\lambda, p_1) < 0.06$ for $\lambda, p_1 \in (0; 1)$ (see Fig.3.3d). For $p_1 \to 0$ and $p_1 \to 1$ we prove that $f(\lambda, p_1) \to 0$.

For $p_1 \to 0$ (corresponding to ultra-sparse patterns with sub-logarithmic $k(n)$, see section

3.2.1) we have also $p_{1,\lambda} \to 0$ and therefore from eq. 3.36

$$m_\lambda \approx \frac{p_{1,\lambda}}{p_1} = p_1^{(1-\lambda)/\lambda} \to 0, \tag{3.45}$$

and from eq.3.41

$$i_\lambda \approx \frac{p_1^{1/\lambda} \cdot (1/\lambda) \cdot \mathrm{ld}\, p_1}{p_1 \cdot \mathrm{ld}\, p_1} = \frac{p_1^{(1-\lambda)/\lambda}}{\lambda} \to 0, \tag{3.46}$$

and therefore $c^{\mathrm{cmpr}}(\lambda, p_1) \approx \lambda$.

By applying L'Hospital's rule one can easily verify that for given λ and $p_1 \to 1$ (corresponding to moderately-sparse patterns with super-logarithmic but still sub-linear $k(n)$, see section 3.2.2) we obtain from eq. 3.36

$$m_\lambda \approx \frac{\ln(1-p_{1,\lambda})}{\ln(1-p_1)} \to 1. \tag{3.47}$$

Similarly we can apply L'Hospital's rule on eq. 3.41 and find

$$c^{\mathrm{cmpr}}(\lambda, p_1) \approx \frac{\ln(1-p_{1,\lambda})}{\ln(1-p_1)} \cdot \frac{(1-p_1) \cdot \mathrm{ld}\,(1-p_1)}{(1-p_{1,\lambda}) \cdot \mathrm{ld}\,(1-p_{1,\lambda})} = \frac{1-{p_1}^{1/\lambda}}{1-p_1} \to \lambda. \tag{3.48}$$

Consequently we have also $i_\lambda \to 1/\lambda$ for $p_1 \to 1$.

In summary, using only a fraction λ of the ones contained in the original address patterns reduces the storage capacity by a factor of λ. This seems to be a very general result which has been shown to be asymptotically true in this section for storage capacities $C_\lambda(n)$ (see eq. 3.39), $C_\lambda(p_1)$ (see eq. 3.40), $C_\lambda^{\mathrm{cmpr}}(n)$, and $C_\lambda^{\mathrm{cmpr}}(n,k)$ (see eqs. 3.43 and 3.44).

3.3.2 False ones in the address patterns

The analysis of missing ones in the address pattern was quite easy. Unfortunately, the analysis of false ones is more complicated. Moreover, the Willshaw retrieval strategy (threshold equal to the number of active units in the address patterns) cannot be applied if the address pattern contains false ones.

For the classical Willshaw model Sommer and Palm [171] have derived exact formulas for the error probabilities in the retrieved pattern given the error probabilities in the address pattern (i.e., numbers of false and missing ones) and the threshold. Thus the threshold can be chosen somewhere between k and the number of ones in the address pattern to minimize the error probabilities in the retrieval result.

However, if the error probabilities in the address pattern are not known then it is impossible to choose a priori an appropriate threshold. Instead it is necessary to adjust a posteriori the threshold repeatedly until the number of ones in the retrieval result matches the desired pattern activity.

Even if the error probabilities in the address pattern are known it is difficult to derive general results about the impact on number of storable patterns and storage capacity as has been done in section 3.3.1 for missing ones. While the formulas of Sommer and Palm [171] can be used to numerically compute m_λ, i_λ, and $c^{\mathrm{cmpr}}(\lambda, p_1)$ for certain given n and k they still lack appropriate approximations necessary for more general results.

As we will see in the next chapter spiking associative memory can have the nice property to apply *implicitly* the Willshaw retrieval strategy even if the address pattern contains false ones. Therefore we leave out a further analysis of false ones at this point and refer to section 4.4 where we try to investigate the impact of false ones on the retrieval results of spiking associative memory.

3.4 Retrieval efficiency for sequential implementations

So far we were concerned with the storage capacity and fault tolerance of the Willshaw associative memory when storing M binary pattern vectors of length n, each containing exactly k ones. Another important question is *how fast* the information can be retrieved. A straight-forward measure for sequential implementations is the **information current** or **information flow**

$$f_I \quad := \quad \frac{I_{\mathrm{ret}}}{t_{\mathrm{ret}}} = \frac{C_A}{M t_{\mathrm{ret}}} \tag{3.49}$$

defined as the fraction of the information $I_{\mathrm{ret}} := C_A/M$ obtained in a retrieval over the time (or number of steps) t_{ret} required for the retrieval, where C_A is the totally stored information, or absolute storage capacity (see section 3.1.2 and appendix B.1). Thus the information flow expresses the average number of retrieved information units per time step.

For example, for a sequential implementation of the classical (uncompressed) Willshaw model we require asymptotically $t_{\mathrm{ret}} = k \cdot n$ steps (with typically logarithmic $k = c \ln n$): for each of the k address units all n entries in the corresponding row of the memory matrix must be tested to obtain the membrane potentials. The subsequent threshold operation requires n further steps that can be neglected asymptotically. The retrieved information per pattern is $I_{\mathrm{ret}} \approx n \cdot I(k/n) \approx k \cdot \mathrm{ld}\, n$ (cf. eq. 3.8), and thus we have asymptotically $f_I = \mathrm{ld}\, n/n$ for the Willshaw model.

The information flow f_I will depend non-trivially on the absolute storage capacity C_A (i.e., the total amount of information stored in the associative memory). To allow comparison of f_I among different pattern sizes n, pattern activities k, and even different models (e.g., compressed vs. uncompressed) we require f_I to be normalized with respect to a *reference model* of associative memory. For this reference model we use simply the classical uncompressed Willshaw model with parameters n, k, M, and $\lambda = 1$ such that f_I is maximal for given C_A as in the alternative model under consideration.

For the uncompressed Willshaw model we have $C_A = C \cdot n^2$ with the normalized storage capacity C (see eq.3.12), and thus $n = \sqrt{C_A/C}$. From this we get for given C_A a reference information flow of $f_I^{\mathrm{ref}} = \mathrm{ld}\, n/n = 0.5 \mathrm{ld}\,(C_A/C)/\sqrt{C_A/C}$. We observe that f_I^{ref} is asymptotically maximal for the classical maximum $C = \ln 2$ corresponding to $k = \mathrm{ld}\, n$ and $p_1 = 0.5$ (see section 3.1.2). This motivates the definition of the *retrieval efficiency*

$$E \quad := \quad \frac{f_I}{f_I^{\mathrm{ref}}} \approx 2\sqrt{\ln 2} \cdot \frac{\sqrt{C_A}}{\ln C_A} \cdot f_I \tag{3.50}$$

For example, if we find that a certain model exhibits $E = 5$ this means that it is five times faster in retrieving one bit of information than the classical Willshaw model storing the same total amount of information. In the following we will compute the retrieval efficiency for different models of information storage.

3.4.1 Look-up-tables with $M = n^d$

As a first example, consider the case of a simple *look-up table* to store M pattern pairs. This will result in a $M \times 2n$ matrix for pattern size n, i.e. each line of a matrix contains a pattern pair. To allow fault tolerant retrieval for each stored index pattern the Hamming distance to the address pattern must be computed which requires $t_{\mathrm{ret}} = M \cdot n$ steps. For $k = n/2$ (or equivalently $p_1 = 0.5$, i.e. non-sparse patterns) we have $I_{\mathrm{ret}} = n$, and thus for such a look-up table we have $f_I = 1/M$. Further we have $C_A = M \cdot n$ and if we require to store many patterns, e.g., $M = n^d$ for a constant d, we find $f_I = n^{-d}$ and $C_A = n^{d+1}$ and thus we obtain from eq.3.50 a retrieval efficiency

$$E^{\mathrm{lut}} \quad = \quad \frac{2\sqrt{\ln 2}}{d+1} \cdot \frac{n^{(1-d)/2}}{\ln n}. \tag{3.51}$$

Thus for $d > 1$, i.e., if we require to store many patterns, the classical Willshaw model is more efficient than a look-up table (cf. [132]).

3.4.2 Classical Willshaw model

What is the retrieval efficiency for the classical uncompressed Willshaw model? Obviously we have $E = 1$ for $k = \mathrm{ld}\, n$ and $\lambda = 1$. In the following we consider the situation for general $k(n)$ and fault tolerance according to parameter λ. The latter allows not only addressing with $(1-\lambda)k$ missing ones (see section 3.3.1) but also the addressing with up to κk additional false ones (which is, however, more difficult to analyze; cf. section 4.4). We assume that the address pattern contains $\alpha \cdot k$ one-entries where obviously $\alpha \in [\lambda; \lambda + \kappa]$ (if both missing and false ones occur then α must lie somewhere *inside* the interval). We have

$$t_{\mathrm{ret}} = \alpha k n \tag{3.52}$$

$$I_{\mathrm{ret}} \approx k \,\mathrm{ld}\, n \tag{3.53}$$

$$f_I \approx \frac{\mathrm{ld}\, n}{\alpha n} \tag{3.54}$$

$$C_A \approx C_\lambda \cdot n^2 = \lambda \cdot n^2 \cdot \mathrm{ld}\,(p_1) \cdot \ln(1-p_1) \tag{3.55}$$

where for C_A we used eq. 3.38, and $p_1 := p_{1,\max}(n,k,\epsilon,\lambda) = (\epsilon k/n)^{1/(\lambda k)}$ (cf. eq. 3.34). From the definition eq. 3.50 we obtain for large $n \to \infty$ and constant λ, α, and p_1 (corresponding to logarithmic pattern activity $k = c\,\mathrm{ld}\, n$, see section 3.1.2) with $\ln(C_A) \approx 2 \ln n$,

$$E \approx \frac{1}{\ln 2} \cdot \frac{\sqrt{\lambda}}{\alpha} \cdot \sqrt{\mathrm{ld}\,(p_1) \cdot \mathrm{ld}\,(1-p_1)}. \tag{3.56}$$

Thus the retrieval efficiency is also constant and therefore the corresponding optimal information flow differs only by a constant factor.

3.4.3 Compressed Willshaw model with ultra-sparse patterns

For ultra-sparse patterns with sub-logarithmic pattern activity k we have $p_1 = (\epsilon k/n)^{1/(\lambda k)} \to 0$ asymptotically for large n and fixed λ (cf. eq. 3.34). In the following we compute the retrieval efficiency for the optimally compressed Willshaw model (see section 3.2) applying Golomb coding and decoding of the memory matrix. The Golomb compressed matrix essentially contains information about the interval between two active synapses in a matrix row, and decoding of the position of an active synapse requires only (almost) constant time a ([57], see appendix A.3.2). Thus we obtain for large n and with address pattern size αk for constant α (see section 3.4.2)

$$t_{\mathrm{ret}} = \alpha \cdot k \cdot a \cdot p_1 \cdot n = \alpha \cdot \epsilon^{1/(\lambda k)} \cdot a \cdot k^{1+1/(\lambda k)} \cdot n^{1-1/(\lambda k)} \tag{3.57}$$

$$I_{\mathrm{ret}} \approx k \cdot \mathrm{ld}\, n \tag{3.58}$$

$$f_I \approx \frac{1}{\alpha \cdot a \cdot p_1} \cdot \frac{\mathrm{ld}\, n}{\alpha \cdot n} = \frac{\mathrm{ld}\, n}{\alpha \cdot a \cdot \epsilon^{1/(\lambda k)} \cdot k^{1/(\lambda k)} \cdot n^{1-1/(\lambda k)}} \tag{3.59}$$

$$C_A \approx \lambda \cdot I(p_1) \cdot n^2 \approx -\lambda \cdot n^2 \cdot p_1 \cdot \mathrm{ld}\, p_1 \approx \frac{n^2 \cdot p_1 \cdot \mathrm{ld}\, n}{k} \tag{3.60}$$

$$= \frac{\epsilon^{1/(\lambda k)} \cdot n^{2-1/(\lambda k)} \cdot \mathrm{ld}\, n}{k^{1-1/(\lambda k)}}. \tag{3.61}$$

Note that for constant k we can have a sub-linear retrieval time t_{ret}. For C_A we have used that the normalized storage capacity for optimal compression is λ asymptotically (see section 3.3.1). With $\ln C_A \approx (2 - \frac{1}{\lambda k}) \cdot \ln n$ we obtain from the definition eq. 3.50

$$E \approx \frac{1}{(1 - 1/(2\lambda k))} \cdot \frac{1}{\alpha \cdot a} \cdot \sqrt{\frac{n^{1/(\lambda k)} \cdot \ln n}{\epsilon^{1/(\lambda k)} \cdot k^{1+1/(\lambda k)}}}. \tag{3.62}$$

Taking logarithms we see that $\ln(E) \sim (\ln n)/(\lambda k) + \ln \ln n - \ln k \to \infty$ and therefore also $E^{\mathrm{cmpr}} \to \infty$ for *any* sub-logarithmic $k(n)$. Thus the compressed Willshaw model is more efficient than the classical Willshaw model for ultra-sparse patterns.

3.4.4 Compressed Willshaw model with moderately-sparse patterns

For moderately-sparse patterns with super-logarithmic (but still sub-linear) pattern activity k we have $p_1 = (\epsilon k/n)^{1/(\lambda k)} \to 1$ asymptotically for large n and fixed λ (cf. eq.3.34). In the following we compute the retrieval efficiency for the optimally compressed Willshaw model (see section 3.2) applying Golomb coding and decoding of the memory matrix. The Golomb compressed matrix essentially contains information about the interval between two *inactive* synapses in a matrix row, and decoding of the position of an inactive synapse requires only (almost) constant time a ([57], see appendix A.3.2). Thus we obtain for large n and with address pattern size αk for constant α (see section 3.4.2) using $1 - p_1 \approx -\ln p_1 \approx \ln(n/k)/(\lambda k)$ (cf. section 3.2.2)

$$\begin{aligned}
t_{\mathrm{ret}} &= \alpha \cdot k \cdot a \cdot (1-p_1) \cdot n \approx a \cdot \frac{\alpha}{\lambda} \cdot n \cdot \ln(n/k) \tag{3.63}\\
I_{\mathrm{ret}} &\approx k \cdot \mathrm{ld}\,(n/k) \tag{3.64}\\
f_I &\approx \frac{1}{\ln 2} \cdot \frac{\lambda}{\alpha \cdot a} \frac{k}{n} \tag{3.65}\\
C_A &\approx \lambda \cdot I(p_1) \cdot n^2 \approx -\lambda \cdot n^2 \cdot (1-p_1) \cdot \mathrm{ld}\,(1-p_1) \tag{3.66}\\
&\approx \frac{\ln(n/k) \cdot \ln(k)}{\ln 2} \cdot \frac{n^2}{k} \tag{3.67}
\end{aligned}$$

For C_A we have used that the normalized storage capacity for optimal compression is λ asymptotically (see section 3.3.1). With $\ln C_A \approx \ln(n^2/k)$ we obtain from the definition eq. 3.50

$$\begin{aligned}
E &\approx \frac{1}{\ln 2} \cdot \frac{\lambda}{\alpha \cdot a} \cdot \frac{\sqrt{\ln(n/k)}}{\ln(n/\sqrt{k})} \cdot \sqrt{k \cdot \ln k} \tag{3.68}\\
&\sim \sqrt{\frac{k \ln k}{\ln n}} \tag{3.69}
\end{aligned}$$

Since already $\sqrt{k/\ln n} \to \infty$ we have $E^{\mathrm{cmpr}} \to \infty$ for $n \to \infty$ and *any* super-logarithmic but sub-linear pattern activity $k(n)$. Thus the compressed Willshaw model is more efficient than the classical Willshaw model also for moderately-sparse patterns.

3.5 Retrieval efficiency for parallel implementations

For parallel implementations of associative memory we can define the information flow f_I analogously as for sequential implementations (see section 3.4). To obtain the information flow through individual processors $f_{I,N}$ we can normalize f_I with respect to the number of processors.

$$\begin{aligned}
f_I &:= \frac{I_{\mathrm{ret}}}{t_{\mathrm{ret}}} \tag{3.70}\\
f_{I,N} &:= \frac{I_{\mathrm{ret}}}{t_{\mathrm{ret}} \cdot \#\mathrm{processors}} \tag{3.71}
\end{aligned}$$

As an example we compute the information flow for a parallel implementation of the classical Willshaw model with parameters n (number of neurons), k (pattern activity), M (number of stored patterns), and $\lambda = 1$ (fault tolerance parameter). Using n processors (i.e., one processor per neuron) it is possible to perform a retrieval in only $t_{\mathrm{ret}} = k$ steps: For each of the k address units all n neurons have to update their neuron potentials in parallel. After that the threshold operation is performed in parallel in constant time. For typically $k = c \ln n$ and large n, the information obtained from one retrieval is still $I_{\mathrm{ret}} \approx k \mathrm{ld}\, n$, and the totally stored information is still $C_A = C \cdot n^2$ which can be transformed to $n = \sqrt{C_A/C}$. Thus we have $f_I \approx \mathrm{ld}\, n$ and $f_{I,N} \approx \mathrm{ld}\, n/n$ for this case.

In order to compare different models we require the information flow to be normalized with respect to a *reference model* of associative memory. Similar as in section 3.4 we use the classical uncompressed Willshaw model with parameters n, k, M, and $\lambda = 1$ such that f_I is maximal for given C_A. From $C_A = C \cdot n^2$ (see eq.3.12) we obtain $n = \sqrt{C_A/C}$, and thus $f_I^{\text{ref}} \approx \mathrm{ld}\, n = 0.5\mathrm{ld}\,(C_A/C)$ and $f_{I,N}^{\text{ref}} \approx \mathrm{ld}\, n/n = 0.5\mathrm{ld}\,(C_A/C)/\sqrt{C_A/C}$. Using the classical maximum $C = \ln 2$ we can define in analogy to eq. 3.50 the retrieval efficiency E and the normalized retrieval efficiency E_N,

$$E := \frac{f_I}{f_I^{\text{ref}}} \approx \frac{2\ln 2}{\ln C_A} \cdot f_I \tag{3.72}$$

$$E_N := \frac{f_{I,N}}{f_{I,N}^{\text{ref}}} \approx 2\sqrt{\ln 2} \cdot \frac{\sqrt{C_A}}{\ln C_A} \cdot f_{I,N}. \tag{3.73}$$

Note that the definition of E_N is the same as for the sequential implementation (cf. eq.3.50).

3.5.1 Classical Willshaw model

For the parallel implementation of the classical uncompressed Willshaw model we have obviously $E = E_N = 1$ for $k = \mathrm{ld}\, n$ and $\lambda = 1$. For general $k(n)$, fault tolerance according to a constant parameter λ (see section 3.3.1) and an address pattern containing $\alpha \cdot k$ one-entries (cf. section 3.4.2), we have for large n

$$t_{\text{ret}} = \alpha \cdot k \tag{3.74}$$

$$I_{\text{ret}} \approx k \cdot \mathrm{ld}\, n \tag{3.75}$$

$$f_I \approx \frac{\mathrm{ld}\, n}{\alpha} \tag{3.76}$$

$$f_{I,N} \approx \frac{\mathrm{ld}\, n}{\alpha \cdot n} \tag{3.77}$$

$$C_A \approx C_\lambda \cdot n^2 = \lambda \cdot n^2 \cdot \mathrm{ld}\,(p_1) \cdot \ln(1 - p_1) \tag{3.78}$$

where C_A is the same as in eq. 3.55 with $p_1 := p_{1,\max}(n, k, \epsilon, \lambda) = (\epsilon k/n)^{1/(\lambda k)}$ (cf. eq. 3.34). From the definitions of eqs. 3.72 and 3.73 we obtain for large $n \to \infty$ and constant λ, α, and p_1 (corresponding to logarithmic pattern activity $k = c\mathrm{ld}\, n$, see section 3.1.2) using $\ln(C_A) \approx 2\ln n$,

$$E \approx \frac{1}{\alpha} \tag{3.79}$$

$$E_N \approx \frac{1}{\ln 2} \cdot \frac{\sqrt{\lambda}}{\alpha} \cdot \sqrt{\mathrm{ld}\,(p_1) \cdot \mathrm{ld}\,(1 - p_1)}. \tag{3.80}$$

Thus the retrieval efficiency is also constant and therefore the corresponding optimal information flow differs again only by a constant factor.

3.5.2 Parallel implementation of the compressed Willshaw model

In the following we develop an efficient implementation of the compressed Willshaw model with parameters n (number of neurons), k (number of one-entries in a pattern), λ (fault tolerance parameter, cf. section 3.3.1) and $p_1 := p_{1,\max}(n, k, \epsilon, \lambda) = (\epsilon k/n)^{1/(\lambda k)}$ (cf. eq. 3.34) when the address pattern contains $\alpha \cdot k$ one-entries (cf. section 3.4.2).

In a *naive* parallel implementation of the compressed Willshaw model each processor would receive all the $\alpha \cdot k$ address indices and then compute the corresponding neuron potential. Thus the retrieval would require $t_{\text{ret}} = a \cdot \min(p_1, 1 - p_1) \cdot n$ steps (a is the time required to perform Golomb decoding of one active/inactive synapse),i.e., only factor $\alpha \cdot k$ better than the sequential implementation (cf. sections 3.4.3 and 3.4.4).

A more promising approach is to implement something analogous to the classical algorithm: Each processor should compute in only $\alpha \cdot k$ steps the potential of the corresponding neuron. Thus for each of the $\alpha \cdot k$ address neurons a processor must decide in constant time if the corresponding synapse is active. Since the memory matrix is Golomb compressed this cannot be done in a direct access as for the uncompressed case. To avoid searching through the whole compressed matrix row (representing essentially the index intervals between active/inactive synapses) which would require $\min(p_1, 1-p_1)n$ steps (for each of the $\alpha \cdot k$ address neurons), we can try to use a *look-up-table* for each neuron as illustrated in Fig.3.4. The idea is to divide the matrix row into compartments of length d. Thus each compartment contains

$$\# := d \cdot \min(p_1, 1-p_1) \tag{3.81}$$

active synapses on average (or inactive synapses for $p_{1,\lambda} > 0.5$). The look-up-table consists of two rows. To decide if the i-th synapse is active (inactive) we have to look into the i DIV d-th column. The value from the second row is the smallest index j of an active (inactive) synapse with $j \leq i$ DIV d, and the value from the first row the corresponding start index in the compressed matrix row. From this starting point we can reach the i-th synapse after skipping of $<$ # Golomb coded numbers. Thus to decide if the i-th synapse is active (inactive) we will require less than $a \cdot \#$ steps. Therefore the whole retrieval will require

$$t_{\text{ret}} = a \cdot \# \cdot \alpha \cdot k \tag{3.82}$$

steps which is almost as fast as for the uncompressed model if # is small (e.g., constant, cf. eq.3.74).

On the other hand we should try to keep the size of the look-up-table as small as possible which requires large #. More exactly, if we still want to obtain storage capacity 1 we require that the size $2 \cdot n^2/d \cdot \log n$ of the look-up-table is much smaller than the size $n^2 \cdot I(p_1)$ of the compressed memory matrix, i.e. for $n \to \infty$ we require

$$\frac{\min(p_1, 1-p_1) \cdot \log n}{\# \cdot I(p_1)} \to 0(!). \tag{3.83}$$

With $I_{\text{ret}} = k \text{ld}\,(n/k)$ and eq.3.82 we can write for the information flows

$$f_I = \frac{\text{ld}\,(n/k)}{\alpha \cdot a \cdot \#} \tag{3.84}$$

$$f_{I,N} = \frac{\text{ld}\,(n/k)}{\alpha \cdot a \cdot \# \cdot n}. \tag{3.85}$$

In the following we try to analyze the parallel implementation of the compressed Willshaw model for codes of different degrees of sparseness.

3.5.3 Compressed Willshaw model with ultra-sparse patterns

For ultra-sparse patterns with sub-logarithmic pattern activity $k(n)$ we have $p_1 = (\epsilon k/n)^{1/(\lambda k)} \to 0$ asymptotically for large n and fixed λ (cf. eq. 3.34; cf. section 3.2.1). With $I(p_1) \approx -p_1 \text{ld}\, p_1 \approx p_1 \cdot \text{ld}\,(n)/(\lambda k)$ for large n the requirement of eq. 3.83 writes

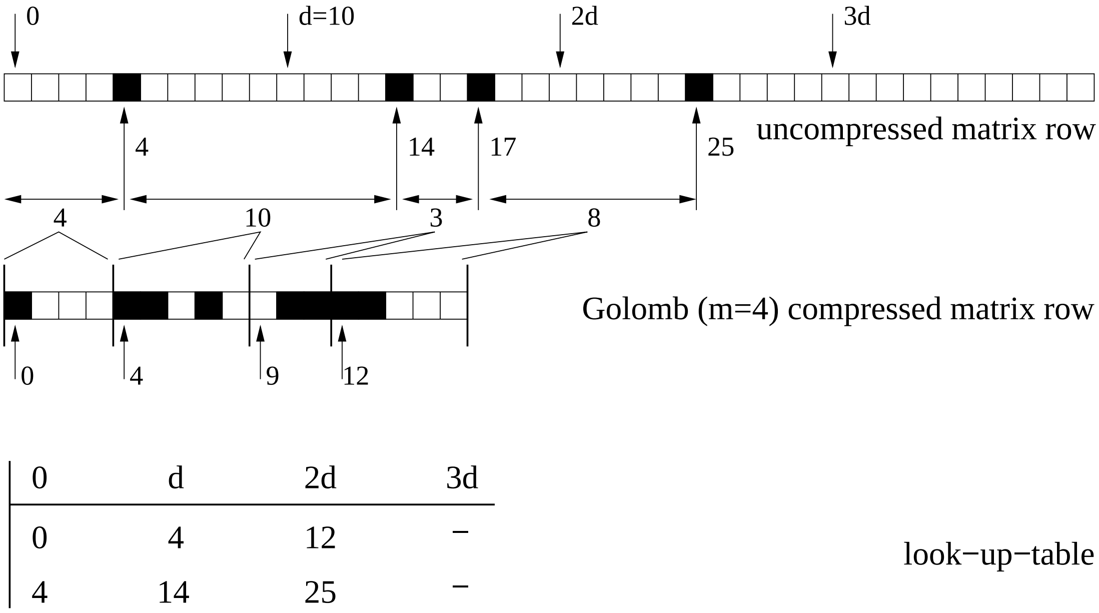
$$\frac{\lambda \cdot k}{\#} \to 0 \quad (!). \tag{3.86}$$

Thus constant # is not possible, unfortunately. This means that storage capacity 1 would require a # that increases faster than k. However, for constant k and large but constant # we can still obtain a significant storage capacity.

With C_A as in eq. 3.60, $\sqrt{C_A} \approx \epsilon^{1/(2\lambda k)} n^{1-1/(2\lambda k)} \sqrt{\text{ld}\, n}/k^{1/2-1/2\lambda k}$, and $\ln C_A \approx (2 - 1/(\lambda k)) \ln n$ we obtain from eqs. 3.72 and 3.73 with eqs. 3.84 and 3.85

$$E \approx \frac{1}{\alpha \cdot a \cdot \# \cdot (1 - 1/(2\lambda k))} \tag{3.87}$$

$$E_N \approx \frac{1}{\ln 2} \cdot \frac{\epsilon^{1/2k}}{(1 - 1/(2\lambda k))} \cdot \alpha \cdot a \cdot \# \cdot \sqrt{\frac{\ln n}{k^{1-1/(\lambda k)} \cdot n^{1/(\lambda k)}}}. \tag{3.88}$$



0	d	2d	3d
0	4	12	−
4	14	25	−

Figure 3.4: Illustration of the use of the look-up-table for Golomb-decoding of a matrix row. Top: The uncompressed matrix row. The sparse (e.g., one) entries are drawn in black (columns 4, 14, 17, 25). Middle: The Golomb-coded matrix row ($m = 4$, cf. appendix A.3.2) contains a compressed representation of the intervals between sparse entries in the matrix row (intervals 4, 10, 3, 8). Bottom: The look-up-table for Golomb-decoding. The matrix row is divided in compartments of length $d = 10$. For each compartment the look-up-table contains two entries: the entry in the first row is the index in the compressed matrix row corresponding to the first sparse entry in the relevant compartment. The second entry is the corresponding (absolute) index in the uncompressed matrix row. E.g., to find out if the i-th bit in the matrix row is active one can use the i DIV d-th column in the look-up-table to start searching in the compressed row with the first sparse entry in the relevant compartment. On average no more than $\# := d \cdot \min(p_1, 1 - p_1)$ further Golomb decoding steps (each requiring time a) are necessary.

For non-constant # both E and E_N vanish. This means that the classical Willshaw model is superior to the compressed Willshaw model for parallel implementation.

3.5.4 Compressed Willshaw model with moderately-sparse patterns

For moderately-sparse patterns with super-logarithmic but still sub-linear pattern activity $k(n)$ we have $p_1 = (\epsilon k/n)^{1/(\lambda k)} \to 1$ and therefore $I(p_1) \approx -(1 - p_1)\mathrm{ld}\,(1 - p_1)$. With $(1 - p_1) \approx -\ln p_1 \approx \ln(n/k)/(\lambda k)$ and $-\mathrm{ld}\,(1 - p_1) \approx \mathrm{ld}\,k$ the requirement of eq. 3.83 writes

$$\frac{\mathrm{ld}\,n}{\# \cdot \mathrm{ld}\,k} \to 0 \quad (!). \tag{3.89}$$

Again constant # is not possible, unfortunately. However, if k is a root of n then # is allowed to increase arbitrarily slowly in order to obtain asymptotically storage capacity 1. Even for constant but large # a significant storage capacity is still possible in this case.

With C_A as in eq. 3.67, $\sqrt{C_A} \approx n\sqrt{\ln(n/k)\ln(k)/(k \ln 2)}$, and $\ln C_A \approx 2\ln(n/\sqrt{k})$ we obtain from eqs. 3.72 and 3.73 with eqs. 3.84 and 3.85

$$E \approx \frac{\ln(n/k)}{\ln(n/\sqrt{k})} \cdot \frac{1}{\alpha \cdot a \cdot \#} \approx \frac{c_1}{\alpha \cdot a \cdot \#} \tag{3.90}$$

$$E_N \approx \frac{1}{\ln 2} \cdot \frac{\ln(n/k)}{\ln(n/\sqrt{k})} \cdot \frac{1}{\alpha \cdot a \cdot \#} \cdot \sqrt{\frac{\ln(n/k)\ln k}{k}} \tag{3.91}$$

$$\approx \frac{c_1}{\ln 2} \cdot \frac{1}{\alpha \cdot a \cdot \#} \cdot \sqrt{\frac{\ln(n/k)\ln k}{k}}, \tag{3.92}$$

for a constant c_1 (where $c_1 = 1$ if k is less than a root of n). Again for non-constant # both E and E_N vanish. This means that the classical Willshaw model is superior to the compressed Willshaw model for parallel implementation.

3.6 A critique of the classical binomial analysis

Almost all the classical results [204, 129, 134] as well as the analysis conducted in the last sections are based on the hypothesis that the true distribution of membrane potentials can be approximated by a binomial, at least asymptotically. In this section we will see why this assumption is justified only for large numbers of neurons and low address pattern activity.

3.6.1 The binomial approximation of the Willshaw distribution can be bad

Let us consider similarly as before the case of a Willshaw (hetero-)associative memory with n neurons and M stored pattern pairs, where each patterns contains exactly k one-entries and $n-k$ zero-entries. Further let us assume for retrieval that an *address* pattern contains λk ones of a previously stored pattern, and additionally κk randomly chosen false ones. Then the membrane potential of a neuron is determined by the number of ones in the relevant subcolumn of the memory matrix which contains $z := (\lambda+\kappa)k$ entries. For random patterns we will call the resulting potential distribution for a non-addressed neuron (which is not part of the pattern corresponding to the λk correct ones of the address pattern) the *Willshaw distribution* $W_{z,n,k,M}$.

Already in 1980 Palm derived a formula for the probability of a false one-entry in a retrieval based on the true distribution of the membrane potentials (see appendix in [129]). Then the probability density function of the Willshaw distribution has been determined in 1991 by Buckingham and Willshaw [31, 29], and was subsequently refined by Sommer and Palm [171]. However, the resulting formulas are not very convenient for the analysis of storage capacities and error probabilities, and therefore the classical binomial approximation B_{z,p_1} (with mean zp_1 and variance $zp_1(1-p_1)$) has been used so far, where the matrix load p_1 can be computed as a function of n, k, and M (see eq. 3.4).

Using a binomial distribution assumes that the active synapses are generated independently of each other. This is not true since storing of one pattern activates k^2 synapses simultaneously, causing dependencies within the memory matrix. Figure 3.5 illustrates some cases where the binomial approximation is quite bad. It is tempting to assign the observed discrepancy to finite size effects for finite n, k, and M. For example we may argue that the characteristic modulations in the histograms of membrane potential (cf. Fig.3.5a,b) are due to relatively large k and small M. Similarly, one could think that the massive underestimation of the true variance of the membrane potential distribution (see Fig.3.5c,d; note that the mean is still correct) origins from an address pattern activity z that is too large compared to the number of neurons n. Thus it appears reasonable to argue that for sub-linear k these effects can be neglected for large $n \to \infty$, and that requiring

$$(\lambda + \kappa)k \ll n \tag{3.93}$$

is sufficient to justify the binomial approximation.

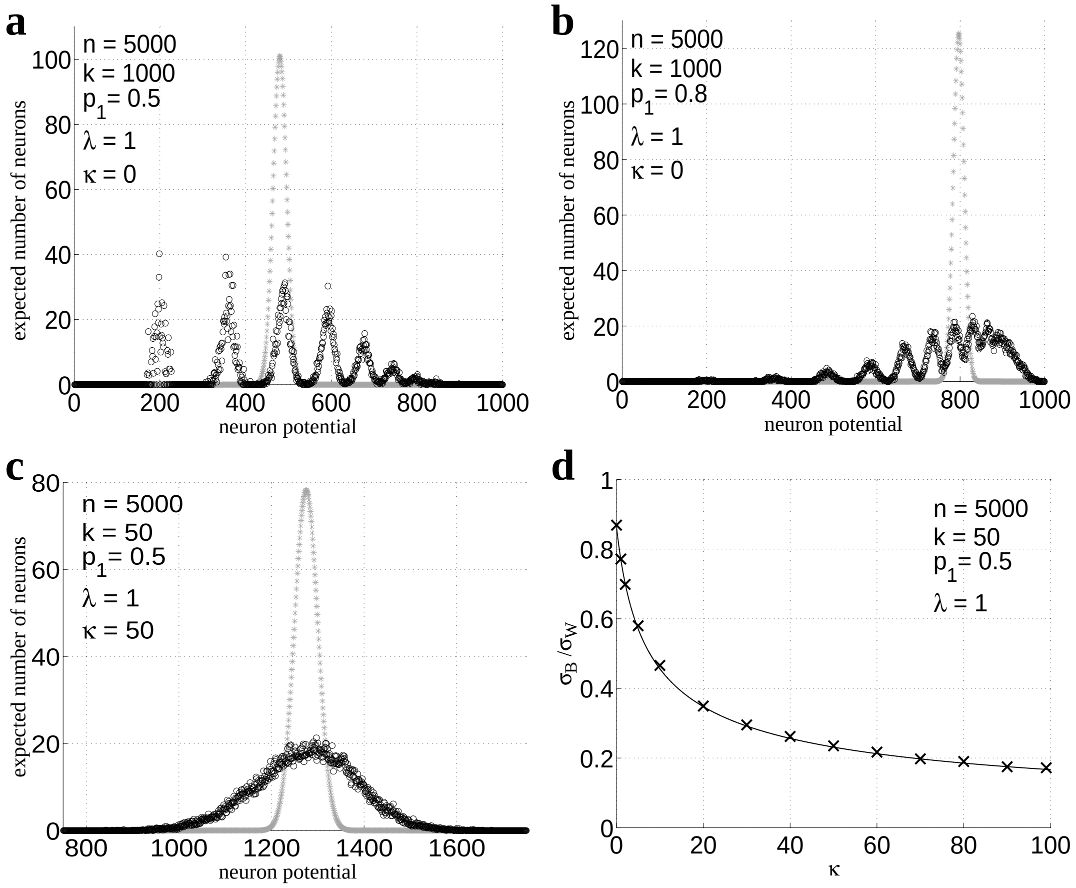


Figure 3.5: The binomial approximation of the true distribution of neuron potentials can be bad. **a**: True potential distribution (○) for non-addressed neurons for $n = 5000$, $k = 1000$, $p_1 = 0.5$ when addressing with $\lambda = 1$ (no missing ones in the address pattern) and $\kappa = 0$ (no false ones). The binomial approximation ($*$) is quite bad. Since $k \not\ll n$ only $M = 16 \not\gg 1$ patterns can be stored to obtain $p_1 = 0.5$. **b**: Same as (a) but $p_1 = 0.8$. A larger number of patterns $M = 39$ can be stored. The binomial approximation is still bad. **c**: Same as (a) but $k = 50$ and $\kappa = 50$. Although a large number of patterns $M = 6931$ can be stored (due to small k), the binomial approximation massively underestimates the variance of the true distribution of neuron potentials. Note that the activity of the address pattern ($\lambda k + \kappa k = 2550$) is on the order of n. **d**: Fraction of the standard deviations of the binomial approximation (σ_B) and the true Willshaw distribution (σ_W) of neuron potentials as a function of κ. The plot illustrates results from experiments ($\times$) for κ=0,1,2,5,10,20,...,90,99) and theory (solid line, cf. eq. 3.122). The binomial approximation is valid only for very low pattern activity $\lambda k + \kappa k \ll n$. Experimental estimations of the true potential distributions were computed by averaging potential histograms for 10 different address patterns.

We may be interested more deeply why the binomials are quite bad approximations of the true potential distributions, and more specifically, what causes the characteristic modulations (Fig.3.5a,b) and the massive underestimation of the true variance (Fig.3.5c,d) leading to a massive overestimation of fault tolerance at least against false ones.

An intuitive argument for the underestimation of the variance is the following. Assume that $k\kappa$ is on the order of n as in the examples illustrated in Fig.3.5c. Then it is very likely that the address pattern will not only address the single addressed assembly but also many others of the stored patterns. Even worse, the memory matrix may contain many more spurious stable states. For auto-association (and also similarly for hetero-association) each of the stored patterns is represented by a k-clique in the memory matrix, i.e., each of the pattern neurons is connected with the $k-1$ other neurons of the pattern. Note that the high-fidelity criterion guarantees only the occurrence of no or only few spurious $k+1$-cliques (i.e., the probability of a false one is small). On the other hand there exist many more smaller cliques, e.g., $k-1$-cliques most of which emerge from parts of several of the stored patterns. These spurious states would occur only with a negligible probability if the synapses were activated truly independently of each other as assumed by the binomial approximation.

In the following we will analyze *quantitatively* the phenomena illustrated in Fig.3.5. The true distribution of membrane potentials as determined by Buckingham and Willshaw [29] and subsequently refined by Sommer and Palm [171] covers the observed phenomena. When the address pattern contains exactly $z := (\lambda+\kappa)k$ ones then with the binomial probability $p_B(k;n,p) := \binom{n}{k}p^k(1-p)^{n-k}$ the probability that a non-addressed neuron has dendritic potential $X_R = x$ is

$$\begin{aligned}
\mathrm{pr}[X_R = x] &= \sum_{i=0}^{M} p_B(i;M,k/n)\cdot p_B(x;z,1-(1-k/n)^i) \qquad (3.94)\\
&= \binom{z}{x}\sum_{i=0}^{M}\binom{M}{i}(k/n)^i(1-k/n)^{M-i}[1-(1-k/n)^i]^x[(1-k/n)^i]^{z-x} \qquad (3.95)\\
&= \binom{z}{x}\sum_{s=0}^{x}(-1)^s\binom{x}{s}[1-\frac{k}{n}(1-(1-k/n)^{s+z-x})]^M - \delta_{x,0}(1-q)^M, \qquad (3.96)
\end{aligned}$$

where $\delta_{x,0}$ is the Kronecker symbol (1 for $x=0$, and 0 else). The transformation of eq. 3.95 to eq. 3.96 is detailed in [171]. Note that the refined formula of Sommer and Palm (eq. 3.96) requires the addition of only $x \le z \ll n$ terms where the original formula of Buckingham and Willshaw requires the addition of M terms (with typically $M \gg n$) which provides a considerable relief for computation.

The sum in the formula of Buckingham and Willshaw (eqs. 3.94 and 3.95) is over the so-called *unit usage* i. That means, for a given neuron, i is the number of assemblies (stored patterns) the neuron belongs to. For random patterns, the unit usage i is a binomially distributed quantity. Similarly, given the unit usage i, the number of active synapses x (out of z) is also binomially distributed where $1-(1-k/n)^i$ is the probability for any of the z relevant synapses to be active. It should be noted that the formula of Buckingham and Willshaw is only exact for $\mathrm{pr}[X_R = x]$ if the ones in the stored random pattern vectors were generated independently with probability k/n for each component, whereas we used random patterns each containing *exactly* k ones (see section 3.1). However, this should cause only minor deviations.

3.6.2 Oscillatory modulations in the membrane potential distribution

The formula of Buckingham and Willshaw (eq. 3.94) helps us to understand why sometimes the potential histograms show characteristic oscillatory modulations (see Fig.3.5a,b). This effect origins obviously from the additive superposition of the binomial probability density functions $p_B(x;z,1-(1-k/n)^i)$ in eq. 3.94. The expectation and variance of the i-th binomial can be written as

$$\begin{aligned}
E(B_{z,1-(1-k/n)^i}) &= z(1-(1-k/n)^i) = z\frac{k}{n}\sum_{j=0}^{i-1}(1-\frac{k}{n})^j, \qquad (3.97)\\
\mathrm{Var}(B_{z,1-(1-k/n)^i}) &= (1-k/n)^i\cdot E(B_{z,1-(1-k/n)^i}). \qquad (3.98)
\end{aligned}$$

Additionally the i-th binomial in eq.3.94 is weighed with a factor $p_B(i;M,k/n)$. For i around the expectation $i_0 := Mk/n$ the weights differ significantly from zero. Thus to avoid oscillatory modulations as in Fig.3.5a,b we have to require that for i around i_0 the standard deviations of the binomials $B_{z,1-(1-k/n)^i)}$ are much greater than the distance between two neighboring expectations, i.e., we require

$$\sqrt{z(1-(1-\frac{k}{n})^{i_0})(1-\frac{k}{n})^{i_0}} \gg z\frac{k}{n}(1-\frac{k}{n})^{i_0} \tag{3.99}$$

$$\Leftrightarrow \quad z\cdot(\frac{k}{n})^2\cdot(1-\frac{k}{n})^{Mk/n} \ll 1. \tag{3.100}$$

For the parameters of Fig.3.5a ($n = 5000$, $k = 1000$, $M = 16$, $z = 1000$) we have $z\cdot(k/n)^2\cdot(1-k/n)^{Mk/n} \approx 19.6 \not\ll 1$. Therefore we can expect oscillatory modulations for the potential histograms with peaks at $x = 0, 200, 360, 488, 590, 672, 738, 790, ...$ (evaluate eq.3.97 for $i = 0, 1, 2, ...$), cf. Fig.3.5a. For the parameters of Fig.3.5c ($n = 5000$, $k = 50$, $M = 6931$, $z = 2550$) we have $z\cdot(k/n)^2\cdot(1-k/n)^{Mk/n} \approx 0.69 < 1$, and therefore the modulations are almost absent. For cortical parameters ($n = 100000$, $p_1 = 0.1$, $k = 100$; see section 2.3; cf. [27, 136, 203, 169]) we have $z\cdot(k/n)^2\cdot(1-k/n)^{Mk/n} \approx z\cdot 9\cdot 10^{-7} \ll 1$ for $z \leq n$. Thus the described effect of oscillatory modulations plays probably no important role for physiological parameters, at least if the estimated assembly size [136, 203, 169] is correct. However, in sections 3.7.2 and 4.4.3 we will discuss possible functional implications of a related effect that will occur for any reasonable set of parameters.

3.6.3 Underestimation of the variance of membrane potentials

We still have to explain why the binomial approximations in Fig. 3.5c,d massively underestimate the variance of (non-addressed) neuron potentials which can lead to an overestimation of the fault tolerance of Willshaw associative memory. For this we try to extract the variance $\mathrm{Var}(X_R) = E({X_R}^2) - E(X_R)^2$ from the formula of Buckingham and Willshaw (eq. 3.94). First note that the binomial approximation already gives us the correct expectation (cf. eq. 3.4)

$$E(X_R) = z\cdot p_1 = z\cdot(1-(1-\frac{k^2}{n^2})^M) \tag{3.101}$$

Nevertheless it turns out to be useful to compute $E(X_R)$ for a second time from eq.3.94,

$$E(X_R) = \sum_{x=0}^{z} x \sum_{i=0}^{M} p_B(i; M, k/n)\cdot p_B(x; z, 1-(1-k/n)^i) \tag{3.102}$$

$$= \sum_{i=0}^{M} p_B(i; M, k/n)\cdot \sum_{x=0}^{z} x p_B(x; z, 1-(1-k/n)^i) \tag{3.103}$$

$$= \sum_{i=0}^{M} p_B(i; M, k/n)\cdot z(1-(1-k/n)^i). \tag{3.104}$$

Note that the second sum in eq.3.103 equals the expectation of a binomial $B_{z,1-(1-k/n)^i}$. Comparing eq.3.101 and eq.3.104 we obtain for $p := k/n$

$$\sum_{i=0}^{M} p_B(i; M, p)\cdot(1-p)^i = (1-p^2)^M \approx e^{-Mp^2} \tag{3.105}$$

which will become helpful immediately when computing $E(X_R{}^2)$. From eq.3.94 we obtain

$$\begin{aligned}
&(3.106) & E(X_R{}^2) &= \sum_{x=0}^{z} x^2 \sum_{i=0}^{M} p_B(i;M,k/n)\cdot p_B(x;z,1-(1-k/n)^i) \\
&(3.107) & &= \sum_{i=0}^{M} p_B(i;M,k/n)\cdot \sum_{x=0}^{z} x^2 p_B(x;z,1-(1-k/n)^i) \\
&(3.108) & &= \sum_{i=0}^{M} p_B(i;M,k/n)\cdot [z(1-(1-k/n)^i)(1-k/n)^i + z^2(1-(1-k/n)^i)^2] \\
&(3.109) & &= z\sum_{i=0}^{M} p_B(i;M,k/n)\cdot [z-(2z-1)(1-k/n)^i + (z-1)(1-k/n)^{2i}] \\
&(3.110) & &= z^2 - z(2z-1)(1-(k/n)^2)^M + z(z-1)\sum_{i=0}^{M} p_B(i;M,k/n)\cdot(1-k/n)^{2i}.
\end{aligned}$$

Note that the second sum in eq. 3.107 equals the sum of the squared expectation and the variance of a binomial $B_{z,1-(1-k/n)^i}$. For eq.3.110 we used eq.3.105. Simplifying the remaining sum in eq.3.110 turns out to be more difficult. For $M(k/n)(1-k/n) \gg 1$ we can approximate $p_B(i;M,k/n)$ by a Gaussian with expectation μ and variance σ^2, and the sum by a integral. Using

$$(3.111) \qquad \int_{-\infty}^{\infty} e^{-at^2+bt}dt = e^{\frac{b^2}{4a}} \cdot \sqrt{\frac{\pi}{a}}$$

(which can be derived, for example, from [28] no. 3), we can write for arbitrary $0 < p \ll 1$, $\sigma^2 := Mp(1-p) \gg 1$, $\mu := Mp$, and $q = (1-p)^r$,

$$\begin{aligned}
&(3.112) & \sum_{i=0}^{M} p_B(i;M,p)\cdot q^i &\approx \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty} e^{-\frac{(t-\mu)^2}{2\sigma^2}} e^{t\ln q}dt \\
&(3.113) & &= \frac{e^{-\frac{\mu^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}t^2+(\ln q+\frac{\mu}{\sigma^2})t}dt \\
&(3.114) & &= e^{-\frac{\mu^2}{2\sigma^2}} \cdot e^{(\ln q+\frac{\mu}{\sigma^2})^2\frac{\sigma^2}{2}} \\
&(3.115) & &= e^{(\ln q)^2\frac{\sigma^2}{2}+\mu\ln q} \\
&(3.116) & &\approx e^{-rMp^2(1+\frac{p}{2}(1-r(1-p)(1+p+\frac{p^2}{4})))} \\
&(3.117) & &\approx e^{-rMp^2(1-p(r-1)/2)} \\
&(3.118) & &\approx e^{-rMp^2}
\end{aligned}$$

where for the second approximation we used $\ln q \approx -rp(1+p/2)$. The last approximation is valid for $r(r-1)Mp^3/2 \ll 1$. Note that for $p = k/n$ and M chosen according to the high-fidelity criterion (see eq.3.10) we have asymptotically for $n \to \infty$ indeed $Mp^3 = -\ln(1-p_1)\cdot k/n \to 0$ for sublinear k (even for $p_1 \to 1$, cf. eq.3.22). Thus we obtain for $p = k/n$ and $r = 2$ from eqs.3.110 and 3.117

$$(3.119) \qquad E(X_R{}^2) \approx z^2 - z(2z-1)(1-(k/n)^2)^M + z(z-1)e^{-2M(k/n)^2(1-k/(2n))}$$

For $k/n \ll 1$, large M, using eq. 3.4 and equivalently $\exp(-M(k/n)^2) = 1-p_1$ (cf. eq.3.105), and the linear approximation $a^{1-b} \approx a(1-b\ln a)$ (with $a := (1-p_1)^2$, $0 < b := k/(2n) \ll 1$), we can write for the variance ${\sigma_W}^2$ of a Willshaw distributed random variable $X_R \sim W_{z,n,k,M}$,

$$
\begin{aligned}
&(3.120) \quad & {\sigma_W}^2 &:= \mathrm{Var}({X_R}^2) = E({X_R}^2) - E(X_R)^2 \\
&(3.121) \quad & &\approx z^2 - z(2z-1)(1-p_1) + z(z-1)(1-p_1)^2(1-\frac{k}{n}\ln(1-p_1)) - z^2{p_1}^2 \\
&(3.122) \quad & &= zp_1(1-p_1) - (z^2-z)\frac{k}{n}(1-p_1)^2\ln(1-p_1) \\
&(3.123) \quad & &\approx zp_1(1-p_1) \quad =: {\sigma_B}^2.
\end{aligned}
$$

The second approximation yields the variance ${\sigma_B}^2$ suggested by the classical binomial analysis, and is justified only if the relative error of the standard deviations, $\delta := 1 - \sigma_B/\sigma_W$, is small. For a given small δ we have to require

$$(3.124) \qquad z \;<\; 1 + \delta^2 \cdot \frac{p_1}{-(1-p_1)\ln(1-p_1)} \cdot \frac{n}{k}.$$

For our example in Fig.3.5c this would require $z = (\lambda+\kappa)k < 1 + 144\delta^2$. Since already $k = 50$ and $\lambda = 1$ the binomial approximation is justified only for $\kappa < 2.88\delta^2 - 1$, i.e., not at all for reasonable $\delta^2 \ll 1$. Indeed, the analysis in this section reveals that the binomial approximation underlying almost all classical results is justified only for very large n. For the classical optimum which occurs for $k = \mathrm{ld}\, n$ and $p_1 = 0.5$ we would require when addressing with correct address patterns (i.e., $z = k$),

$$(3.125) \qquad \mathrm{ld}\, n - \frac{\delta^2}{\ln 2} \cdot \frac{n}{\mathrm{ld}\, n} < 1.$$

Thus for $\delta = 0.1$ we would require $n > 11729$, and for $\delta = 0.01$ we would require already $n > 3069494$. The quality of the binomial approximation depends on k/n. Thus for finite n and ultra-sparse patterns with sub-logarithmic pattern activity $k < \mathrm{ld}\, n$ (see section 3.2.1) the binomial approximation is much better than for the classical or super-logarithmic pattern activity.

3.6.4 Consequences

Do the results of this section imply that the binomial analysis conducted in this chapter as well as in much of the classical literature [204, 129, 134] is rather irrelevant for practical finite n? In particular, the underestimation of the potential variance will lead to an underestimation of the error probabilities and therefore to an overestimation of fault tolerance especially against false one-entries in the address pattern. On the other side, in any case the results of the classical analysis concerning with number of storable patterns, error probabilities, and storage capacity deliver upper limits for the true values. Besides, we have up to date no better method of analyzing the binary Willshaw model in such a general fashion as provided by the binomial approximation. The structure of the theory is correct, and any future theory for the Willshaw model probably has the same structure. For example, for address patterns with $\kappa = 0$ (no false ones) the binomial analysis in this chapter is correct except for the assumption of the error probability p_{01} in eq. 3.6 (leading to a bad approximation of the maximal matrix load in eq. 3.9). The correct error probability p_{01} for a false one-entry when applying the Willshaw retrieval strategy can be derived from eq. 3.94 (see [31, 29, 30, 171]; cf. also appendix in [129]). For $x = \lambda k$ and $z := \lambda k$ we obtain

$$
\begin{aligned}
&(3.126) \quad & p_{01} &= \sum_{i=0}^{M} p_B(i; M, k/n)(1-(1-k/n)^i)^{\lambda k} \\
&(3.127) \quad & &= \sum_{s=0}^{\lambda k} (-1)^s \binom{\lambda k}{s} [1 - \frac{k}{n}(1-(1-k/n)^s)]^M,
\end{aligned}
$$

where the latter equation can be derived either by using eq.3.105 and expanding $(1-(1-k/n)^i)^{\lambda k}$ or directly by using eq.3.96. Unfortunately it turns out to be difficult to find an appropriate approximation for p_{01} such that we could repeat the analysis in this chapter.

3.6.5 An experiment to test the hypothesis of cell assemblies in the brain

On the one hand the results of this section are disappointing. The underestimation of the true neuron potential variance implies that the classical binomial analysis can be very bad for practical finite n, especially for large address pattern activity z (see eq. 3.122, cf. Fig. 3.5).

However, the difference between the true Willshaw variance (eq. 3.122) and the binomial variance (eq. 3.123) extracts the most important difference between independently generated synapses and synapses generated by Hebbian learning of neural assemblies. This difference should also be measurable in neurophysiological experiments, for example using brain slices.

The *basic idea* is to *stimulate* the neurons in a small piece of cortex as unspecifically as possible and then *measure* the mean μ_m and variance ${\sigma_m}^2$ of the postsynaptic potentials (or currents). For a given stimulation strength it should be possible to estimate the probability p_s that a neuron emits a spike. If we know the number n of neurons in the relevant piece of cortex, we can estimate the number $z := p_s \cdot n$ of relevant neurons activated by the stimulus. If we additionally know the probability p_1 and mean strength (here assumed to be 1) of a synapse between two neurons in our piece of cortex then we can even verify our estimate since $z = \mu_m/p_1$ (cf. eq. 3.101). Thus we can determine all variables in eq. 3.122 except for the assembly size k.

To decide if the piece of cortex contains assemblies or not, we have to repeat the experiment with different stimulation strengths corresponding to different values of z, and then plot the experimentally measured variance ${\sigma_m}^2$ versus z. If the synapses in the piece of cortex were generated independently of each other then we should obtain a *linear* relationship between ${\sigma_m}^2$ and z (eq. 3.123). In contrast, if the piece of cortex contained Hebbian cell assemblies then we should see rather a *quadratic* relationship (eq.3.122). (Alternatively we could simply check if σ_m/μ_m is correlated with the stimulation strength: if not this would support Hebbian learned cell assemblies, while a negative correlation corresponding to $\sigma_m/\mu_m \sim 1/\sqrt{z}$ would support independently generated synapses.)

If we really observe a quadratic relationship between ${\sigma_m}^2$ and z (or the absence of a correlation between σ_m/μ_m and the stimulation strength) then we can even estimate the assembly size k by resolving eq.3.122 for k,

$$k = \frac{{\sigma_m}^2 - zp_1(1-p_1)}{-(z^2-z)(1-p_1)^2\ln(1-p_1)} \cdot n. \tag{3.128}$$

Neuroanatomical findings about the extensions of dendrites and local axon collaterals suggest that an appropriate size for the relevant piece of cortex (where we can expect local cell assemblies) would be about 1mm^3 containing about $n = 100000$ neurons where the chance of a synaptic connection is about $p_1 = 0.1$ (see [27, 65]). Furthermore, theoretical considerations suggest an assembly size of perhaps a few hundred neurons [136, 203, 169]. If we assume $k = 10$ or $k = 100$ we can expect to measure variances ${\sigma_m}^2$ as illustrated in Figure 3.6. The test seems to be very robust especially for relatively large $k \geq 100$ corresponding to the theoretical estimations.

Note that this analysis provides only a raw sketch for the design of a real neurobiological experiment. There are a number of additional factors that have to be considered: (1) The unspecific stimulation will activate both excitatory and inhibitory neurons, but we are only interested in the excitatory postsynaptic potentials (or currents) assuming that cell assemblies are constituted mainly by excitatory cells (see [64, 130, 27]). (2) When determining the variance ${\sigma_m}^2$ we are only interested in the postsynaptic potentials (or currents) caused *monosynaptically* by the stimulation, and not of polysynaptic recurrent spikes. Thus the time delay between stimulation and measurement is crucial for the results. (3) The stimulation will yield temporally dispersed spikes such that the postsynaptic potentials will not perfectly overlay in time. This effect will be even stronger due to axonal transmission delays, in particular when testing *hetero-associative* cortico-cortical connections by stimulating a distant cortical location. (4) The Willshaw model assumes that all synapses have equal strength. In the real brain this is certainly not the case. Thus the present analysis is valid only if the true variance of the synaptic strength is small. However, experiments rather suggest that the standard deviations of synaptic strengths are on the order of the mean value (see dashed lines in Fig.3.6; see Fig.2 in [115], Fig. 52 in [27]; cf. [137]). (5) The

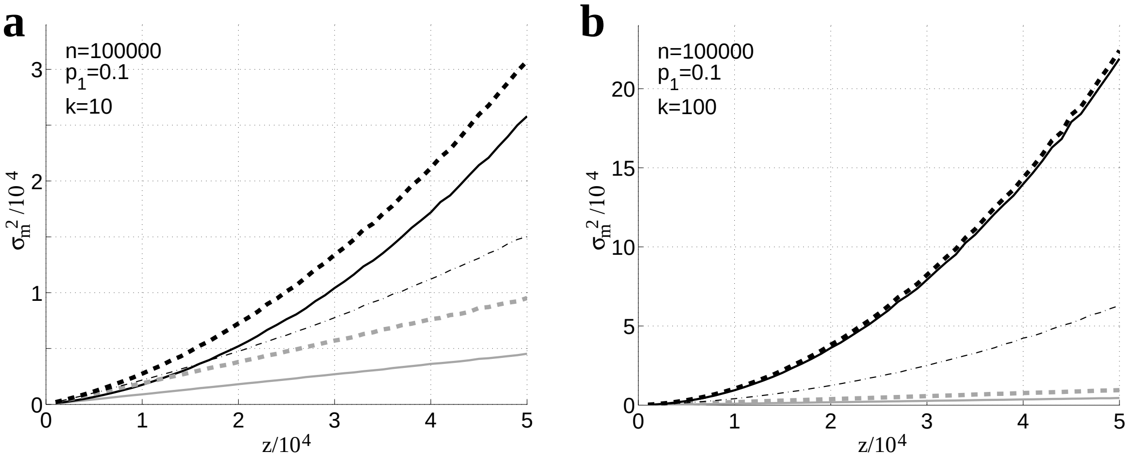


Figure 3.6: Expected variance ${\sigma_m}^2$ when measuring dendritic potentials in the proposed neurophysiological experiment (see text) vs. number z of 'address' neurons spiking by initial stimulation. For independently generated synapses ${\sigma_m}^2$ should be a linear function of z (gray), while ${\sigma_m}^2$ should be quadratic in z if the synapses were generated by Hebbian learning of cell assemblies (black). Unit of σ_m corresponds to the mean synaptic strength of active synapses. Strength of active synapses are assumed either to be the same for all active synapses (solid), or to be generated by Gaussians with mean and variance 1 (dashed or dash-dotted). For the dash-dotted line only half of the synapses were generated by Hebbian learning, and the probability of a synapse between two neurons of the same assembly was only 0.5. The difference to the corresponding curve expected for independently generated synapses (gray, dashed) is still significant. **a** : Expected neurophysiological results for $n = 100000$, $p_1 = 0.1$, $k = 10$. **b** : Same as (a) but $k = 100$.

described analysis assumes the linear summation of distributed postsynaptic currents. Actually, there is considerable controversy whether dendritic integration of synaptic inputs in real neurons is linear or non-linear [32, 110]. In the latter case it might be difficult to estimate the number of activated synapses from only one local measurement such as the somatic potential. (6) The classical Willshaw model assumes that synaptic connections are *possible* between *all* local neurons, and that within an assembly a neuron is connected with *all* other neurons of the same assembly. This is certainly not the case in the real brain. Even if the neurons are organized in assemblies it is likely that within an assembly the connection probability is increased compared to the mean connection probability (e.g., from 0.1 to 0.5), and that not all synapses correspond to Hebbian cell assemblies (see dash-dotted lines in Fig.3.6). (7) While the Willshaw model assumes that the probability of a synapse is p_1 for any pair of neurons, real cortical tissue exhibits an *anatomical bias*: Near neighbored neurons are more likely to be connected than more distant neurons, and only on average we have about $p_1 = 0.1$ within 1mm^3 of cortex [27, 65]. (8) A serious problem may be the unreliability of synaptic transmission [9]: A presynaptic spike can fail to evoke a postsynaptic response. Although in slice preparations the probability of a synaptic failure can be quite large (often > 0.5) it is not clear to what extent the failures are due to unphysiological recording conditions (e.g. low temperatures; see [63]). (9) In the analysis it is assumed that independently generated random patterns are stored in the memory matrix. But if there are local assemblies in the cortex it is probable that they will exhibit correlations.

In remains to be seen in discussions with neurobiologists if and how the problems (1) and (2) can be solved by experimental techniques. For problems (3), (4), (5), (6), (7), (8), and (9) it might be possible to refine the theoretical analysis. In any case, the effect of increased potential variance for Hebbian cell assemblies is strong enough especially for large k and z (cf. Fig. 3.6) such that we should be able to observe it despite the described problems.

3.7 Implications

3.7.1 Implications for technical applications

In section 3.2 we have demonstrated that optimal matrix compression by Huffman or Golomb coding [74, 57] (see also appendices A.3.1 and A.3.2) yields storage capacity $C^{\mathrm{cmpr}} = 1$ asymptotically. Surprisingly, this happens for almost all sublinear pattern activities $k(n)$. Indeed, it is somewhat ironic that choosing $k = c \log n$, which was widely analyzed so far and thought to be optimal, is the *only* possible choice of a (sublinearly) sparse code where $C^{\mathrm{cmpr}} \to 1$ is *not* possible.

Similarly as for the classical capacity C the convergence of C^{cmpr} is rather slow, and values near 1 are obtained only for n too large for real implementations (Fig. 3.2d). However, at least for ultra-sparse patterns (e.g. constant k with still $M \sim n^{2-1/k}$) values $C^{\mathrm{cmpr}} \approx \ln 2$, the classical upper bound, can be achieved already for practical n (e.g. $C^{\mathrm{cmpr}}(n) \approx 0.7$ for $k = 3$ and $n = 10^8$ in Fig.3.2d).

The fact that high storage capacities of the order of or above the classical optimum occur for *all* sublinear pattern activities $k(n)$ (and not only for $k = \mathrm{ld}\, n$, cf. Fig.3.2b) is a big relief for finding adequate codes. For certain applications $k = \mathrm{ld}\, n$ is simply too small for practical n to express complex hierarchical relations between the patterns [144], and one would prefer $k = \sqrt{n}$ where still a large number of (random) patterns $M \sim n \ln n$ can safely be retrieved.

On the other hand, using a large (i.e., super-logarithmic) pattern activity with $p_1 \to 1$ will cause at least two severe problems: First, for larger k the cross-talk between the patterns will also be larger if the patterns are *not* perfectly uncorrelated which is usually the case for practical applications. Second, $p_1 \to 1$ yields only storage capacity 1 if really the maximal possible number of patterns is stored. Since many applications will start with relatively few stored patterns (while many new patterns are learned only with passing time) this may lead to the paradox effect that the required physical memory is larger for fewer stored patterns than for the maximal number of stored patterns. Even worse, this effect occurs also if a smaller than maximal number of patterns is stored to improve fault-tolerance (section 3.3). Thus it remains to be shown if the $p_1 \to 1$ regime proves useful for technical applications. In any case, these problems do not occur for ultra-sparse patterns with with sub-logarithmic $k(n)$ and $p_1 \to 0$.

Furthermore, for sequential implementations optimal matrix compression applying Golomb coding not only improves the storage capacity but also the retrieval speed (section 3.4). For ultra-sparse patterns ($k < c \log n$) the Golomb-compressed Willshaw model is superior to the classical model with respect to the time t_{ret} required for a retrieval, the information flow f_I (information retrieved per time step), and the retrieval efficiency E (f_I normalized to the classical Willshaw model storing the same total amount of information). For moderately-sparse patterns ($k > c \log n$) the compressed Willshaw model is still superior with respect to f_I and E (but not for t_{ret}).

For parallel implementation (section 3.5) the classical Willshaw model remains superior to the compressed model with respect to t_{ret}, f_I, $f_{I,N}$, E, and E_N ($f_{I,N}$ and E_N are normalized to the number of processors) for both ultra-sparse and moderately-sparse patterns. Nevertheless, the Golomb-compressed Willshaw model can also quite efficiently be implemented in parallel if look-up tables are used to accelerate accessing individual synapses (see section 3.5.2).

In summary, for any sequential implementation Golomb-compression of the memory matrix is recommended as long as no excessive online-learning is required (learning of the association for a pattern pair needs k^2 steps for the classical model, but $\sim kn$ steps for the compressed model involving decompression and subsequent compression of k matrix rows). For parallel implementations, matrix compression can only be recommended if saving physical memory is more important than saving computing time. This may indeed be the case for some parallel architectures (like CNAPS, cf. [62]) where the individual processors have relatively limited working memory.

3.7.2 Implications for biological models

As detailed in section 2.3 associative memory can be interpreted as a model for the cerebral cortex where the stored patterns can be identified with *neural assemblies*, i.e., sub-groups of neurons which are highly interconnected by Hebbian learning of synapses [64, 26, 130, 133]. Therefore the results presented in this chapter might be also relevant for biological modeling and even neurophysiological experiments.

One important finding is that the classical binomial analysis of the Willshaw associative memory [204, 129, 134] can be very bad (see section 3.6) for finite parameters n (number of neurons), k (assembly size), and M (number of assemblies). This results from the binomial assumption of independently generated synapses leading to binomial B_{z,p_1} distributed dendritic potentials for input from z address neurons (typically $z = k$) and memory load p_1. Actually, for Hebbian learned cell assemblies the dendritic potentials follow rather a Willshaw $W_{z,n,k,M}$ distribution (see section 3.6.1), which converges towards the binomial distribution only for $n \to \infty$ if k and z are sub-linear in n (see section 3.6.3).

For finite parameters there can occur at least two important differences between the binomial and Willshaw distribution: (1) The probability density function of the Willshaw distribution can exhibit characteristic oscillatory modulations (see Fig. 3.5a,b). (2) The variance of the Willshaw distribution can be much larger than the variance of the corresponding binomial distribution (see Figs. 3.5c,d and 3.6).

On the one hand these results are disappointing when concerned with technical applications since the underestimation of the variance leads to an overestimation of storage capacity and fault tolerance (which means that the true performance of the Willshaw model is worse than expected from the classical theory). On the other hand the described statistical differences between the binomial and Willshaw distributions can in principle be used to *test the cell assembly hypothesis* in neurophysiological experiments (see section 3.6.5). The difference in variance is so large for plausible parameters (e.g., n=100000, $p_1 = 0.1$, $k = 100$, large z) that it should be measurable also in the real brain. While the variance should be linear in z for independently generated synapses the relationship should be rather quadratic if the cell assembly hypothesis is true (see Fig.3.6). For the latter case we could even estimate the assembly size k (see eq.3.128). However this will require a further refined theory that would take into account many additional factors influencing the measured variances not captured by the Willshaw model (e.g., distributed synaptic strengths, incomplete connections within assemblies, etc.; see section 3.6.5).

For three reasons it is probably difficult to observe the other difference between the binomial and Willshaw distribution in neurophysiological experiments, namely oscillatory modulations in the histograms of the dendritic potentials (Fig.3.5a,b). First, oscillatory modulations occur only for a large assembly size k and a small number M of stored patterns (see eq.3.100). For plausible cortical parameters ($n = 100000$, $k = 100$, $p_1 = 0.1$) we cannot expect oscillatory modulations. Second, even if the true assembly size in cortical tissue is much larger than the estimated number ($k = 100$, [136, 203, 169]) the effect may be an artifact of the idealizations of the Willshaw model since in the real brain the effect may be wiped out due to other factors influencing the potential distribution (such as distributed synaptic strengths, incomplete connections within assemblies, etc.; see section 3.6.5). Third, even if the oscillatory modulations occur in the brain, they will probably be very difficult to measure for technical reasons.

We can only speculate about functional implications of such oscillatory modulations in the membrane potential: If membrane potential distributions would really exhibit multiple peaks as illustrated in Fig.3.5a,b one effect could be that the neurons corresponding to each peak would fire in synchrony (if we assume that the 'potentials' in Fig.3.5 correspond to temporal change of the potentials in the real brain - see also the argumentation in chapter 4 about spiking neurons and pattern separation). But what would be synchronized then? Actually, this effect would not synchronize the activity of an assembly, but rather those neurons that belong to the same number of assemblies, i.e., the neurons which have the same *unit-usage* (see eq.3.94 and the remarks on page 47). The functional relevance of such a behavior is not clear.

Note that the described analysis of the oscillatory modulations in the membrane histograms

(Fig. 3.5a,b; see section 3.6.2) assumes that the z address neurons are selected at random. In this case we cannot expect oscillations if we assume parameters plausible for cortical tissue (see above). However, a related effect occurs trivially if the z address neurons constitute one of the assemblies as it is the case in the classical scenario of the analysis (see section 3.1.2). Then the distribution of the neuron potentials exhibits two peaks: The high-potential peak corresponds to the addressed neurons, while the low-potential peak corresponds to the non-addressed neurons. If the address pattern contains parts of two (N) assemblies then we will obtain four (2^N) peaks in the membrane potentials corresponding to neurons belonging to both assemblies, only to the first assembly, only to the second assembly, neither to the first nor to the second assembly. This related effect for superpositions of patterns or assemblies will occur obviously for any reasonable set of parameters, and here the assemblies themselves would be synchronized. In the next chapter we will deal with the separation of superimposed patterns in associative memory using spiking neurons.

In the last sections we have seen that it is possible to improve the asymptotic storage capacity of Willshaw associative memory from $\ln 2$ to 1 if the memory matrix is optimally compressed (see section 3.2). By applying Golomb coding this happens without severe computational costs (see appendix A.3.2). Performance is even improved, at least for sequential implementation (section 3.4). For implementing biological models on computers this obviously reduces the costs in terms of memory and computation time (see appendix C). But what are the relations to the biological reality of the brain?

As detailed in section 2.3.3 associative memory can be interpreted as a model for the cerebral cortex. The fact that the Willshaw model requires sparse coding to obtain high storage capacity was often interpreted as a striking coincidence with physiological findings of sparse spike activity in the brain: most neurons in the brain are only seldom activated and exhibit only low spontaneous spike rates of few (1-5) spikes per second. This might lead to the conclusion that the neural nets in the brain could be optimized with respect to the information theoretic storage capacity. On the other hand, it is known that the probability of a synapse between two local cortical neurons is approximately $p_1 = 0.1$. This obviously contradicts the maximal-capacity-hypothesis since the classical Willshaw model requires $p_1 = 0.5$ for optimal storage capacity (but see [24] for incompletely connected networks). Arguing that p_1 should be below the optimal value in order to stabilize the neuronal assemblies and allow pattern completion is also not really convincing.

In contrast, we have seen that the storage capacity for compressed memory matrices is minimal for non-sparse $p_1 = 0.5$, and the highest capacities are obtained for $p_1 \to 0$ (or $p_1 \to 1$). This should draw our attention to sparse synaptic connectivity as found in the brain. Of course, matrix compression has no obvious biological equivalent: The brain *cannot* 'compress' its synapses as we can do in our models of the brain. Nevertheless there are other arguments for sparse synaptic connectivity that are independent of whether the synaptic matrix is compressed or not. First, the maximum for the number of storable patterns M does not coincide with the maximum for the storage capacity. While maximal storage capacity is obtained asymptotically for $p_1 = 0.5$, maximal M occurs at only $p_1 \approx 0.16$ (see Fig.3.1c) which is much nearer to the physiological value $p_1 = 0.1$. Now the remaining difference might more convincingly be assigned to stabilizing patterns and the possibility of pattern completion or fault tolerance (see section 3.3).

Thus we should not rule out the possibility that, from an anatomical point of view, the brain is designed to maximize the number of stable activity states rather than the storage capacity. In the following chapter we will find another argument for sparse synaptic connectivity when investigating the use of spiking neurons for associative memory.

Chapter 4

Spiking associative memory

Associative memories can be used as models for the cerebral cortex. In the Willshaw model a binary synaptic memory matrix contains information about a certain set of stored patterns (see chapter 3). Since connections are required between all neurons of a pattern (or assembly), and local connections between local cortical neurons reach only few millimeters [27] we think that the classical Willshaw model can be used to model approximately 1mm^3 of cortical tissue (see sections 2.3.3 and 2.1.3). To model larger parts of the cortex it is more appropriate to connect many such elementary associative memories by hetero-associative (i.e. Hebbian learned) links (see chapter 5, or to use topographical models (see section 5.3.5).

In its basic form associative memory is a model of synaptic connectivity (i.e. the memory matrix) plus a "neuronal" algorithm to read out the stored information. However, in the real brain this involves *spiking* neurons, i.e. binary events in space and time. On the one hand the binary Willshaw model of associative memory can already be interpreted as a spiking model since the neurons are binary, i.e. the ones in the patterns correspond to single spikes (in space). On the other hand the temporal dimension collapsed completely since a retrieval is performed in a single atomic step (see section 3.1.1; but see [69]). Moreover, in other models of associative memories the patterns and synaptic interactions are rather gradual, and thus even the spatial aspect of a spike is missing (e.g., [103, 104]). Since all those models work apparently quite well in applications as well as in neural field models of brain parts one may question what the purpose or advantage of spikes is?

In this chapter we will study in more detail a problem where spiking neurons seem to be superior to gradual ones: The separation of superpositions of patterns. Together with the results from the last chapter (e.g., sparseness) this will lead us to some tentative answers why spikes occur in the brain.

4.1 Addressing with superpositions

The classical scenario how to read out information from an auto-associative network is to use just *one* of the previously stored patterns as address pattern to retrieve the complete pattern.

For the binary Willshaw model the situation might be as follows: The address pattern contains half of the ones of a previously stored pattern, i.e., $k/2$ of the k ones in a pattern. Additionally it may contain f further (spurious) ones generated at random. When addressing, i.e., applying the matrix-vector-multiplication as described in section 3.1.1, the potentials x of the neurons will be distributed approximately according to two binomial random variables. A neuron that does not belong to the addressed pattern is connected with one of the address neurons with probability p_1 (matrix load, cf. eq. 3.4), approximately independently for all address neurons (but see section 3.6; cf. [29, 171]). Thus the neuron potential is the outcome of a binomial random variable $B_{k/2+f,p_1}$ corresponding to $k/2 + f$ Bernoulli random experiments (true, false) where the experiment is "true" with probability p_1. Similarly, a potential of a neuron belonging to the addressed pattern will be distributed as a random variable $k/2+B_{f,p_1}$, because the neuron will be connected to all the $k/2$ neurons of the half pattern and the Bernoulli experiments will be performed only for the f spurious neurons. The situation is illustrated in Fig. 4.1a: One obtains a bimodal distribution of the neuron potentials, where one mode corresponds to the addressed neurons and the other mode corresponds to the rest of the cells. If the conditions for retrieval are met (i.e., not too many stored patterns with respect to the pattern activity k, see section 3.1.2 for details) the two binomial modes will be separated well, and we just have to choose an appropriate threshold between the two modes to retrieve the completed pattern.

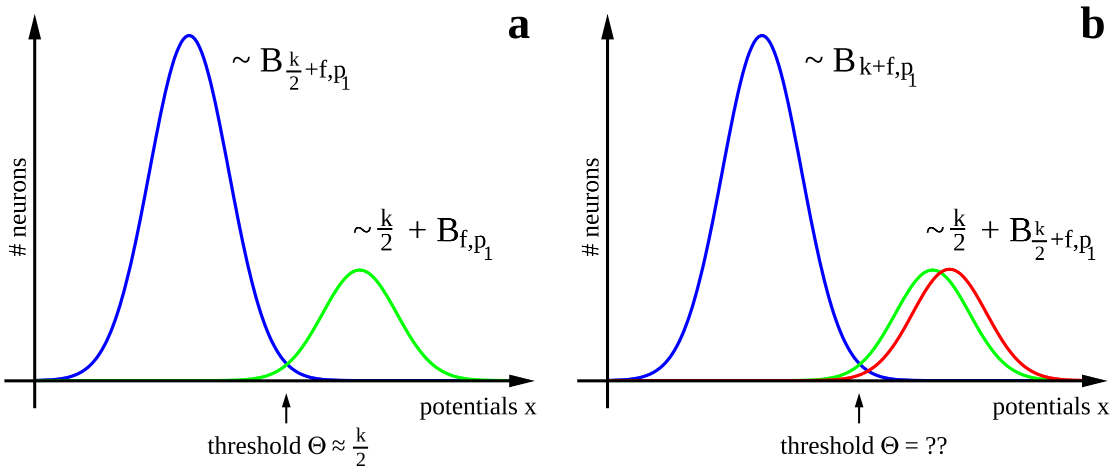


Figure 4.1: Potential distributions for one-step retrieval when addressing with a single pattern (a) or with a superposition of two patterns (b). **a**: When the address pattern consists of half ($k/2$ one-entries) of a previously stored pattern (the *addressed pattern*) plus noise (f “false” one-entries) then the neuron potential distribution is bimodal. The non-addressed neurons have potentials according to a binomial $B_{k/2+f,p_1}$, while the addressed neurons receive at least $k/2$ inputs plus input according to a binomial B_{f,p_1}. In this case the two modes are well separated, and the addressed pattern can be easily extracted by choosing a threshold Θ between the two modes. **b**: When addressing with the superposition of two halves of two previously stored patterns ($k/2$ one-entries for each half) plus f noisy one-entries, then an additional mode appears for the second addressed pattern (we have neglected a further very small mode corresponding to the neurons lying in the intersection of the two addressed patterns). The two modes corresponding to the two addressed patterns overlap closely. It is difficult to separate the two patterns by one-step retrieval because the retrieval result will always be a superposition of the two addressed patterns for any choice of the threshold Θ.

However, if we perform the retrieval using a superposition of, for example, two previously stored patterns the situation is as illustrated in Fig. 4.1b. Now we have a trimodal distribution where the two modes of the two addressed neuron groups may closely overlap (we neglected a further mode corresponding to the intersection of the two addressed patterns which is valid for random patterns): If we address with two half patterns plus f spurious ones we will obtain one mode distributed according to B_{k+f,p_1}, and two modes distributed with $k/2+B_{k/2+f,p_1}$. Thus, it is impossible to choose a threshold in order to extract only a single completed pattern. Instead we will always obtain a superposition of the two addressed patterns, irrespective of the threshold. It is argued that this is not desired: For example, the superposition of two patterns reduced to activity k may correspond to a third pattern that is not at all related to the current situation. In the next section a solution to this form of the superposition problem is described.

4.2 Pattern separation

Several different *technical* models of neural associative memory have been introduced to demonstrate the capability of neural networks to learn and retrieve patterns (e.g. [204, 103, 104, 129, 69, 70]), whereas the issue of pattern separation has not attracted as much attention so far. Often the classical models use continuous variables that are interpreted as firing rates, and they exhibit some problems when addressed with a superposition of several patterns. Normally, this will result in an activation of a superposition of stored patterns. This is true in particular for linear associative memory (e.g., [103, 104]) where per definition the result is the linear superposition of single input components.

For non-linear models retrievals can be performed iteratively [160, 171]: If the asymmetries in the address superposition are sufficient, the output pattern can converge towards one of the components, but only after several iterations. Using spiking neurons suggests a complementary solution for the problem of pattern separation. Here the basic idea is to interpret the synaptic inputs delivered by an addressed pattern *not* as the *potentials* x (see Fig.4.1), but rather as the temporal change of the potentials dx/dt as illustrated in Figure 4.2. Then the different (binomial) modes start moving towards a fixed threshold Θ. At some time the first neuron will reach the threshold and therefore emit a spike. This spike is immediately fed back which will further excite the neurons of the corresponding pattern and suppress somehow the neurons of other rivaling patterns. Thus in the ideal case one complete pattern will pop out. Similarly as for classical one-step retrieval the result can be further improved by iterating the spiking retrieval. For extraction of the other components of the superposition further spiking retrievals have to be performed, where the neurons activated in previous retrievals can be suppressed (e.g., by eliminating the corresponding address inputs) in order to prevent multiple activations of the same pattern.

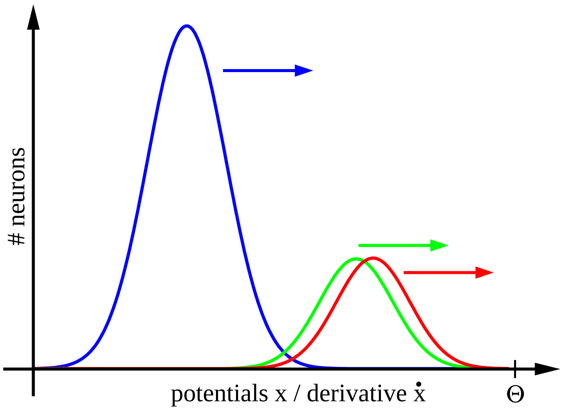


Figure 4.2: A solution to the superposition problem illustrated in Fig. 4.1. Instead of the membrane potential x the address input is interpreted as the temporal change dx/dt of the membrane potential. Then the three binomial modes start moving towards a fixed threshold Θ. At some time the first neuron reaches the threshold and a spike is emitted. By immediate selective feedback one of the pattern can pop out, while the others are suppressed.

The described spike mechanism reminds of the asynchronous updating rule of the Hopfield model [69], although in the original Hopfield model for the next spike an arbitrary superthreshold neuron is chosen *at random* whereas it would be more plausible to simply choose the most excited neuron (on the other hand this prevents the expensive computation of the maximum of a set containing n values; cf. section 4.5). The postulated use of spiking neurons for scene segmentation or pattern separation has already been described by several authors simulating rather *biological* (usually time-continuous) versions of the Hopfield model (e.g., [150]) or the Willshaw model (e.g., [199, 197, 201]).

Wennekers and Palm demonstrated the pattern separation capability of a spiking variant of the Willshaw model ([199], see also [172]). They used two populations of neurons, one excitatory spiking population that was auto-associatively connected according to the clipped Hebbian learning rule [64, 204, 129], and a single gradual inhibitory interneuron reciprocally coupled with the excitatory population. Connections from an input population coupled to the excitatory population and initiated a retrieval process. The function of the inhibitory population was to control excitation so that only one of the addressed populations got activated. However, this model was faced with two problems. First, the strength of necessary inhibitory connections depends strongly on the size of the addressing pattern, i.e. different synaptic strengths were needed when addressing by patterns with either missing or additional active elements, and similarly when addressing with a superposition of several incomplete patterns. Second, introducing realistic synaptic and axonal delays, the pattern separation capability vanishes, since the indirect activation of inhibition is too slow.

In the following section we will develop a *technical* extension of classical one-step retrieval for the Willshaw model, the so-called *spike-counter model* [86, 89]. This model is based on the principles of spiking retrievals as discussed in this section and implements an instantaneous version of the Willshaw retrieval strategy of the classical model (see section 3.1.1). It turns out that the spike counter model can be implemented almost as efficiently as one-step retrieval (see section 4.5). Further, in section 4.6 we can derive a biological version of the spike counter model that is much more robust to synaptic delays and differences in address pattern size than previously suggested models [199].

4.3 Spike counter model

When addressing an auto-associative memory in the Willshaw model with a certain pattern, the result is that every neuron i can be assigned a counter value c_i corresponding to the number of connected address inputs. The result of the retrieval is the set of neurons with counter values exceeding a global threshold Θ. For threshold setting, two strategies can be used: (1) *Willshaw retrieval strategy* (cf. section 3.1.1): If we can assume that the address pattern is an incomplete version of a stored pattern, then the threshold can be chosen equal to the number of active units in the address pattern. This way we obtain all neurons of the stored pattern, plus perhaps some false ones. (2) When nothing is known about the address pattern, one has to adjust the threshold step by step until the number of active units in the resulting pattern has a desired value (cf. [202]). Note that the second strategy constrains the pattern sizes to be about the same for all patterns.

Interpreting this technical model as a biological one, the idea is to interpret the counter values c_i as membrane potentials x_i, and Θ as the firing threshold of the neurons. Thus only neurons above the threshold get activated (see Fig.4.1) with a certain rate (which saturates immediately in the case of binary patterns).

An alternative would be to interpret the counter values c_i as temporal changes of membrane potentials dx_i/dt. Then we obtain temporal dynamics (see Fig. 4.2): Given the instantaneous membrane potentials we can compute for each neuron, if and when it will exceed its threshold. The idea is to integrate the membrane potentials only until the next spike occurs, and then to increment the count values c_i for each neuron connected to the spiking one. This gives us new values of dx_i/dt and we continue to integrate the potentials until a sufficient number of spikes has occurred. Since already the first spike of a retrieval is fed back immediately, fast pattern separation is accomplished: the first spikes determine which of the simultaneously addressed assemblies becomes activated.

This is essentially what happened in the model of Wennekers and Palm [199] (cf. section 4.2). The function of the inhibitory neuron was to increase gradually the threshold Θ to let only the strongest activated neurons fire. But this happened in a quite uncontrolled way.

To get an optimal threshold control, we should consider the relation between the spike counter and the temporal change of membrane potential more closely. To this end we split up our counter values c_i in several categories and propose that the temporal derivative of the potentials $dx_i(t)/dt$

should be a function f

$$\frac{d}{dt}x_i = f(c_i^H, c_i^A, c^\Sigma) \tag{4.1}$$

of the number of spikes ${c^H}_i$ received from the address area (by hetero-association or essentially by auto-association if one-to-one connections from the address area are used), the number of spikes ${c^A}_i$ received *auto-associatively* from intra-areal neurons, and the *total* number of spikes c^Σ generated by *all* intra-areal neurons until time t (see Figure 4.3).

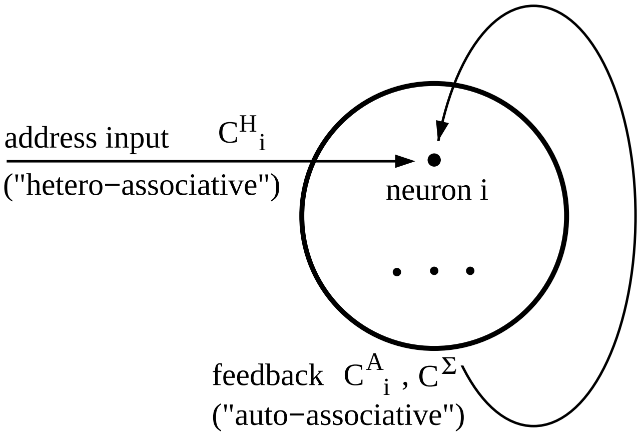


Figure 4.3: The spike counter model: Each neuron i has three counter states: (1) ${c^H}_i$ counts the input spikes obtained from the address area (e.g., hetero-associatively; see text); (2) ${c^A}_i$ counts the number of spikes received auto-associatively by feedback from intra-areal neurons; (3) c^Σ counts the total number of spikes generated by all intra-areal neurons (and therefore c^Σ is equal for all local neurons).

To implement something analogous to the Willshaw threshold strategy (1) described above, f should exhibit the following properties:

- If $c^A \approx c^\Sigma$, then $f(c^H, c^A, c^\Sigma)$ should be positive.
- If $c^A \ll c^\Sigma$, then $f(c^H, c^A, c^\Sigma)$ should be negative.

Essentially, the role of c^Σ is that of an instantaneous threshold that allows increasing membrane potentials only for neurons connected with (nearly) all the neurons that already have emitted a spike, and therefore we can expect the exclusive activation of one stored pattern (remember that patterns in the Willshaw model are completely connected subsets of the neurons). Figure 4.4 illustrates the retrieval process and the meaning of the spike counters.

Note that our model exhibits a very friendly behavior, if one chooses f as proposed above:

- The problem in the Willshaw model of adjusting the threshold Θ dependent on the size of the address pattern is eliminated and it is even possible to store assemblies of different sizes. This is because as long as all neurons that exhibited a spike belong to one assembly, for all other neurons of this assembly we have $c^A = c^\Sigma$, and therefore exclusively their potentials continue to increase towards the threshold.
- Miscarried retrievals (e.g. if neurons of different assemblies have already fired) exhibit a tendency to break down at an early time, because the condition $c^A \approx c^\Sigma$ is no longer fulfilled for any neuron.

Thus we obtain autonomous threshold control and autonomous detection of failed retrievals.

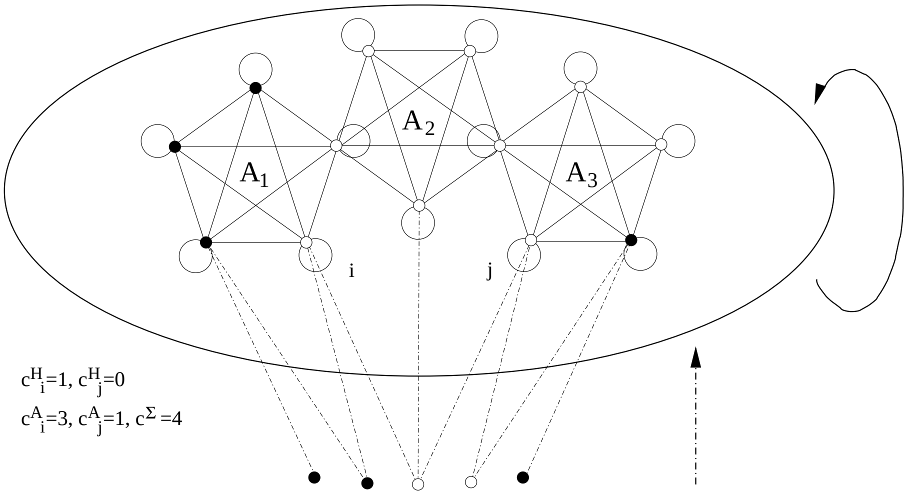


Figure 4.4: Illustration of a retrieval in the spike counter model according to the Willshaw associative memory. The associative memory (ellipse) contains three assemblies (completely connected subgraphs A_1, A_2 and A_3) addressed by extra-areal input (five units below ellipse). Neurons that already have spiked are represented by black circles, other neurons by unfilled circles. Neurons i and j get input from ${c^H}_i = 1$ and ${c^H}_j = 0$ hetero-associatively connected active addressing units. At the moment, $c^\Sigma = 4$ neurons of the associative memory have already spiked, three neurons of assembly A_1 and one neuron of A_2. Therefore neurons i and j get auto-associative input ${c^A}_i = 3$ and ${c^A}_j = 1$ respectively. Since for neuron i (as well as for the other neurons of A_1) we have ${c^A}_i \approx c^\Sigma$ the potential of neuron i will continue to increase towards the firing threshold (for an appropriate input integration function f). In contrast, the potential of neuron j will decrease because ${c^A}_j \ll c^\Sigma$. The result will be a complete activation of assembly A_1 and a suppression of the remaining neurons.

A simple linear example for an appropriate input integration function is the following:

$$f_1(c^H, c^A, c^\Sigma) = ac^H + b\left(c^A - \alpha c^\Sigma\right), \quad a, b > 0, \quad 0 < \alpha \leq 1 \tag{4.2}$$

Parameters a and b determine strengths of inter-areal and intra-areal inputs respectively. Parameter α determines the so called *separation strength*. E.g. a separation strength of $\alpha = 0.9$ means that the auto-associative feedback is only excitatory for neurons connected to more than 90 percent of the neurons that have fired in the addressed area. For all other neurons, the auto-associative feedback is inhibitory.

For $\alpha \approx 1$, the first required property for f is obviously fulfilled. The second can be achieved, if one weighs the influence of the auto-associative feedback much stronger than the influence of the hetero-associatively addressing area. This is also justified for biological models (cf. sections 2.1.3 and 2.3.3). According to [27], half of the synapses on a cortical pyramidal cell are made by axons belonging to local (auto-associatively coupled) neurons of the same area, while the other half results from possibly remote areas. Experiments revealed that a local cortical neuron group is connected to five to eight other local groups. This suggests that the local auto-associative feedback may be about five to eight times stronger than input from one extra-areal neuron group.

An efficient implementation of a technically optimized variant of this model is described in section 4.5. In the following we will analyze the spike counter model with respect to fault tolerance and compare it to classical one-step retrieval in the classical Willshaw model.

4.4 Analysis of the spike counter model

The analysis of the classical Willshaw model applies to a certain degree also to the spike counter model of Willshaw associative memory. In particular the analysis of using single address patterns with missing ones (see section 3.3.1) delivers a lower bound for the fault tolerance of the spike counter model. In section 3.3.2 we have postponed the analysis of address patterns with false ones for the classical Willshaw model. Now this is caught up for the spike counter model. In the following we will investigate conditions for the number of false ones in the address pattern, κk such that the neurons with the highest membrane potentials can still be expected to be correct. First using the binomial approximation similar as in the classical analysis we can compute explicitely the maximal possible κ (see eq. 4.34). Then we will conduct the refined analysis (see section 3.6) where we can still obtain an implicit formula for for κ (see eq. 4.64).

4.4.1 Single noisy address patterns

Consider addressing with one of the original address patterns (size n) containing only λk out of k ones of the original address pattern ($0 < \lambda \leq 1$), but also κk false ones ($\kappa \geq 0$). If we further assume a matrix load p_1 (which should equal $(\epsilon k/n)^{1/(\lambda k)}$ if we require high-fidelity, cf. section 3.3.1) the potentials of addressed neurons and the remaining non-addressed neurons are distributed according to random variables X_1 and X_R, respectively,

$$X_1 \sim \lambda k + B_{\kappa k, p_1} \approx N_{\mu_1,\sigma_1} \tag{4.3}$$

$$X_R \sim B_{(\lambda+\kappa)k, p_1} \approx N_{\mu_R,\sigma_R} \tag{4.4}$$

where the $B_{n,p}$ are binomial random variables with expectation np and variance $np(1-p)$. From the DeMoivre-Laplace theorem we know that for $np(1-p) \gg 1$ (cf. section 4.4.4) the binomials can be approximated by Gaussians $N_{\mu,\sigma}$ with corresponding expectation μ and variance σ^2. Thus

$$\mu_1 = (\lambda + \kappa p_1)k \tag{4.5}$$

$$\mu_R = (\lambda + \kappa)p_1 k \tag{4.6}$$

$$\sigma_1 = \sqrt{\kappa p_1 q_1 k} \tag{4.7}$$

$$\sigma_R = \sqrt{(\lambda + \kappa)p_1 q_1 k} \tag{4.8}$$

with $q_1 := 1 - p_1$. We are interested in the probability that the potential of a certain neuron takes at least a value of x. To analyze this we use the Gaussian error function

$$G(x) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt = \mathrm{pr}[N_{0,1} \leq x] \tag{4.9}$$

For general Gaussians with parameters μ and σ we have

$$\mathrm{pr}[N_{\mu,\sigma} \leq x] = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x} e^{-(t-\mu)^2/(2\sigma^2)} dt \tag{4.10}$$

$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{(x-\mu)/\sigma} e^{-s^2/2} ds \tag{4.11}$$

$$= G\left(\frac{x-\mu}{\sigma}\right), \tag{4.12}$$

where for the integrals we applied the substitution $s = (t-\mu)/\sigma$. For a given number $\#_1$ we can determine a value x_1 such that on average $\#_1$ of the k addressed neurons have a potential larger than x_1 by demanding

$$k \cdot \mathrm{pr}[X_1 \geq x_1] = \#_1 \quad (!) \quad \Leftrightarrow \quad k\left(1 - G\left(\frac{x_1 - \mu_1}{\sigma_1}\right)\right) = \#_1 \tag{4.13}$$

$$\Leftrightarrow \quad \frac{x_1 - \mu_1}{\sigma_1} = G^{-1}(1 - \#_1/k) =: g_1 \tag{4.14}$$

$$\Leftrightarrow \quad x_1 = \sigma_1 g_1 + \mu_1. \tag{4.15}$$

Analogously we can determine a value x_R such that on average $\#_R$ of the $n-k$ non-addressed neurons have a potential larger than x_R,

$$(n-k)\cdot \mathrm{pr}[X_R \geq x_R] = \#_R \quad (!) \quad \Leftrightarrow \quad x_R = \sigma_R g_R + \mu_R, \tag{4.16}$$

where $g_R := G^{-1}(1-\#_R/(n-k))$. We assume reasonable values $n > k > 2\#_1 \leq 2\#_R$ which implies $g_R > g_1 > 0$. Note also that g_R and g_1 increase only very slowly with n and k (see eq. 4.28). Fig. 4.5 illustrates the situation for cortical parameters $n = 100000, k = 100, p_1 = 0.1$ (see section 2.3.3) and $\lambda = 1, \kappa = 10, \#_1 = 30, \#_R = 10$.

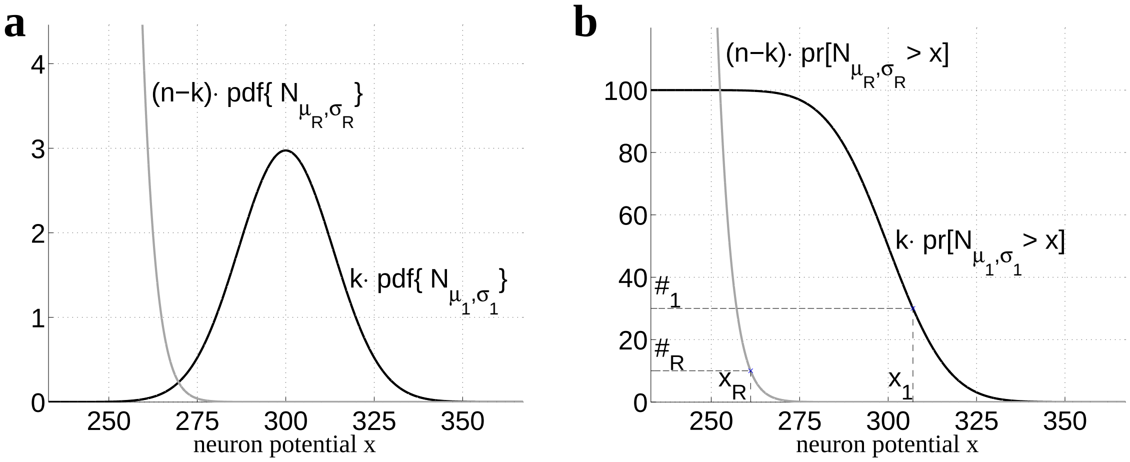


Figure 4.5: Neuron potential distributions when addressing with a single pattern plus noise for cortical parameters $n = 100000, k = 100, p_1 = 0.1$ and $\lambda = 1, \kappa = 20, \#_1 = 30, \#_R = 10$ (see text). **a**: Gaussian probability density functions (pdf) for addressed neurons (black) and non-addressed neurons (gray), scaled to obtain the expected neuron potential histograms (bin size 1). **b**: Expected number of neurons exceeding potential x for addressed neurons (black) and non-addressed neurons (gray). We can expect $\#_1$ ($\#_R$) of the addressed (non-addressed) neurons to have a potential of x_1 (x_R) or larger.

For a small # (e.g., # = 1) we can argue that $x_1 > x_R$ implies that on average the neurons firing first will be *addressed* neurons (see Fig.4.5). Therefore we can expect a good retrieval result for the spike counter model (see section 4.3). The condition $x_1 > x_R$ is equivalent to

$$x_1 > x_R \quad (!) \quad \Leftrightarrow \quad \sigma_1 g_1 + \mu_1 \; > \; \sigma_R g_R + \mu_R \tag{4.17}$$

$$\Leftrightarrow \quad g_1\sqrt{\kappa p_1 q_1 k} + (\lambda + \kappa p_1)k \; > \; g_R\sqrt{(\lambda+\kappa)p_1 q_1 k} + (\lambda+\kappa)p_1 k \tag{4.18}$$

$$\Leftrightarrow \quad \sqrt{p_1 q_1 k}\left(g_R\sqrt{\lambda+\kappa} - g_1\sqrt{\kappa}\right) \; < \; \lambda k(1-p_1) \tag{4.19}$$

$$\Leftrightarrow \quad g_R\sqrt{\lambda+\kappa} \; < \; \lambda\sqrt{kq_1/p_1} + g_1\sqrt{\kappa} \tag{4.20}$$

$$\Leftrightarrow \quad g_R^2(\lambda+\kappa) \; < \; \lambda^2 kq_1/p_1 + g_1^2\kappa - 2\lambda g_1\sqrt{\kappa k q_1/p_1} \tag{4.21}$$

$$\Leftrightarrow \quad (g_R^2 - g_1^2)\cdot\sqrt{\kappa}^2 - 2\lambda e_1\sqrt{kq_1/p_1}\cdot\sqrt{\kappa} + g_R^2\lambda - \lambda^2 kq_1/p_1 \; < \; 0. \tag{4.22}$$

To obtain eq.4.20 we have squared both (positive) sides of the inequality in eq.4.21. The expression on the left side of eq.4.22 is a parabola in $\sqrt{\kappa}$ with a minimum at the vertex. Thus we require

$$\sqrt{\kappa} \quad < \quad \frac{2\lambda g_1\sqrt{kq_1/p_1} + \sqrt{4\lambda^2 g_1^2 kq_1/p_1 - 4(g_R^2-g_1^2)(g_R^2\lambda - \lambda^2 kq_1/p_1)}}{2(g_R^2-g_1^2)} \tag{4.23}$$

$$= \quad \frac{\lambda g_1\sqrt{kq_1/p_1} + g_R\sqrt{\lambda(g_1^2 + \lambda kq_1/p_1 - g_R^2)}}{g_R^2 - g_1^2} \tag{4.24}$$

For many practical parameter sets the term $\lambda k q_1/p_1$ is dominating over ${g_R}^2 - {g_1}^2$ (e.g., asymptotically for high-fidelity, $n \to \infty$, and sub-logarithmic $k(n)$; cf. section 4.4.2). In this case we can simplify eq.4.24 to

$$\sqrt{\kappa} < \frac{\lambda g_1 \sqrt{k q_1/p_1} + g_R \sqrt{\lambda^2 k q_1/p_1}}{g_R^2 - g_1^2} \tag{4.25}$$

$$= \frac{\lambda \sqrt{k q_1/p_1}(g_1 + g_R)}{g_R^2 - g_1^2} \tag{4.26}$$

$$= \frac{\lambda \sqrt{k/p_1 - k}}{g_R - g_1} \tag{4.27}$$

What do we get from this analysis? We have derived a maximal κ from which we can infer a threshold ($> x_1 = x_R$) such that the average number $\#_1$ of *addressed* super-threshold neurons *exceeds* the average number $\#_R$ of *non-addressed* super-threshold neurons. Clearly this alone does not guarantee a good retrieval result, in particular not for the classical Willshaw model. However, we have postulated for spiking associative memory that the first firing neurons during a retrieval determine the retrieval result (see section 4.3). For a small $\#_R$ (e.g., $\#_R = 1$) eq. 4.24 (with p_1 as in eq. 3.34 for high-fidelity) guarantees that the first spike on average origins from an addressed neuron. If we strictly implement the *instantaneous Willshaw strategy* (see section 4.3) then after the first spike the distribution of the non-addressed neurons will be weakened by a factor of p_1. I.e., only the neurons connected to the first spiking neuron will still have a chance to fire. Thus if p_1 is not too large (e.g., $p_1 < 0.5$) the majority of the competing non-addressed neurons will lose this chance. This is true especially for ultra-sparse patterns with sub-logarithmic k and $p_1 \to 0$ (cf. section 3.2.1).

Since the analysis applies only for the average case it may happen that the first spiking neuron does not belong to the addressed pattern. Then the retrieval will usually be aborted (cf. section 4.3) with only a few neurons spiking such that the miscarried retrieval can be detected. Then a subsequent second or third trial (e.g., with newly generated false ones) can lead to a good retrieval with high probability.

Fig.4.6 illustrates the situation for physiological cortical parameters $n = 100000, k = 100, p_1 = 0.1$ and $\lambda = 1, \#_1 = 30, \#_R = 10$ where we have maximized κ according to eq.4.24 and obtained $\kappa = 87.04$. This means that we can expect good retrievals (note that p_1 is quite small) even if the address patterns contains 8704 randomly chosen false ones.

Usually one would choose rather $\#_R = 1$. Then we can expect that $\#_1 = 25$ of the addressed neurons have fired before the first non-addressed neuron would fire. For this case we still have a maximal $\kappa = 63.19$. Additionally we may interpret the relatively low cortical matrix load $p_1 = 0.1$ [27, 65] for $n = 100000$ and $k = 100$ (which is far below the high-fidelity matrix load $p_1 = (\epsilon k/n)^{1/k} = 0.89$ that cortical associative memories must also be robust against missing ones in the address pattern (cf. section 3.3.1). From eq. 3.34 we obtain $\lambda = 0.05$. And even in this case we would still have a maximal $\kappa = 0.1$ meaning that it might be possible to obtain good retrieval if the address pattern contains only 5 correct ones but 10 false ones. However, we have to be cautious with such a small number of active neurons in the address pattern since then our analysis using Gaussians in order to approximate binomials might no longer be valid (see section 4.4.4). Moreover $\lambda = 0.05$ seems not to be very plausible. For more realistic $\lambda = 0.5$ we would still obtain a maximal $\kappa = 15$. Note that the results in this section are based on the classical binomial approximation which can be very bad for finite n (see section 3.6); a refined analysis is conducted in section 4.4.5.

4.4.2 Asymptotic analysis

Now we derive some *asymptotic* results. First we determine how fast g_1 and g_R increase in n. For this we compare $g(n) := G^{-1}(1 - 1/n)$ to $f(n) := (\ln n)^d$. Since g increases monotonically in n the inverse function g^{-1} exists, and we have $f(n)/g(n) = f(g^{-1}(g(n)))/g(n) = f(g^{-1}(x))/x$ for

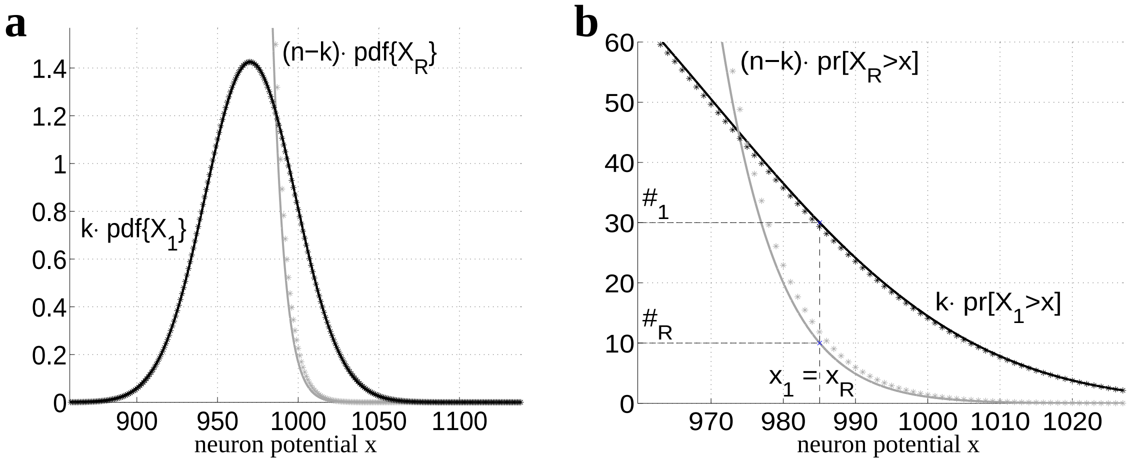


Figure 4.6: Neuron potential distributions when addressing with a single pattern plus noise with cortical parameters as in Fig.4.5 except κ ($n = 100000, k = 100, p_1 = 0.1, \lambda = 1, \kappa = 87.04, \#_1 = 30, \#_R = 10$). Noise parameter κ was chosen maximal according to eq.4.24. **a**: Gaussian (solid) and binomial (*) approximation of the neuron potential distributions of the addressed neurons (X_1, black) and non-addressed neurons (X_R, gray), scaled to obtain the expected neuron potential histograms (bin size 1). **b**: Expected number of neurons exceeding potential x for addressed neurons (black) and non-addressed neurons (gray) using Gaussian (solid) and binomial (*) approximations. We can expect $\#_1$ ($\#_R$) of the addressed (non-addressed) neurons to have a potential of x_1 (x_R) or larger. Noise parameter $\kappa = 87.04$ from eq.4.24 is chosen maximal such that $x_1 \leq x_R$. Usually one would choose $\#_R = 1$ where we still have a maximal $\kappa = 63.19$ (see text). The Gaussian approximation of the binomials is quite good.

$x := g(n)$. With $g^{-1}(x) = 1/(1 - G(x))$ we obtain $f(n)/g(n) = (-\ln(1 - G(x))^d/x$. By applying de l'Hospital's rule one can easily verify that $(1 - G(x))/(\exp(-x^2/2)/x) \to 1/\sqrt{2\pi}$ for $x \to \infty$. Therefore asymptotically $-\ln(1 - G(x)) \approx x^2/2$ and $f(n)/g(n) \approx 0.5^d \cdot x^{2d-1}$. From this we can infer for $d = 0.5$ with $g_R(n) = g(n/\#_R)$ asymptotically

$$g_R{}^2 = (G^{-1}(1 - \#_R/n))^2 \approx 2\ln(n/\#_R) \approx 2\ln n, \tag{4.28}$$

where the latter approximation is true if $\#_R$ is asymptotically less than any root of n.

Next we analyze how fast kq_1/p_1 (cf. eq.4.24) increases in n for high-fidelity $p_1 = (\epsilon k/n)^{1/(\lambda k)}$ (cf. sections 3.1.2 and 3.3.1). For $c := k/\ln n$ we obtain with $q_1 = 1 - p_1$ and $n^{1/\lambda k} = \exp(1/(\lambda c))$

$$kq_1/p_1 = k \cdot \left(\left(\frac{n}{\epsilon k}\right)^{1/(\lambda k)} - 1\right) \tag{4.29}$$

$$= c \cdot \left(\frac{e^{1/(\lambda c)}}{(\epsilon k)^{1/(\lambda k)}} - 1\right) \cdot \ln n. \tag{4.30}$$

$$= c_1 \cdot \ln n \tag{4.31}$$

with $c_1 := c \cdot (e^{1/(\lambda c)}/(\epsilon k)^{1/(\lambda k)} - 1)$. Note that kq_1/p_1 dominates over $g_R{}^2$ asymptotically for sub-logarithmic k with $c_1 \to \infty$ and high-fidelity, which validates the simplified eq.4.27 for that particular case. Generally, if we assume $g_R \gg g_1$ (e.g., $g_1 = 0$ for $\#_1 = k/2$) we obtain from eq.4.24 asymptotically for high-fidelity using eq.4.28 and eq.4.31

$$\sqrt{\kappa} < \frac{\lambda g_1 \sqrt{c_1 \ln n} + \sqrt{2\ln n} \cdot \sqrt{\lambda^2 c_1 \ln n - 2\lambda \ln n}}{2\ln n} \tag{4.32}$$

$$= \frac{\lambda g_1 \sqrt{c_1}}{2\sqrt{\ln n}} + \sqrt{\frac{\lambda^2 c_1}{2} - \lambda} \tag{4.33}$$

For sub-logarithmic k we have $c \to 0$, $c_1 \to \infty$, and $g_1/\sqrt{\ln n} \sim g_1/g_R \leq \sqrt{\ln k/\ln n} \to 0$, and therefore the second term of the sum in eq.4.33 dominates. For logarithmic k we have constant c and c_1 and therefore also the second term in eq.4.33 dominates. For super-logarithmic k we have $c \to \infty$, and with $\exp(x) \to 1 + x$ for $x \to 0$ we obtain $c_1 \to 1/\lambda$. This would result in a negative term under the right square root in eq.4.33, and already a negative discriminant for the parabola in eq. 4.22. This might be interpreted such that for super-logarithmic k we have asymptotically no fault-tolerance against false ones since for any fixed $\kappa > 0$ we obtain for $n \to \infty$ no solution for the inequality of eq. 4.22. On the other hand we should not be too disappointed since we cannot expect good pattern separation anyhow by spiking associative memory for $p_1 \to 1$ implied by super-logarithmic patterns (cf. eq.3.16 in section 3.2). Asymptotically we obtain from this analysis

$$\kappa \leq \begin{cases} \frac{\lambda^2}{2} \cdot (\epsilon k)^{1/(\lambda k)} \cdot \frac{k \cdot n^{1/(\lambda k)}}{\ln n}, & \mathrm{sub-logarithmic}\ k \\ \frac{\lambda^2}{2} \cdot c \cdot (e^{1/(\lambda c)} - 1) - \lambda, & \mathrm{logarithmic}\ k = c \ln n \text{ with } 0 < c < c_0(\lambda) \\ 0, & \mathrm{else.} \end{cases} \tag{4.34}$$

Thus according to this analysis even logarithmic $k = c \ln n$ yields fault-tolerance against false ones only for $0 < c < c_0$ where c_0 depends on λ. For $\lambda = 1$ we can compute $c_0 \approx 0.80$ (and correspondingly $p_1 < 0.29$, cf. eq.3.16 and eq.3.34), for $\lambda = 0.5$ we obtain $c_0 \approx 1.59$ (and $p_1 < 0.28$). Note that the classical optimum for the Willshaw model is obtained for $k = \mathrm{ld}\, n = \ln n/\ln 2 \approx 1.44 \ln n$ (see section 3.1) where also no fault tolerance is possible for $\lambda = 1$.

Note that for $\kappa = 0$ and $\#_R \leq \epsilon k$ these results contradict the classical asymptotical analysis of section 3.1.2: Our high-fidelity choice of $p_1 = (\epsilon k/n)^{1/(\lambda k)}$ actually guarantees that for $\kappa = 0$ we have good retrievals even for the classical one-step algorithm. In contrast, the analysis in this section suggests that for logarithmic or super-logarithmic k even for $\kappa = 0$ there are no good retrievals possible. This paradox is due to the Gaussian approximation of the binomial distribution in the present calculations. In section 4.4.4 we will argue that the Gaussian approximation is not adequate for small $\kappa \to 0$.

As will be further analyzed in section 4.4.4 we also have to be cautious for sub-logarithmic k since the approximation of binomials by Gaussians may no longer be valid. It turns out that the approximations (and therefore also eq.4.34) are valid if we have at least $k = (\log n)^d$ for $d > 0.5$. If k is asymptotically smaller than $\sqrt{\ln n}$ then the analysis in this section may overestimate the robustness against false ones (cf. Fig.4.6). For $k = (\log n)^{0.5+\delta}$ we have $\ln n^{1/(\lambda k)} = (\ln n)^{0.5-\delta}/\lambda$. Thus we can infer from eq.4.34 that the maximally possible κ grows faster than $(\ln n)^d$ but slower than n^d for any $d > 0$.

In summary we conclude that spiking associative memory implementing the instantaneous Willshaw retrieval strategy can cope with a very large number of false ones ($\sim \ln n$ or even $\sim k^2 n^{1/\lambda k}/\ln n = O((\ln n)^d)$ for any $d > 0$) in the address pattern at least if the memory load is not too high. The best results can be expected for sub-logarithmic k and $p_1 \to 0$. It is remarkable that for $\lambda = 1$ and $p_1 \to 0$ we can have storage capacity 1 when applying optimal matrix compression (see section 3.2.1) and at the same time we can have the described fault tolerance against false ones.

4.4.3 Superpositions of several noisy address patterns

Now consider addressing with a superposition of N noisy versions of the original patterns where for $i = 1, 2, ..., N$ the i-th original pattern contributes with $\lambda_i k$ correct ones ($0 < \lambda_i \leq 1$), and additionally the address patterns contains κk false ones ($\kappa \geq 0$) not related to the involved original patterns. If we neglect overlaps within the involved original pattern parts we can now distinguish $N + 1$ potential distributions corresponding to the N groups of addressed neurons and the remaining non-addressed neurons. Similarly as for $N = 1$ in section 4.4.1 the distributions of the corresponding random variables $X_1, X_2, ..., X_N, X_R$ can be approximated by binomials or

Gaussians, and for $i = 1, 2, ..., N$ we have

$$X_i \sim \lambda_i k + B_{(\kappa+\sum_{j\neq i}\lambda_j)k,p_1} \approx N_{\mu_i,\sigma_i} \tag{4.35}$$

$$X_R \sim B_{(\lambda+\kappa)k,p_1} \approx N_{\mu_R,\sigma_R} \tag{4.36}$$

where the parameters of the Gaussians are

$$\mu_i = (\lambda_i + (\kappa + \sum_{j=1,j\neq i}^{N} \lambda_j)p_1)k \tag{4.37}$$

$$\mu_R = (\kappa + \sum_{j=1}^{N} \lambda_j)p_1 k \tag{4.38}$$

$$\sigma_1 = \sqrt{(\kappa + \sum_{j=1,j\neq i}^{N} \lambda_j)p_1 q_1 k} \tag{4.39}$$

$$\sigma_R = \sqrt{(\kappa + \sum_{j=1}^{N} \lambda_j)p_1 q_1 k}. \tag{4.40}$$

For $\lambda_1 \geq \lambda_i$ for $i = 1, 2, ..., N$ we can again determine values x_1 and x_R such that on average $\#_1$ of the k neurons of the first most strongly addressed group have a potential $\geq x_1$, and $\#_R$ of the $n - Nk$ non-addressed neurons have a potential $\geq x_R$,

$$x_1 = \sigma_1 g_1 + \mu_1 \tag{4.41}$$

$$x_R = \sigma_R g_R + \mu_R \tag{4.42}$$

with

$$g_1 := G^{-1}(1 - \#/k) \tag{4.43}$$

$$g_R := G^{-1}(1 - \#/(n - Nk)). \tag{4.44}$$

For $\#_R \leq \#_1$ and $n - Nk > k$ (we should assume N to be a *small* integer) we still have $g_R > g_1$, and the situation is very similar to $N = 1$ in section 4.4.1. If we require again $x_1 > x_R$ this condition is equivalent to eqs. 4.17- 4.22 if λ and κ are replaced by λ_1 and

$$\kappa_1 := \kappa + \sum_{j=2}^{N} \lambda_j, \tag{4.45}$$

respectively. Therefore we can demand from eqs.4.24 and 4.27

$$\sqrt{\kappa_1} < \frac{\lambda_1 g_1 \sqrt{kq_1/p_1} + g_R\sqrt{\lambda_1(g_1^2 + \lambda_1 kq_1/p_1 - g_R^2)}}{g_R^2 - g_1^2} \tag{4.46}$$

$$\approx \frac{\lambda_1\sqrt{k/p_1}}{g_R - g_1}, \tag{4.47}$$

where the latter approximation is valid for $\lambda_1 kq_1/p_1 \gg g_R^2 - g_1^2$. From this and eq.4.45 we obtain

$$\kappa < \frac{{\lambda_1}^2 \cdot k/p_1}{(g_R - g_1)^2} - \sum_{j=2}^{N} \lambda_j \tag{4.48}$$

where for high-fidelity with $p_1 = (\epsilon k/n)^{1(\lambda k)}$ the asymptotic analysis can be conducted analogously to section 4.4.2.

Again note that fulfilling eq.4.48 (or eq.4.46) alone does not guarantee a good retrieval result. It guarantees only that on average the first spike corresponds to one of the addressed neurons. Then it is possible by implementing the *instantaneous Willshaw retrieval strategy* to pop-out the corresponding addressed patterns. But for this we need in addition a low memory load p_1, in particular if the addressed patterns are excited equally, i.e., $\lambda_1 = \lambda_2 = ... = \lambda_N$.

4.4.4 Relevance and applicability of the analysis

In sections 4.4.1 and 4.4.2 we have analyzed spiking Willshaw associative memory with respect to the number of false ones in the address pattern. The results of this analysis are somewhat counter-intuitive. Numerical simulations rather showed that classical Willshaw associative memory is quite sensitive to false ones in the address patterns [160]. In contrast, this analysis suggests that the Willshaw model can be robust against a very large number of false ones κk with $\kappa \geq \ln n \to \infty$. However, we have to be cautious since evaluation of eq. 4.24 for practical values of n, k, ϵ sometimes yields implausible values for κ. For example using parameters $n = 100000, k = 500, p_1 = 0.1$ and $\lambda = 1, \#_1 = 30, \#_R = 10$ (which still are plausible values for cortex, cf. section 2.3.3; cf. Fig. 4.6) would result in $\kappa = 960$. Thus the possible number of false ones $\kappa k = 480000$ would exceed the number of neurons $n = 100000$ which obviously cannot be true.

Where does this discrepancy origin from? There are at least three possible pitfalls where the results from the analysis above should *not* be applied! First it may be that the analysis is correct, but nevertheless the results are irrelevant for a matrix load $0.5 < p_1 < 1$ since then we cannot expect good retrievals because the separation will be very difficult even if the first occurring spikes are correct. Second, the analysis may be invalid because the Gaussians do not sufficiently well approximate the binomials. Third, the analysis may be invalid because already the binomial approximation of the true distribution of membrane potentials is inadequate (cf. section 3.6). In the following we will discuss the three corresponding requirements in more detail.

Only a low matrix load $p_1 < 0.5$ allows good pattern separation

Our analysis so far for the possible number of false ones κk in address patterns guarantees only that on average the addressed neurons will fire first, but possibly followed immediately by non-addressed ones. As mentioned in sections 4.3 and 4.4.1, for good retrievals we require that the first correct spikes suppress the non-addressed (i.e., non-correct) neurons by immediate feedback. Implementing strictly the instantaneous Willshaw retrieval strategy (see section 4.3) the first correct spike will exclude a fraction $1 - p_1$ of the non-addressed neurons from spiking. And if the first i spikes are correct this will even exclude a fraction $1 - {p_1}^i$ of the non-addressed neurons. Obviously this suppression mechanism can only be effective if the matrix load p_1 is small. In principle, the situation after the i-th spike can be analyzed similarly as in section 4.4.1. Reconsider the situation illustrated by the curves $k \cdot \text{pr}[X_1 > x]$ and $(n-k) \cdot \text{pr}[X_R > x]$ in Fig. 4.6b. We have computed a maximal κ such that we can expect the firing of $\#_1$ addressed (correct) neurons before $\#_R$ non-addressed (false) neurons, i.e., such that $x_1 \geq x_R$. After the i-th (correct) spike we can similarly compute $x_{1,i}$ and $x_{R,i}$ such that on average $\#_1$ addressed and $\#_R$ non-addressed neurons have potentials larger than $x_{1,i}$ and $x_{R,i}$, respectively. For $i \ll k$ we can write simply

$$k \cdot \text{pr}[X_1 \geq x_{1,i}] - i = \#_1 \quad (!) \tag{4.49}$$

$${p_1}^i \cdot (n-k) \cdot \text{pr}[X_R \geq x_{R,i}] = \#_R \quad (!). \tag{4.50}$$

If we can prove that for this we still obtain

$$x_{1,i} \geq x_{R,i} \tag{4.51}$$

then we can be at least as confident to obtain again a correct spike as in the beginning of the retrieval. Obviously we can always fulfill eq.4.51 by requiring a sufficiently small p_1.

The requirement of a small $p_1 \leq 0.5$ fits well with the results from section 3.2.1 where we obtained storage capacity 1 asymptotically for ultra-sparse patterns with sub-logarithmic $k <$

$c \log n$ and $p_1 \to 0$. In contrast, our analysis will overestimate the robustness of spiking associative memory with respect to false ones in the address pattern if $p_1 > 0.5$ as it is the case, e.g., for high-fidelity and super-logarithmic patterns $k > c \log n$ where $p_1 \to 1$ (cf. section 3.2.2).

The Gaussian approximation of the potential distribution must be adequate

It is possible that the approximation of the binomials by Gaussians is not valid. As discussed in section 4.4.1 according to the DeMoivre-Laplace theorem we require $np(1-p) \gg 1$ for a $B_{n,p}$ binomial distribution. Thus for our analysis we have to require from eq. 4.7

$$\sigma_1{}^2 = \kappa \cdot k \cdot p_1 \cdot (1 - p_1) \gg 1 \quad (!). \tag{4.52}$$

Clearly this requirement is not always met for practical finite values of n, k, p_1. Asymptotically for high-fidelity we can infer from eq. 4.34 in which cases eq. 4.52 can be fulfilled. For sub-logarithmic k we have $\kappa \sim kn^{1/(\lambda k)/\ln n}$, $p_1 \sim 1/n^{1/(\lambda k)}$, and therefore $\sigma_1{}^2 \sim k^2/\ln n$. Thus in this case eq. 4.52 is met only for $k \gg \sqrt{\ln n}$. For logarithmic $k = c \ln n$ we have constant p_1 and from eq. 4.34 we obtain also constant κ if c is sufficiently small (otherwise we can expect no fault tolerance against false ones; see below). Thus we have $\sigma_1{}^2 \sim k \to \infty$ and therefore eq. 4.52 is met.

Both for logarithmic $k = c \ln n$ with $c \geq c_0$ (and constant p_1) and super-logarithmic k (with $p_1 \to 1$) we can infer from eq. 4.34 that the maximally possible κ is zero or negative (or more exactly, eq.4.17 has no solution). Thus even for $\kappa = 0$ on average the non-addressed (false) neurons should fire first. This contradicts the classical high-fidelity analysis of sections 3.1.2 and 3.3.1 which guarantees for $\kappa = 0$ a good separation between addressed and non-addressed neurons where even for classical one-step retrieval we would expect only $\leq \epsilon \cdot k$ false one-entries (see section 3.1.2). For $\kappa \to 0$ it may be that eq. 4.52 is not met although this depends on how fast κ converges to zero. But there is also another problem for $\kappa = 0$ with a threshold $x_1 = x_R = \lambda k$: Although the variance of the non-addressed neurons is $\sigma_R{}^2 = \lambda \cdot k \cdot p_1 \cdot (1 - p_1) \sim \ln n \to \infty$ the expected number of super-threshold non-addressed (false) neurons is according to the Gaussian approximation $\#_R = (n - k) \cdot \mathrm{pr}[X_R \geq \lambda k] \approx (n - k) \cdot (1 - G(\sqrt{\lambda k (1 - p_1)/p_1}))$ (cf. eq. 4.16). This turns out to be a massive overestimation with $\#_R/k \to \infty$ instead of $\#_R/k \approx \epsilon$ as suggested by the classical analysis: From section 4.4.2 we know $1 - G(x) \sim \exp(-x^2/2)/x$. For high-fidelity and logarithmic $k = c \ln n$ with $x \approx \sqrt{\lambda c (\ln n)(1 - p_1)/p_1}$ and constant p_1 we obtain $\#_R \sim n^{1-\lambda(1-p_1)p_1/2}/\sqrt{\ln n}$ and therefore $\#_R/k \to \infty$. For high-fidelity and super-logarithmic k with $p_1 \approx \ln(n/k)/k$ (see eq.3.22) we have $x \approx \sqrt{\lambda \ln(n/k)}$ and $\#_R \sim n^{1-\lambda/2} k^{\lambda/2}/\sqrt{n/k}$ and therefore also $\#_R/k \to \infty$. Thus the reason for the paradox is simply that the Gaussian approximation does not take into account the limitation of the binomial random variable X_R to values $\leq \lambda k$ for $\kappa = 0$. To avoid this paradox we have to require additionally

$$\frac{\mu_1 - \mu_R}{\sigma_R} = \sqrt{\frac{\lambda^2}{\lambda + \kappa} \cdot \frac{1 - p_1}{p_1} \cdot k} \quad \ll \quad 1 \quad (!), \tag{4.53}$$

where μ_1, μ_R, and σ_R are as in section 4.4.1.

The binomial approximation of the potential distribution must be adequate

In the initially mentioned example we obtained an absurd maximal $\kappa = 960$ for $n = 100000$ and $k = 100$, i.e., $\kappa \cdot k > n$. This senseless result cannot be explained by a bad approximation of the binomial by a Gaussian. It rather shows that the binomial approximation of the true distribution of the membrane potentials can be bad. Using a binomial distribution assumes that the active synapses are generated independently of each other. This is not true since storage of one pattern activates k^2 synapses causing dependencies within the memory matrix. In particular if k is relatively large (e.g., super-logarithmic) one observes characteristic modulations in the histograms of membrane potentials (cf. Fig.3.5a,b). Moreover, if the activity of an address pattern (number z of one-entries) is too large (i.e., in the order of n) then the binomial approximation massively underestimates the true variance of the membrane potential distribution although the mean is still correct (see Fig.3.5c,d).

These phenomena have been investigated in detail in section 3.6. It turned out that the variance ${\sigma_W}^2 \sim z + z^2 \cdot k/n$ of the true Willshaw distribution of the membrane potentials is quadratic in the pattern activity $z := (\lambda + \kappa)k$ while the variance ${\sigma_B}^2 \sim z$ of the binomial approximation is only linear in z. This explains why the binomial approximation massively underestimates the variance of the non-addressed neurons' potentials and therefore overestimates the fault tolerance of the Willshaw model. For the sake of completeness we rewrite the conditions for avoiding oscillatory modulations (eq. 3.100),

$$z \cdot (\frac{k}{n})^2 \cdot (1 - \frac{k}{n})^{Mk/n} \quad \ll \quad 1, \tag{4.54}$$

and for avoiding underestimation of the variance (eq. 3.124),

$$z \quad < \quad 1 + \delta^2 \cdot \frac{p_1}{-(1-p_1)\ln(1-p_1)} \cdot \frac{n}{k}. \tag{4.55}$$

In the next section we will repeat the analysis of section 4.4.1 by approximating the true Willshaw distribution by Gaussians with the correct variances (cf. eq. 3.122).

4.4.5 Refined analysis

In section 3.6 we have seen that the classical binomial analysis of the Willshaw model of associative memory is only valid for a very large number of neurons n, and sub-linear address pattern size $z := (\lambda + \kappa)k$ where k is the pattern activity (number of one-entries in a stored pattern), λ is the fraction of the original k one-entries used in the address pattern, and κk is the number of false one-entries in the address pattern.

For small n or large z the binomial approximation B_{z,p_1} of the membrane potential distribution underestimates the true variance of the Willshaw distribution $W_{z,n,k,M}$ (see section 3.6.1; cf. eqs. 3.123 and 3.122). In the last sections we used the binomial approximation to analyze retrievals in a spiking variant of the Willshaw model. We have derived a formula for the maximal possible κ, i.e., for the maximal number of false one-entries in an address pattern such that we still can expect a good retrieval (see eqs. 4.24 and 4.34). The binomial analysis is correct asymptotically for large $n \to \infty$, sub-linear k, and $z \ll n$ (cf. eq.4.55). For general parameters n, k and p_1 we can try to apply a refined analysis using the Willshaw distribution ([29, 171], see section 3.6).

Similar as in section 4.4.1 we consider addressing with one of the M original address pattern vectors (length n) containing only λk out of k ones of the original address pattern ($0 < \lambda \leq 1$), but also κk false ones ($\kappa \geq 0$). If we further assume a matrix load p_1 (which should equal $(\epsilon k/n)^{1/(\lambda k)}$ if we require high-fidelity, cf. section 3.1.2) the potentials of addressed neurons and the remaining non-addressed neurons are distributed according to random variables X_1 and X_R, respectively,

$$X_1 \quad \sim \quad \lambda k + W_{\kappa k,n,k,M} \approx N_{\mu_1,\sigma_1} \tag{4.56}$$

$$X_R \quad \sim \quad W_{(\lambda+\kappa)k,n,k,M} \approx N_{\mu_R,\sigma_R} \tag{4.57}$$

We assume that the Willshaw distribution $W_{z,n,k,M}$ can be approximated by Gaussians $N_{\mu,\sigma}$ with corresponding expectation μ as in eq.3.101 and variance σ^2 as in eq.3.122. As a working hypothesis we can assume that this Gaussian approximation is good for $\sigma^2 \gg 1$, similarly as for the Gaussian approximation of a binomial according to the DeMoivre-Laplace theorem (cf. eq. 4.52).

Thus

$$\mu_1 = (\lambda + \kappa p_1)k \tag{4.58}$$

$$\mu_R = (\lambda + \kappa)p_1 k \tag{4.59}$$

$$\sigma_1 = \sqrt{\kappa p_1 q_1 k - (\kappa^2 k^2 - \kappa k)\frac{k}{n}{q_1}^2 \ln q_1} \tag{4.60}$$

$$\sigma_R = \sqrt{(\lambda + \kappa)p_1 q_1 k - ((\lambda + \kappa)^2 k^2 - (\lambda + \kappa)k)\frac{k}{n}{q_1}^2 \ln q_1} \tag{4.61}$$

with $q_1 := 1 - p_1$. Similarly as in section 4.4.1 we require that on average $\#_1$ of the address neurons have at least potential x_1, and $\#_R$ of the non-addressed neurons have at least potential x_R. With the same definitions of g_1 and g_R as in section 4.4.1 we choose typically large $\#_1$ (e.g., $\#_1 = k/2$) and small $\#_R$ (e.g., $\#_R = 1$), and require

$$x_1 > x_R \quad (!) \tag{4.62}$$

$$\Leftrightarrow \quad \sigma_1 g_1 + \mu_1 > \sigma_R g_R + \mu_R \tag{4.63}$$

$$\Leftrightarrow \quad g_R\sqrt{\lambda+\kappa}\sqrt{p_1 - ((\lambda+\kappa)k - 1)\frac{k}{n}q_1 \ln q_1} - g_1\sqrt{\kappa}\sqrt{p_1 - (\kappa k - 1)\frac{k}{n}q_1 \ln q_1} < \lambda\sqrt{kq_1} \tag{4.64}$$

Note that for $k/n \to 0$ the two large roots become simply p_1 and eq. 4.64 becomes equivalent to eq.4.20. In contrast to eq.4.20, however, it is difficult to resolve eq.4.64 for κ (cf. eq. 4.24). For given n, k, M, λ we can compute numerically the maximal κ. In section 4.5.4 we will compare the maximal κ computed from eq. 4.64 and eq. 4.24 with the results from numerical simulations of the spiking Willshaw model. Figure 4.7 shows that the refined analysis approximates the true Willshaw distribution quite well.

4.5 Technical implementations of the spike counter model

There are basically two possibilities how to implement one or more interconnected spike counter neuron populations: (1) As a dynamical system without a global clock; (2) In discrete time steps defined by a global clock: In each time step a retrieval is performed in each of the spike counter populations. The latter possibility (2) can be further subdivided: (2a) The retrieval result is the set of spikes in the populations, (i.e., a set containing for each spike a tuple (i, t) where i is the neuron index and t is the spike time; (2b) The retrieval result is only the set of activated neurons whereas the spike *times* are not used here.

There are certainly efficient technical implementations of type (1) conceivable. However, we have used a type (1) implementation only for simulation of a biologically more realistic model (see section 4.6; cf. also chapter 5). A technical implementation of type (2a) has already been described in [86, 89]. In this work a type (2b) implementation is used which has several advantages: (i) type (2b) is more in accordance with the analysis of the spike counter model in section 4.4 where the address pattern contains also no information about the spike times; (ii) if implementing many interconnected associative areas a type (2b) architecture is compatible with classical non-spiking algorithms (like one-step retrieval) on the global level, i.e., it is more easily possible to encapsulate the local retrieval algorithm; (iii) when simulating only one spike counter population we need not to consider the spike times of the address pattern, i.e., we have less free parameters; (iv) when simulating many populations we do not have the problem (or freedom) of how to mix address inputs from different populations to the same population.

Of course we may loose valuable information by ignoring the spike times on the global level because generally the earliest spikes are most relevant. However, here we are interested mainly in the evaluation of our analysis, and in general results about how efficient spiking associative memory can be implemented on digital computers. In the following we will describe straight-forward extensions of the spike counter model for a type (2b) implementation of many interconnected associative areas [95].

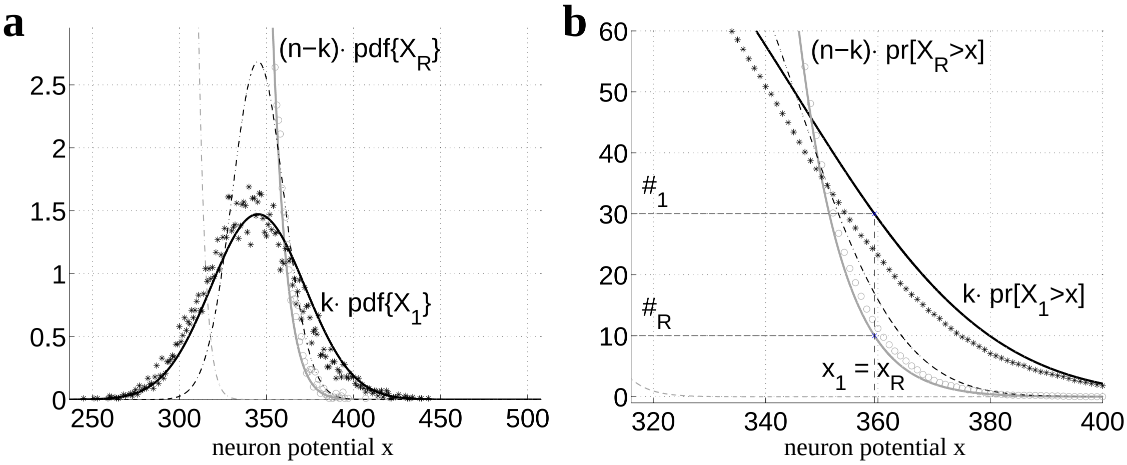


Figure 4.7: Refined analysis using a Gaussian approximation of the Willshaw distribution (see text) for neuron potential distributions when addressing with a single pattern plus noise with physiological (i.e., cortical) parameters $n = 100000, k = 100, p_1 = 0.1, \lambda = 1, \kappa = 24.52, \#_1 = 30, \#_R = 10$ (same parameters as in Fig.4.6 except for κ). Noise parameter $\kappa = 24.52$ was chosen maximal according to eq.4.64. Note the difference to the results using the binomial approximation ($\kappa = 87.04$) in Fig.4.6. **a**: Averaged neuron potential distributions of the addressed neurons (X_1, black, $*$) and non-addressed neurons (X_R, gray, $\circ$), scaled to obtain the expected neuron potential histograms (bin size 1). The true potential histogram of the addressed neurons (black $*$) is slightly asymmetric which is not captured by the refined Gaussian approximation (black solid). The refined Gaussian approximation for the non-addressed neurons (gray solid vs. $\circ$) is quite good. Generally, the refined Gaussian approximations are much better than the Gaussian approximation of the classical binomial approximations (dash-dotted; cf. Fig.4.6). **b**: Expected number of neurons exceeding potential x for addressed neurons (black) and non-addressed neurons (gray) computed from the cumulations of the corresponding histograms shown in (a). The refined Gaussian approximation is quite good for the non-addressed neurons (gray solid vs. $\circ$) while it leads to significant errors for the addressed neurons (black solid vs. $*$) due to the described asymmetry, cf. plot (a). The Gaussian approximation of the classical binomial approximation (dash-dotted) is bad. Estimations of the Willshaw potential distributions ($*, \circ$) were computed by averaging potential histograms for 100 different address patterns.

4.5.1 Model extension

Figure 4.8 illustrates the basic structure of the spike counter model of associative memory for a type (2b) implementation of many interconnected neuron populations or associative areas (see above). A local population of n neurons constitutes an associative area. Each neuron i has five state variables:

1. Membrane potential x_i.
2. Counter ${c^H}_i$ for spikes received hetero-associatively from N other cortical areas (binary memory matrices H_1, H_2, ..., H_N). In this model variant ${c^H}_i$ is not really a counter since the received spikes are *weighted* by connection strengths c_j of the respective hetero-associative input connections ($j = 1, 2, ..., N$) or the last output state of the neuron ($j = 0$). The latter can be used to implement a kind of working memory.
3. Counter ${c^A}_i$ for spikes received by immediate auto-associative feedback via the binary memory matrix A.
4. Counter c^Σ for the total number of spikes that already occurred during the current retrieval within the local area. This counter is the same for all local neurons.

5. Output state y_i. Initially $y_i = 0$ for all neurons. As soon as the potential x_i exceeds a threshold Θ the output state is set to $y_i = 1$.

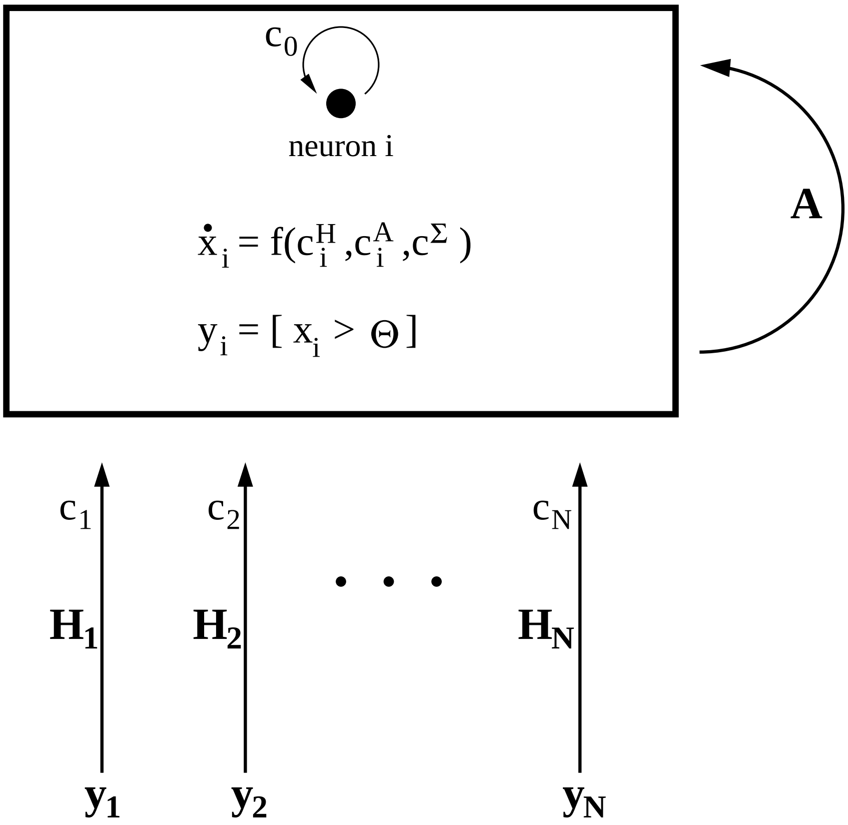


Figure 4.8: Implemented (slightly extended) spike counter model of the Willshaw associative memory. A local population of n neurons receives external input via N hetero-associative connection matrices $\mathbf{H}_j$ each weighted with factor c_i. Additional input comes from the old output state $\mathbf{y}^{\text{old}}$ of the population weighed with c_0. This input initiates the retrieval where local spikes are immediately fed back via the auto-associative connection matrix $\mathbf{A}$. The counters $c^H{}_i, c^A{}_i, c^\Sigma$ represent the number of spikes a neuron receives hetero-associatively and auto-associatively, and summed over the whole local population, respectively.

While in the classical Willshaw model (see chapter 3) synaptic input is summed to obtain the membrane potential, in the spike counter model synaptic input rather determines the *temporal change* of the membrane potential (see section 4.3). This is similar to real neurons where synaptic input causes synaptic *currents* which in turn load the membrane (see section 2.4). The derivative of the membrane potential of a neuron i is a function f of its spike counters (see eq. 4.1). We can implement an instantaneous variant of the Willshaw retrieval strategy (see sections 3.1.1 and 4.3) if we choose the function f such that f is positive if $c^A{}_i \approx c^\Sigma$, and negative for $c^A{}_i \ll c^\Sigma$. We use the simple linear example of eq. 4.2,

$$\frac{dx_i}{dt} = f_1(c^H{}_i, c^A{}_i, c^\Sigma) := ac^H{}_i + b\left(c^A{}_i - \alpha c^\Sigma\right), \tag{4.65}$$

where we can choose $a \ll b$ and $\alpha \approx 1$ which is also neurophysiologically plausible ([27]; see sections 2.3.3 and 2.1.3).

Any type (2b) implementation of the spike counter model (see above) should compute a mapping

$$(\mathbf{y}^{\text{old}}, \mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_N}) \quad \mapsto \quad \mathbf{y} \tag{4.66}$$

from the relevant old activity states of the network at time $t-1$ (where $\mathbf{y}^{\text{old}}$ is the old output state of the local population, and $\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_N}$ are the old activity states of other areas delivering input to the local population) to the new activity state $\mathbf{y}$ of the local population at time t. In the following we discuss efficient sequential and parallel implementations of the described spike counter model.

4.5.2 Sequential implementations

An efficient implementation of the spike counter model as described in section 4.5.1 is given by the following algorithm (cf. [86, 89, 95]):

```
1.  $\mathbf{c^A} := \mathbf{0}$;   $\mathbf{c}^{\Sigma} := \mathbf{0}$;   $\dot{\mathbf{x}} := \mathbf{0}$;   $\mathbf{y} := \mathbf{0}$;
2.  $\mathbf{c^H} := c_0 \cdot \mathbf{y}^{\text{old}} + c_1 \cdot \mathbf{H_1} \cdot \mathbf{y_1} + ... + c_N \cdot \mathbf{H_N} \cdot \mathbf{y_N}$;
3.  $\mathbf{x} := \mathbf{c^H} - \max\{c_i^H\}$;
4.  $(t_s, j) := (0, \arg\max\{c_i^H\})$;
5.  WHILE $t_s \geq 0$ DO
6.      $\mathbf{x} := \mathbf{x} + \dot{\mathbf{x}} \cdot t_s$;
7.      $y_j := 1$;
8.      $c_j^A := -\infty$;
9.      $\mathbf{c^A} := \mathbf{c^A} + \mathbf{A_j}$;
10.     $\mathbf{c}^{\Sigma} := \mathbf{c}^{\Sigma} + \mathbf{1}$;
11.     $\dot{\mathbf{x}} := a \cdot \mathbf{c^H} + b \cdot (\mathbf{c^A} - \alpha \cdot \mathbf{c}^{\Sigma})$;
12.     $S^{\text{next}} := \{(x_i/\dot{x}_i, i)\}$;
13.     $(t_s, j) :=$ next spike in $S^{\text{next}}$ (or $t_s = -1$ if not possible)
14. ENDWHILE
```

Line 1 initializes all the state variables. In line 2 the working memory effect is implemented, i.e., the last output vector $\mathbf{y}^{\text{old}}$ of the local neuron population is weighted with c_0. Together with hetero-associative synaptic input from N external areas this yields the spike counter vector $\mathbf{c^H}$. Line 3 initializes the membrane potentials with $\mathbf{c^H}$ such that only the most excited neurons reach the threshold $\Theta = 0$. Line 4 determines the first neuron that will spike (neuron j at time t_s). The WHILE loop of lines 5 to 14 iterates as long as there still exists a neuron j that is about to spike at time t_s. Line 6 integrates the membrane potentials, line 7 sets the output vector for the currently spiking neuron j, line 8 prevents neuron j from emitting a second spike. In lines 9 and 10 the spike counters $\mathbf{c^A}$ and $\mathbf{c}^{\Sigma}$ are updated ($\mathbf{A_j}$ is the j-th row of the auto-associative memory matrix $\mathbf{A}$, and $\mathbf{1}$ is a vector of length n containing only one-entries). In line 11 the temporal change of the membrane potentials is computed as described above, and lines 12 and 13 determine the next spike. If dx_i/dt is negative for all neurons i no more spike will occur, and this will end the algorithm with the retrieval result $\mathbf{y}$.

If the address patterns $\mathbf{y_j}$ for $j = 1, ..., N$ from the N other areas contain on average $z := (\lambda+\kappa)k$ one-entries where k is the number of one-entries in a stored pattern, lines 1 to 4 require $\sim Nzn$ steps. The WHILE loop will be processed about k times, and each loop run will require no more than $\sim n$ steps. Thus a total of

$$t_{\text{ret}} \sim N \cdot z \cdot n + k \cdot n \tag{4.67}$$

$$\sim k \cdot n \tag{4.68}$$

steps is required to perform a retrieval, where for the second approximation constant N and κ have been assumed. The retrieval time remains up to a constant factor the same as for the classical Willshaw model (see eq.3.52).

Compressing the memory matrices has qualitatively the same effect as discussed in chapter 3: For non-logarithmic k the storage capacity is increased, and at the same time retrievals are even accelerated. With respect to storage capacity note that an *efficient* auto-associative retrieval applying matrix compression requires the *whole* memory matrix, i.e., we cannot make use of the symmetry in an auto-associative matrix. Here the asymptotic storage capacity for efficient retrievals in auto-associative memories is limited by 0.5 (instead of 1).

4.5.3 Parallel implementations

The algorithm described in section 4.5.2 can be implemented efficiently also on parallel architectures with n processors, i.e., one processor per neuron: If the address patterns $\mathbf{y_j}$ for $j = 1, ..., N$ from the N input areas contain on average $z := (\lambda + \kappa)k$ one-entries where k is the number of one-entries in a stored pattern, lines 1 to 3 require $\sim Nz$ steps by parallel implementation of sparse vector-matrix-multiplication (see section 3.5). Again we have to run the WHILE loop about k times. Each line within the WHILE loop can be performed in parallel in constant time except for line 13. Here as well as in line 4 the *minimal* spike time must be extracted from n values. This requires still $\mathrm{ld}\, n$ steps for a parallel implementation. Thus we require a total of

$$t_{\mathrm{ret}} \sim N \cdot z + k \cdot \mathrm{ld}\, n \tag{4.69}$$

$$\sim k \cdot \mathrm{ld}\, n \tag{4.70}$$

steps to perform a retrieval, where for the second approximation we assumed constant N and κ. Although this implementation is still quite efficient the spike counter model can *not* be parallelized optimally (then we would have obtained $t_{\mathrm{ret}} \sim k$ as it is the case for the classical Willshaw model; see eq. 3.74).

At least, *compressing* the memory matrix for spiking retrievals bears no additional costs: In section 3.5.2 we argued that an efficient parallel implementation of the *non-spiking* Willshaw model with compressed memory matrix requires a look-up-table with $n \min(p_1, 1 - p_1)/\#$ columns (and two rows) to accelerate access to the sparse matrix-entries (see Fig. 3.4; cf. eq. 3.81). It has been discussed that we need $\# \rightarrow \infty$ in order to guarantee that the physical memory required for the look-up-table is negligible compared to the compressed memory matrix (see eqs. 3.83, 3.86, and 3.89). On the other hand side this will increase the retrieval time compared to the non-compressed case (see eqs. 3.74 and 3.82). The retrieval time for the parallel implementation of the spike counter model with compressed memory matrix is

$$t_{\mathrm{ret}} \sim k \cdot (\mathrm{ld}\, n + \#) \tag{4.71}$$

$$\sim k \cdot \mathrm{ld}\, n. \tag{4.72}$$

Thus if # grows at most logarithmically (as assumed for the second approximation) we have asymptotically no additional costs while it is guaranteed that the size of the look-up table is asymptotically negligible.

An efficient *parallel* implementation of auto-association *without* matrix compression already requires that each neuron has a list of *all* the connected neurons. Thus the symmetry in the auto-associative memory matrix cannot be exploited to save memory, i.e., we can obtain at most a storage capacity of 0.5 (instead of 1). In a similar way we have argued in section 4.5.2 for *sequentially* implementing auto-association *with* matrix compression. Thus if we allow for the luxury of a parallelized spiking associative memory, applying matrix compression bears asymptotically no additional costs.

4.5.4 Simulation experiments

In this section we discuss results of simulation experiments in order to illustrate and verify the analysis of the Willshaw model conducted in this chapter and chapter 3, and to compare the spike counter algorithm (section 4.5.2) to alternative retrieval algorithms such as classical one-step retrieval. For this we have implemented the following *retrieval algorithms*:

1. **Classical one-step retrieval (1step)**: This is the algorithm for hetero-association similar as described in section 3.1.1. In contrast to the Willshaw threshold strategy (which simply chooses the threshold equal to the number of active units in the address pattern; see section 3.1.1) we assumed that the pattern size k is constant and known. Thus the threshold is chosen such that the number of ones in the retrieval result is near k.

2. **Hetero- plus auto-associative one-step retrieval (1step h/a)**: This algorithms starts with a classical hetero-associative one-step retrieval (algorithm 1), but then adds an auto-associative one-step retrieval. Similar to the spike counter algorithm (see below algorithm 4) this algorithm exploits both the hetero-associative and the auto-associative memory matrices.

3. **X-wise one-step retrieval (xwise-1step)**: This algorithm is the one-step version of the bidirectional retrieval algorithm proposed by Sommer and Palm in [171]. Similar as for the first two algorithms a fixed given pattern size k is required. As additional parameters we used standard values as suggested by the libasso.h/c library of Friedrich Sommer (see module F2_libasso.h/cpp in appendix C.1.2): a first-step activity of 3 and boolean anding.

4. **Spike counter algorithm (s-counter)**: This is the algorithm as proposed in section 4.5.2 with parameters $a = 1$, $b = 1000$, and $\alpha = 1$.

In order to evaluate the performance of the different algorithms we used two *quality measures*:

- $r_N \in [0;1]$ is the *normalized transinformation* between the stored pattern and the retrieved pattern (for details see eq. B.16 in appendix B.3.1).

- pr[perfect retrieval]$\in [0;1]$ is the probability that a retrieved pattern is free of any errors (false or missing one-entries).

Each quality value has been estimated from the simulation of 100 retrieval repetitions.

In the following we will discuss the results from three retrieval scenarios: (i) Missing ones in the address pattern, (ii) false ones in the address pattern, and (iii) addressing with superpositions of several of the original address patterns.

Missing ones in the address pattern

As a first step we validate the analysis of Willshaw associative memory in chapter 3, in particular the analysis of what happens when some of the original one-entries of the original address pattern are missing (section 3.3.1). Figure 4.9 shows the results for $n = 10000$ neurons, pattern activity $k = 50$, and $M = 44699$ stored patterns which corresponds to a matrix load of $p_1 \approx 0.67$. The matrix load has been chosen by requiring high-fidelity ($\epsilon = 0.01$; see section 3.1.2) when addressing with only half of the original k one-entries, i.e., for $\lambda = 0.5$ (see eq. 3.34). Then we tested the retrieval quality when addressing with a fraction λk of the original one-entries in the address patterns where we varied λ from 0 to 1.

Fig. 4.9a plots the relation between λ and r_N. For all four algorithms the results are close to what has been predicted from the theory in section 3.3.1. For small $\lambda \ll 0.5$ we have low quality, but for increasing λ also the quality r_N increases. Around $\lambda = 0.5$ we achieve already $r_N \approx 1$. This is exactly what we would expect, since we have chosen the number M of stored patterns (or equivalently, the matrix load p_1) in order to obtain high-fidelity retrievals for $\lambda = 0.5$ (i.e., when the address patterns contain half of the original one-entries).

Fig. 4.9b reveals some differences between the algorithms for the second quality measure, the fraction (or probability) of a perfect retrieval. Here the algorithms *s-counter* and *1step h/a* perform

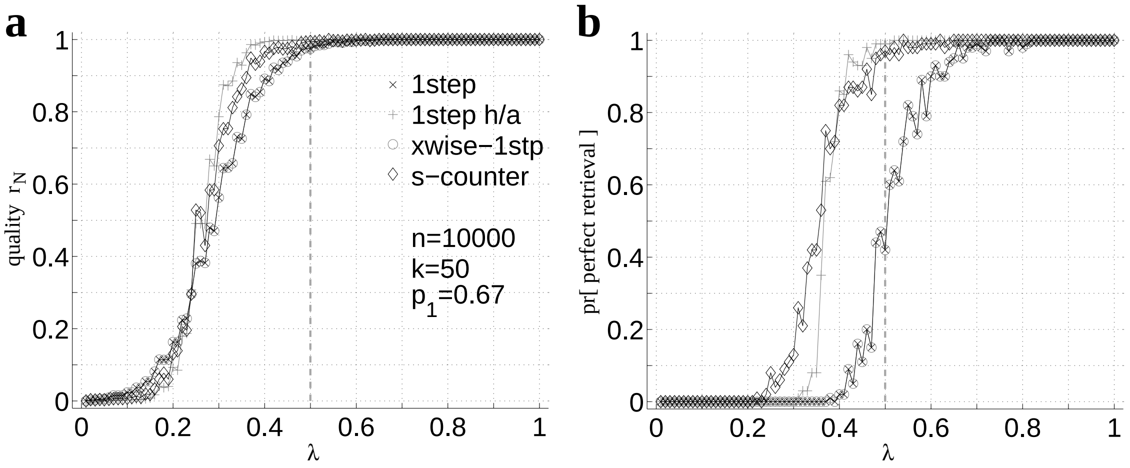


Figure 4.9: Retrieval performance of different one-step algorithms when addressing with λk of the k original one-entries in the address patterns, but without any false one-entries (n = 10000, k = 50, M = 44699). Matrix load $p_1 \approx 0.67$ has been chosen by requiring high-fidelity ($\epsilon = 0.01$) for $\lambda = 0.5$ (see eq. 3.34). Results for the following algorithms are shown: one-step retrieval (×), auto/hetero one-step retrieval (+), xwise one-step retrieval (○), and spike counter retrieval (◇). **a**: Normalized transinformation r_N (see eq. B.16) vs. λ. **b**: Probability of a perfect retrieval vs. λ for the same simulation as for (a). Each plotted value has been estimated from 100 retrievals.

better than the remaining two algorithms. This can be attributed to the fact that both *s-counter* and *1step h/a* make use not only of the hetero-associative matrix but also of the auto-associative matrix of the patterns to be retrieved.

In the following simulation experiment illustrated by Fig. 4.10 it is tested for fixed n whether the pattern activity k has an influence on the retrieval quality. To this end we compared only the two algorithms one-step-retrieval (gray) and spike-counter-retrieval (black) for different pattern activities k =5, 14, 100, and 200.

For the normalized transinformation r_N the curves for different k are very similar (Fig. 4.10a). However, for larger k there is a tendency for a shallower increase from 0 to 1 (e.g., compare k = 100 to k = 200), although this effect is difficult to see for ultra-sparse patterns (e.g., k = 5). One cause for this may be the quadratic dependence of the potential variance for non-addressed neurons on k (see eq. 3.122 in section 3.6.3).

There is a much clearer difference between the different degrees of sparseness when measuring the fraction of perfect retrievals (Fig. 4.10b). However, these differences are mainly due to the high-fidelity requirement limiting the *normalized* number ϵ of expected false one-entries in the retrieval result (see section 3.1.2; cf. eq. B.6). Thus, it is not surprising that for larger k the probability of a perfect retrieval is lower.

False one-entries in the address pattern

Next we tested the different algorithms when addressing with patterns containing κk false one-entries (but no missing one-entries) in order to validate the results from the analysis in section 4.4.1 and to compare the algorithms. Figure 4.11 illustrates the results for different pattern activities k =5,14,100 where in each case M, the number of stored patterns, (or equivalently, the matrix load p_1) has been chosen according to the high-fidelity requirement when addressing with perfect patterns (λ = 1 and κ = 0; see eq. 3.34 in section 3.3.1)

The plots show that for κ = 0 (i.e., no false one-entries in the address pattern) the retrieval performance is quite good in accordance with the chosen matrix load p_1. With increasing κ the performance decreases gradually. For the spike counter algorithm *s-counter* the vertical dashed

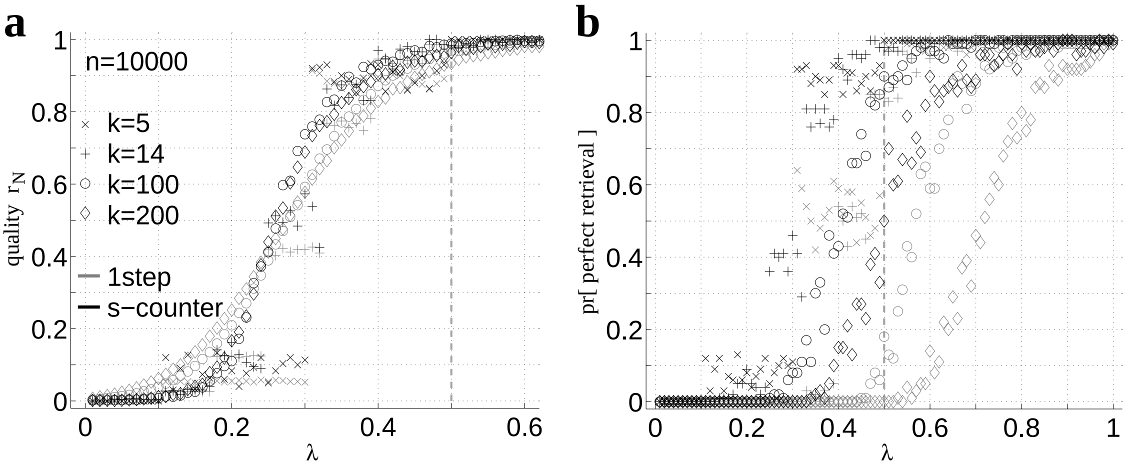


Figure 4.10: Retrieval performance for $n = 10000$ and different pattern activities $k = 5$ ($\times$), $k = 14$ (+), $k = 100$ ($\circ$), and $k = 200$ ($\diamond$) when addressing with λk of the k original one-entries in the address patterns, but without any false one-entries. In each case matrix load p_1 has been chosen by requiring high-fidelity ($\epsilon = 0.01$) for $\lambda = 0.5$ (see eq. 3.34). The plots show results for one-step retrieval (gray) and the spike counter algorithm (black). **a**: Normalized transinformation r_N (see eq. B.16) vs. λ. **b**: Probability of a perfect retrieval vs. λ for the same simulation as for (a). Each plotted value has been estimated from 100 retrievals.

lines in the plots indicate the maximal κ such that we can expect the first spiking neuron to be a correct one ($\#_1 = \#_R = 1$; see analysis in section 4.4), where the thick gray line corresponds to the naive binomial analysis (eq. 4.24 in section 4.4.1), and the thin black line to the refined analysis using Gaussian approximations of the Willshaw distribution (eq. 4.64 in section 4.4.5; cf. section 3.6). The two estimations for maximal κ are quite similar for ultra-sparse ($k = 5$; Fig. 4.11a,d) and classically sparse ($k = 14$; Fig. 4.11b,e), but differ significantly for moderately sparse patterns ($k = 100$; Fig. 4.11c,f).

Similar to the case of *missing* one-entries (cf. Figs. 4.9 and 4.10) the performance of the different algorithms as measured from the normalized transinformation r_N (Fig. 4.11a,b,c; cf. eq. B.16) is quite similar for one-step retrieval (1step) xwise-one-step retrieval (xwise-1step), and the spike counter algorithm (s-counter), although the results for the spike counter algorithm are slightly better than for the other algorithms.

Surprisingly, the combined hetero-/auto-associative one-step retrieval (*1step h/a*) performs worse than the other algorithms. The reason for this is probably that the result of the first hetero-associative one-step retrieval (which is equivalent to algorithm *1step*) contains some false one-entries and therefore missing correct one-entries (since it has been adjusted for pattern activity k). But missing one-entries will result in a bad retrieval result for the subsequent auto-associative step since M and p_1 have been adjusted for address patterns without missing one-entries ($\lambda = 1$; see above). Therefore a better retrieval strategy would be perhaps to lower the threshold in the first hetero-associative step in order to assert that all correct one-entries are active. But this would require additional assumptions on the number of false one-entries contained in the address pattern. A further alternative could be to apply the threshold operation only after the second auto-associative step (but not for the first hetero-associative step).

Even more clearly than for missing-one entries (Figs. 4.9 and 4.10) the difference between the spike counter and the other algorithms can be seen for the retrieval performance as measured from the fraction of perfect retrievals (Fig. 4.11d,e,f). With increasing number of false one-entries in the address pattern it becomes impossible for *1step*, *1step h/a* and *xwise-1step* to maintain perfect retrievals. In contrast, the spike counter algorithm *s-counter* can deliver perfect retrievals even for large κ, in particular for ultra-sparse patterns ($k = 5$).

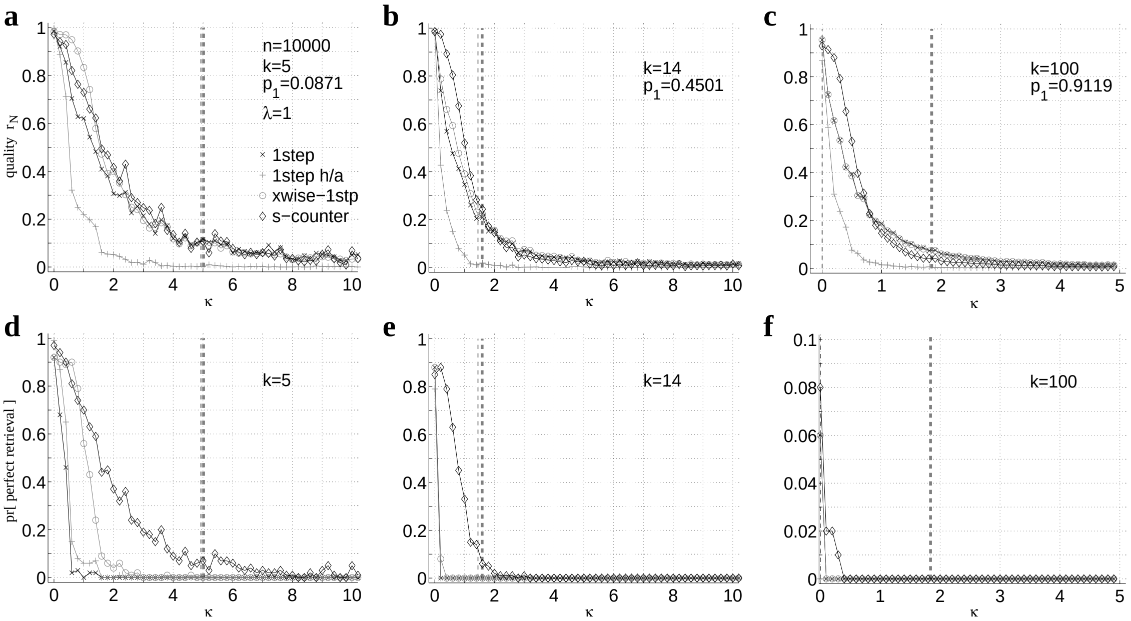


Figure 4.11: Retrieval performance of different one-step algorithms when addressing with all k original one-entries ($\lambda = 1$) but also κk false one-entries ($n = 10000$). In each case matrix load p_1 has been chosen by requiring high-fidelity ($\epsilon = 0.01$) for $\kappa = 0$ and $\lambda = 1$ (see eq. 3.34). Results for the following algorithms are shown: one-step retrieval ($\times$), auto/hetero one-step retrieval ($+$), xwise one-step retrieval ($\circ$), and spike counter retrieval ($\diamond$). *Dashed* lines show maximal κ such that the first firing is correct on average ($\#_1 = \#_R = 1$) as obtained by the binomial approximation (gray; eq. 4.24) and the refined analysis (black; eq. 4.64). Top panels (**a**,**b**,**c**) show normalized transinformation r_N (see eq. B.16) vs. κ. Bottom panels (**d**,**e**,**f**) show probability of a perfect retrieval vs. κ for the same simulations as in the top panels. **a**,**d**: Ultra-sparse patterns ($k = 5$, $M = 364515$, $p_1 = 0.087$) where analysis yields $\kappa < 5.02$ (binomial) and $\kappa < 4.93$ (refined). **b**,**e**: Classically-sparse patterns ($k = 14 \approx \mathrm{ld}\, n$, $M = 305111$, $p_1 = 0.45$) where analysis yields $\kappa < 1.59$ (binomial) and $\kappa < 1.45$ (refined). **c**,**f**: Moderately-sparse patterns ($k = 100$, $M = 24302$, $p_1 = 0.91$) where analysis yields $\kappa < 1.84$ (binomial) and negative κ (refined). Each plotted value has been estimated from 100 retrievals.

Remember that one property of the spike counter algorithm is to break down bad retrievals autonomously: as soon as false neurons begin to spike the activation of further neurons stops. This means that in most cases the spike counter algorithm delivers either a perfect retrieval or the result contains only a few active neurons. Thus in order to maximize the transinformation r_N a variant of the spike counter algorithm could simply add further high-potential neurons in case of an early retrieval break down.

Next we investigated the fault tolerance against κk false one-entries in the address pattern for a lower matrix load. For the simulations shown in Figure 4.12 we required high-fidelity retrievals for $\lambda = 0.5$ (and $\kappa = 0$), i.e., for the case the address pattern contains only half of the original one-entries (but no false one-entries; see section 3.3.1; cf. eq. 3.34).

The results shown in Fig. 4.12 are qualitatively similar to the results for higher matrix loads (Fig. 4.11). Performance of the spike counter algorithm *s-counter* is significantly better than for one-step retrievals *1step* and *1step h/a*, and at least comparable or even better than *xwise-1step*.

For all algorithms the quality measures take optimal value 1 for $\kappa = 0$. For small positive κ qualities remain high on a plateau near 1. But for larger κ the quality measures decrease gradually to zero. As expected from the lower matrix load (and the smaller number of stored pattern) the fault tolerance against false one-entries is quantitatively much better than in Fig. 4.11.

In particular for ultra-sparse patterns fault tolerance is possible for quite large κ. Now the

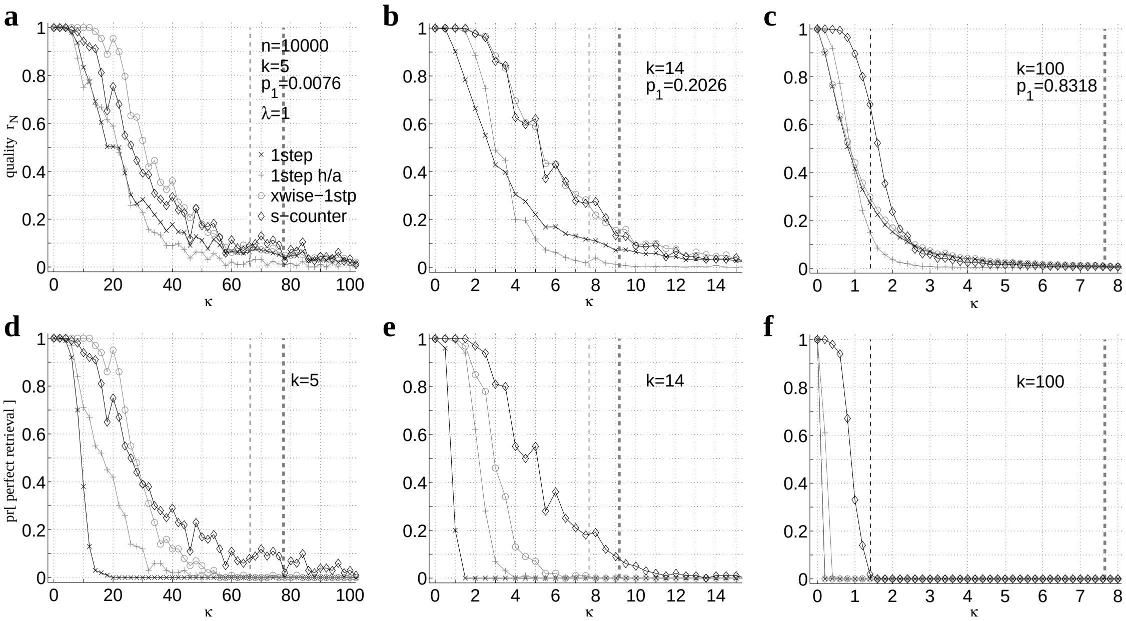


Figure 4.12: Retrieval performance for similar simulations as shown in Fig. 4.11 but where matrix load p_1 was lower. Now p_1 has been chosen by requiring high-fidelity ($\epsilon = 0.01$) even for $\lambda = 0.5$ (see eq. 3.34). Legends are the same as in Fig. 4.11. **a,d**: Ultra-sparse patterns ($k = 5$, $M = 30516$, $p_1 = 0.0076$) where analysis yields $\kappa < 77.57$ (binomial) and $\kappa < 66.25$ (refined). **b,e**: Classically-sparse patterns ($k = 14 \approx \text{ld}\, n$, $M = 115509$, $p_1 = 0.2026$) where analysis yields $\kappa < 9.17$ (binomial) and $\kappa < 7.67$ (refined). **c,f**: Moderately-sparse patterns ($k = 100$, $M = 17825$, $p_1 = 0.8318$) where analysis yields $\kappa < 7.65$ (binomial) and $\kappa < 1.42$ (refined).

maximal possible κ (dashed lines) as predicted from the theory in section 4.4.1 (binomial approximation; see eq. 4.24) and section 4.4.5 (refined analysis; see eq. 4.64) differ significantly also for ultra-sparse ($k = 5$) and classical patterns ($k = 14$). The predictions from the binomial approximation *massively overestimate the fault tolerance*, at least for the moderately sparse case ($k = 100$). In contrast, the predictions from the refined analysis are precise (cf. Fig. 4.7) although the interpretation is not straight-forward since the relation between the retrieval quality and the fact that the first spikes are correct on average is not clear. Of course the predicted maximal κ depends also strongly on the choice of $\#_1$ and $\#_R$ (see section 4.4.1). Here we used $\#_1 = \#_R = 1$ which is the weakest possible quality criterion meaning that the neuron with the highest potential (after hetero-associative one-step retrieval) is on average a correct one. Thus, this criterion just asserts that perfect retrievals are possible. With this argumentation the presented results validate the refined theory quite well: For the predicted maximal κ the fraction of perfect retrievals lies in the range of κ where perfect retrievals become indeed possible (around 10% perfect retrievals; cf. Figs. 4.11, 4.12, and also 4.13).

Addressing with superpositions

In further simulation experiments we tested the different algorithms for different pattern activities $k = 5, 14, 100$ when addressing with superpositions of several patterns. Figure 4.13 illustrates some of the results for the same memory matrices as in Fig. 4.12 and superpositions of two complete address patterns.

Similarly as in the examples before (Fig. 4.9-4.12) we measured the normalized transinformation (Fig. 4.13a,b,c) and the fraction of correct retrievals (Fig. 4.13d,e,f). For determining the transinformation (and to decide if a retrieval was perfect) when addressing with a superposition

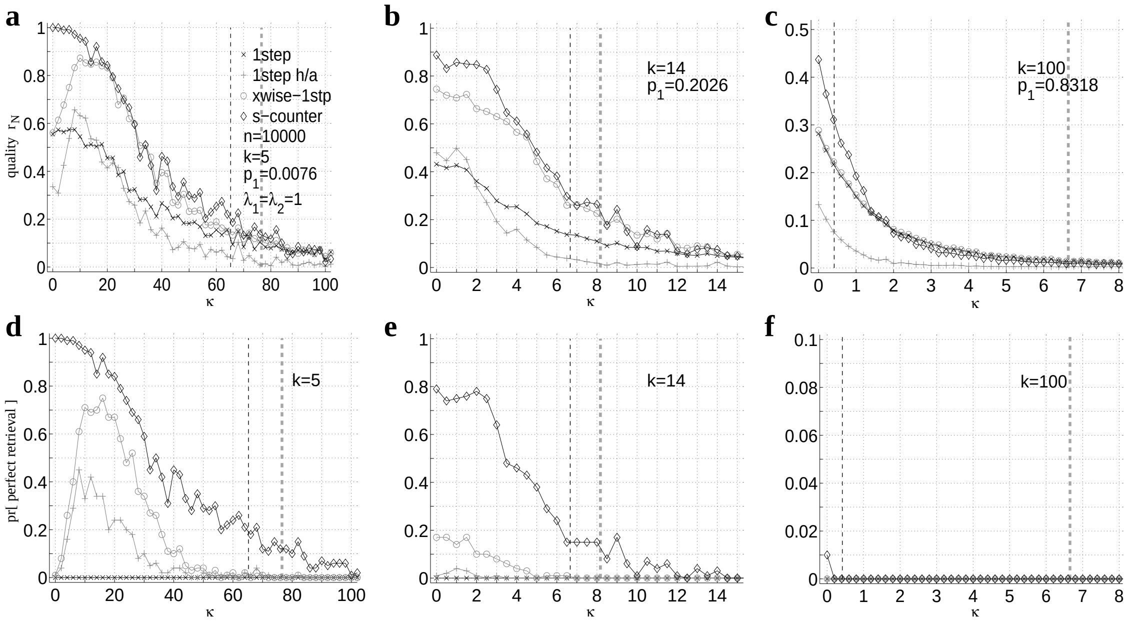


Figure 4.13: Retrieval performance when addressing with a superposition of two complete patterns ($\lambda_1 = \lambda_2 = 1$) for the same matrix and pattern parameters as in Fig. 4.12. Same legends as in Fig. 4.12. **a,d**: Ultra-sparse patterns ($k = 5$, $M = 30516$, $p_1 = 0.0076$) where analysis yields $\kappa < 76.57$ (binomial) and $\kappa < 65.25$ (refined). **b,e**: Classically-sparse patterns ($k = 14 \approx \mathrm{ld}\, n$, $M = 115509$, $p_1 = 0.2026$) where analysis yields $\kappa < 8.17$ (binomial) and $\kappa < 6.67$ (refined). **c,f**: Moderately-sparse patterns ($k = 100$, $M = 17825$, $p_1 = 0.8318$) where analysis yields $\kappa < 6.65$ (binomial) and $\kappa < 0.42$ (refined). The values for maximal κ are by $\lambda_2 = 1$ smaller than in Fig. 4.12 (cf. eqs. 4.64 and 4.48).

of N address patterns we used simply the one out of the corresponding N previously stored target pattern that was most similar to the retrieval result.

Here even for the transinformation measure r_N the spike counter algorithm *s-counter* is significantly better than the other algorithms *1step*, *1step h/a*, and *xwise-1step*. In particular, the classical one-step algorithm *1step* is bad in separating superpositions (cf. Fig. 4.1b): even without noise ($\kappa = 0$) there are practically no perfect retrievals for the *1step* algorithm. In contrast, for ultra-sparse patterns ($k = 5$) performance of the spike counter algorithm *s-counter* seems to be even better than for a single address pattern (compare Fig. 4.13a,d with Fig. 4.12a,d; see also Fig. 4.14a,d; see below for an explanation of this phenomenon).

Another curiosity for ultra-sparse patterns ($k = 5$) can be seen in Fig. 4.13d for algorithms *1step h/a* and *xwise-1step*: The retrieval performance *increases* with increasing noise and a performance maximum occurs at $\kappa \approx 10$. This effect is stable and occurs also for superpositions of more than 2 patterns (data not shown). The most plausible explanation is that a moderate amount of noise serves to break the symmetry between the two addressed patterns and thereby enables pattern separation.

For moderately sparse patterns ($k = 100$) none of the algorithms, not even the spike counter algorithm *s-counter*, performs well. Even without noise ($\kappa = 0$) almost no perfect retrieval occurs. Note that the performance for a single complete address pattern with k additional false one-entries (see $\kappa = 1$ in Fig. 4.12) the situation is much better. Thus this effect can be attributed mainly to a bad pattern separation due to the high memory load $p_1 \approx 0.82 > 0.5$ (see section 4.4.4 on page 69).

In further simulation experiments summarized in Figure 4.14 we have investigated addressing with superpositions of many complete address patterns ($N = 1, 2, 3, 4, 5, 10$) using the same

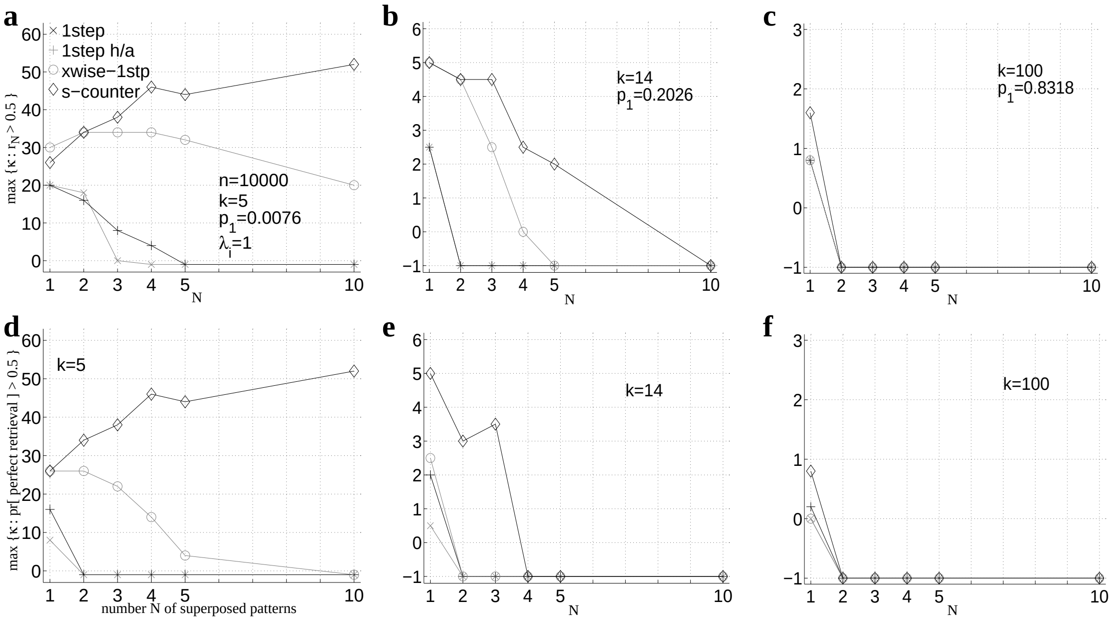


Figure 4.14: Retrieval performance of different algorithms when addressing with a superposition of N complete patterns ($\lambda_1 = \lambda_2 = \ldots = \lambda_N = 1$; $N = 1, 2, 3, 4, 5, 10$) for the same matrix and pattern parameters as in Figs. 4.12 and 4.13. For the algorithms the legends are the same as in Fig. 4.12. Top panels (**a,b,c**) show for given N the maximal κ where the retrieval quality r_N (see eq. B.16) was larger than 0.5. Similarly, the bottom panels (**c,e,f**) show the maximal κ where at least half of the retrievals were perfect. **a,d**: Ultra-sparse patterns ($k = 5$, $M = 30516$, $p_1 = 0.0076$). **b,e**: Classically-sparse patterns ($k = 14 \approx \mathrm{ld}\, n$, $M = 115509$, $p_1 = 0.2026$). **c,f**: Moderately-sparse patterns ($k = 100$, $M = 17825$, $p_1 = 0.8318$). Negative values indicate that even for $\kappa = 0$ (i.e., no false ones) the quality criterion ($r_N > 0.5$ or pr[perfect retrieval]>0.5) was not met. Each plotted value has been estimated from 100 retrievals.

memory matrices as for Figs 4.12 and 4.13. For the two quality measures transinformation (Fig. 4.14a,b,c) and fraction of perfect retrievals (Fig. 4.14d,e,f) the plots show the maximal κ where the measure was at least 0.5. For example, for the transinformation measured for the spike counter algorithm *s-counter* and $k = 5$ (Fig. 4.14a) we have a value of $\max \kappa = 26$ for $N = 1$ because in the corresponding plot (Fig. 4.12a) the maximal κ where the *s-counter* curve exceeded $r_N = 0.5$ was around $\kappa = 26$. Similarly for $N = 2$ we have $\max \kappa \approx 33$ (compare Fig. 4.13a with Fig. 4.14a).

For ultra-sparse ($k = 5$) and classically sparse patterns ($k = 14$) the spike counter algorithm generally performs much better than the other algorithms in particular for many superposed address patterns. However, for high matrix load (as it is the case for moderately sparse patterns with $k = 100$ and $p_1 \approx 0.83$) pattern separation is bad even for the spike counter algorithm (see section 4.4.4 on page 69).

As already mentioned above at first glance it is surprising that the fault tolerance increases with the number N of superposed patterns for ultra sparse patterns ($k = 5$) and the spike counter algorithm *s-counter* (and also for algorithm *xwise*; see Fig. 4.14a). A plausible explanation is the following: First note that for ultra-sparse patterns the first spike decides which pattern is retrieved (this is because the matrix load p_1 is very small for ultra-sparse patterns; cf. section 3.2.1). Then a perfect retrieval occurs (with high probability) as soon as the first spike is correct. This explains the observed phenomenon because the probability that the first spike belongs to an addressed (i.e., correct) neuron is larger for a larger number of addressed neurons and, correspondingly, larger N.

4.5.5 Summary

We have seen that the technical spike counter model can be implemented efficiently both on sequential and parallel architectures. Table 4.1 summarizes retrieval times for sequential and parallel implementations of the classical Willshaw model and the spike counter model with or without matrix compression.

	sequential	parallel
classical	$\sim k \cdot n$	$\sim k$
compressed classical	$\sim \min(p_1, 1-p_1) \cdot k \cdot n$	$\sim \# \cdot k$
spike counter	$\sim k \cdot n$	$\sim k \cdot \mathrm{ld}\, n$
compressed spike counter	$\sim \min(p_1, 1-p_1) \cdot k \cdot n$	$\sim k \cdot \mathrm{ld}\, n$

Table 4.1: Comparison of the (asymptotical) retrieval times t_{ret} for sequential and parallel implementations of different variants of Willshaw associative memory with n neurons, k one-entries per binary pattern vector, and matrix load p_1. # is a parameter for parallel matrix decompression (see text; cf. section 3.5.2); classical: classical Willshaw model (one-step-retrieval) without matrix compression; compressed classical: one-step-retrieval applying optimal matrix compression; spike counter: technical spike counter model without matrix compression; compressed spike counter: technical spike counter model applying optimal matrix compression.

For *sequential* implementations on a digital computer the spike counter model requires asymptotically the same time (up to a constant factor) as the classical Willshaw model, while matrix compression even accelerates the retrieval.

For a *parallel* implementation on a digital computer the spike counter model requires factor ld n more time than the classical Willshaw model. In contrast to the classical Willshaw model, matrix compression causes no additional costs for the parallelized spike counter model. I.e., if one chooses the spike counter model then there is no reason why not also compress the memory matrix.

4.6 Biological implementations of the spike counter model

A direct biological implementation of the more technical model of the spike counters c^A, c^H, c^Σ (see sections 4.3 and 4.5) and the input integration function f_1 (eqs. 4.2 and 4.65) is realized as follows (cf. [86]): Consider the connection scheme of Figure 4.15. It consists of two areas. The addressing area R makes input connections in the addressed area C. For area R we consider only the excitatory population R, since it is known that only excitatory neurons make cortico-cortical connections (see section 2.1.3; cf. [182, 27]). Area C consists of three populations. One excitatory population C, and two inhibitory populations C^S (*separating* inhibition) and C^T (*terminating* inhibition). Note that only population C^S (but not C^T) receives input from the addressing area.

For our base model, we assume the following: Excitatory neurons make specific connections to excitatory as well as to the separating inhibitory population C^S. All other connections are unspecific. This means that beside the auto-associative connection of population C, the connections from R to C and C^S as well as from C to C^S are hetero-associatively learned according to the binary clipped Hebbian learning rule (see section 3.1.1, eq. 3.1). Moreover, the assemblies of C and C^S have the same structure, therefore we call this model TSI (**T**otally **S**pecific separating **I**nhibition) opposed to other (more realistic) model variants introduced below. An important feature of the TSI model is, that connections onto C and C^S are the same. The result is, that in C and C^S always corresponding neuron populations get activated. Thus information about the spike counter c^A is conveyed through the excitatory auto-associative connections of population C, while information about the spike counter c^Σ can be conveyed through the unspecific (e.g. full) connections from population C^S onto C. Note that this property is even independent of many features of the used neuron model (including synaptic and axonal delays). The only requirement is, that excitatory and inhibitory inputs superpose in an approximately subtractive

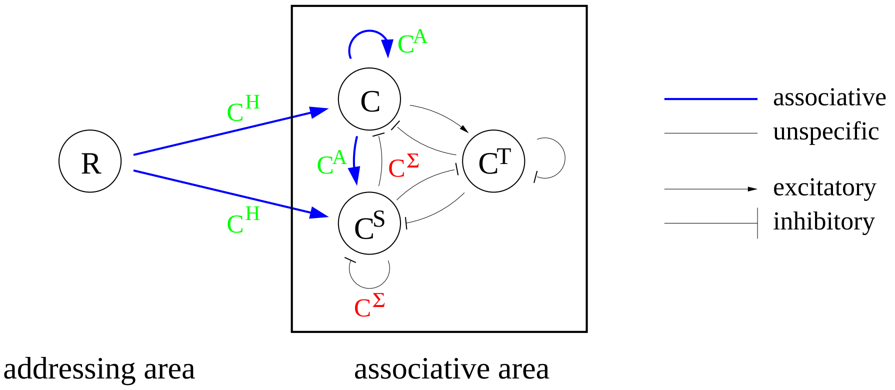


Figure 4.15: Network structure of the spiking associative memory (associative area C). It consists of three neuron populations. Population C is excitatory, while populations C^S and C^T are inhibitory. Populations C and C^S receive extra-areal input from addressing area R, while population C^T exclusively receives input from within area C. Arrows → denote excitation, ⊣ inhibition. Thick arrows correspond to specific associative (i.e., Hebbian learned) synaptic connections between neuron populations, while thin arrows correspond to unspecific (e.g., full) connections. The spike counters c^H, c^A, and c^Σ of the more technical models are implemented analogously by the synaptic connections as indicated (see text for details).

manner on a time scale according to the retrieval length (in our model 4-5msec as we will show below).

Finally we relate this biological model more formally to the spike counter model proposed above. We suggest that excitatory and inhibitory conductances play the role of the spike counters. In our model, excitatory and inhibitory conductances g_{ex} and g_{in} superimpose linearly (eqs. 2.1 and 2.2). Therefore we can write

$$g_{\text{ex}} \approx K \cdot c^H + L \cdot c^A \tag{4.73}$$
$$g_{\text{in}} \approx M \cdot c^\Sigma \tag{4.74}$$

for relative synaptic strengths K, L and M, if the spike counts represent spikes in a short time window (dependent on the decay constants of the synaptic conductances). Below, we will find that a retrieval is done in 4-5msec, and therefore the approximation is justified. Note, that in our model, the synaptic conductance changes have the form of an exponential. Using alpha-functions with prolonged plateaus would even improve this approximation.

Neglecting noise processing and assuming an essentially subtractive superposition of excitatory and inhibitory postsynaptic potentials (EPSPs and IPSPs), we obtain from eq. 2.4 the following approximation.

$$\begin{aligned}
\tau_x \frac{d}{dt}x &= -x + g_{\text{ex}}(E_{\text{ex}} - x) + g_{\text{in}}(E_{\text{in}} - x) \\
&\approx g_{\text{ex}}(E_{\text{ex}} - x) + g_{\text{in}}(E_{\text{in}} - x) \\
&\approx (K \cdot c^H + L \cdot c^A)(E_{\text{ex}} - x) + M \cdot c^\Sigma (E_{\text{in}} - x) \\
&= K(E_{\text{ex}} - x)c^H + L(E_{\text{ex}} - x)\left(c^A - \frac{M(x - E_{\text{in}})}{L(E_{\text{ex}} - x)} c^\Sigma\right)
\end{aligned}$$

But this is essentially the input integration function f_1 of eqs. 4.2 and 4.65

$$\tau_x \frac{d}{dt}x \approx f_1(c^H, c^A, c^\Sigma) = a \cdot c^H + b(c^A - \alpha c^\Sigma) \tag{4.75}$$

with parameters

$$a = \frac{K(E_{\text{ex}} - x)}{\tau_x} \tag{4.76}$$

$$b = \frac{L(E_{\text{ex}} - x)}{\tau_x} \tag{4.77}$$

$$\alpha = \frac{M(x - E_{\text{in}})}{L(E_{\text{ex}} - x)} \tag{4.78}$$

Note that the parameters depend on the membrane potential x. While for biological parameter values, the dependence of a and b may be weak, since x is small in comparison to the excitatory reversal potential E_{ex}, the dependence of α may be much stronger, since the resting potential of neurons ($x = 0$) may be near the inhibitory reversal potential E_{in}. This means that the second requirement for the input integration function f may be fulfilled only for membrane potentials near the firing threshold (see section 4.3 and eq. 4.1). Or the other way, the separation strength α increases when the membrane potential approaches the firing threshold.

Thus we have essentially implemented our technical spike counter model using an excitatory and an inhibitory neuron population where the connectivity for the inhibitory one is a copy of the excitatory one. However, for several reasons there may be a need for a second inhibitory population. Since our model relies on a balanced excitation and inhibition, it is not always guaranteed that the excitation can be controlled. Furthermore, it may even be useful to choose the separation strength to be non-stationary in order to work in different regimes ([130],chapter 12). In an unbalanced regime of weak separation strength, the inhibition produced by C^S may be much too weak to control excitation. Therefore to be sure of uncontrolled spread of excitation there may be a need of further inhibition in a late phase of the retrieval. The idea that inhibition generally lasts longer than excitation is quite common both in physiology and modeling.

The idea is to make the connectivity for the second inhibitory population C^T the same as for neurons of an addressed assembly that get no extra-areal input (e.g. because the address pattern was incomplete). Given that only one assembly gets activated, unspecific (e.g. full) input connectivity from C is sufficient to achieve this. The result is that neurons of C^T fire at the end of the retrieval and guarantee that finally inhibition prevails. This holds if a single assembly gets activated, as required for pattern separation. On the other hand, if a superposition of several assemblies gets activated, then the neurons of C^T fire before the last neurons of the superposed addressed assemblies. Together with the fault detecting properties of C^S as described above (see section 4.3), this breaks down the retrieval and prevents the activation of a superposition of several patterns. Note that it is not necessary to make C^T specific.

Since our base model (TSI) may be not very realistic with respect to the connection scheme, we simulated also some variants successively reducing the assumptions of our strictly symmetric model. For PSI (**P**artially **S**pecific separating **I**nhibition), we stored less assemblies in C^S than in C. The result is that one assembly in C^S controls excitation for several assemblies in C. This may be the case, for example, if exclusively excitatory-to-excitatory connections can be learned, but a certain degree of specificity is achieved through the spatial arrangement of inhibitory neurons. For USI (**UnS**pecific separating **I**nhibition) only one assembly was stored in C^S controlling all the excitatory assemblies. This would happen if connections are learned only for excitatory-to-excitatory connections and we make no special assumptions of the spatial arrangement of the neurons. In NSI (No Separating Inhibition) we inactivated population C^S completely. This model is essential the same as in [199], except that the gradual inhibitory neuron was replaced with the spiking population C^T.

Figure 4.16 demonstrates the typical behavior of our spiking associative memory. For addressing R we activated subsequently two assemblies A_{R1} and A_{R2} hetero-associatively corresponding to the assemblies A_{C1} and A_{C2} stored in area C (Fig. 4.16a). A_{R2} was activated t_{diff} = 2msec after A_{R1}, and the temporal dispersion of activation times of individual neurons was a Gaussian with standard deviation of 1.5 msec. Figure 4.16b shows the activation of the two addressed assemblies in population C. While the first assembly gets activated completely, the second assembly is mostly suppressed with the most activated neurons lying in the overlap with the first assembly. Figure 4.16c shows the activation of the three neuron populations in area C. The populations C and C^S show a symmetric behavior, while population C^T gets activated at the end of the retrieval.

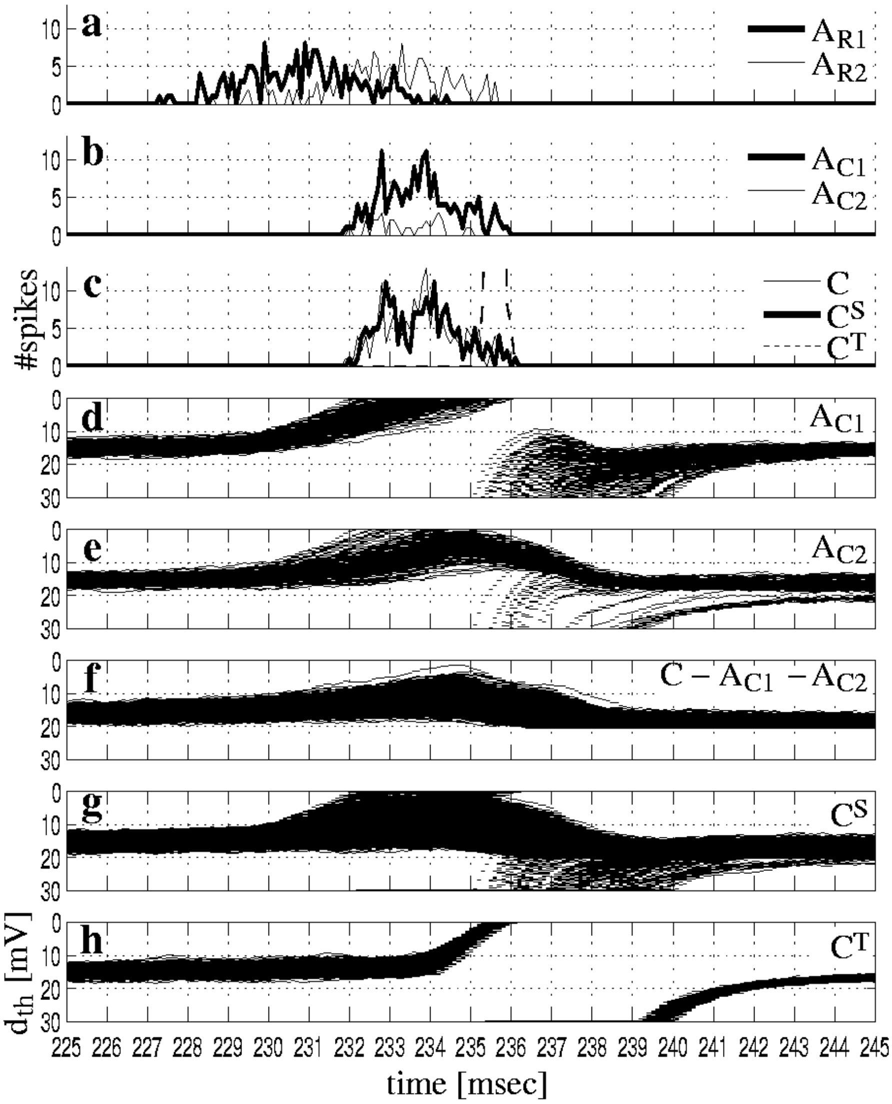


Figure 4.16: Retrieval in the biological model of spiking associative memory when addressed with a superposition of two patterns. **a**: Spike activity in the address area R. Two patterns A_{R1} (thick line) and A_{R2} (thin line) are activated with strong temporal overlap. **b**: Spike activity in population C. Only assembly A_{C1} (thick line) addressed by the earlier pattern A_{R1} gets completely activated. The other addressed assembly A_{C2} (thin line) is almost completely suppressed. **c**: Spike activity in the three populations C, C^S and C^T. Populations C and C^S show a symmetric activation, while C^T is activated at the end of the retrieval. **d-h**: Threshold distance histograms (TDH) over time. For a neuron, threshold distance is the difference between membrane potential and the instantaneous firing threshold (including refractory mechanism and habituation). The histograms are binary, i.e. a black dot is drawn if at least one neuron is in the bin. **d**: TDH for the first addressed assembly A_{C1}. **e**: TDH for A_{C2}. **f**: TDH for the other non-addressed neurons in C. **g**: TDH for population C^S. **h**: TDH for C^T.

A more detailed illustration of the interplay of the different populations is given in Fig. 4.16d-h. Here, histograms of the threshold distance d_{th} are shown over time. The threshold distance for a neuron is defined as the difference of membrane potential and instantaneous firing threshold including refractoriness and habituation (see section 2.4). Note that the histograms plots are binary for the sake of better visibility. Thus a black dot results if at least one neuron is in the bin. Figures 4.16d and e show the histograms for the two addressed assemblies in C, Fig. 4.16f for the other neurons in C not being addressed, and Figs. 4.16g and h show the histograms for the two inhibitory populations C^S and C^T. Omitting population C^S in our model leads generally to the simultaneous activation of both addressed assemblies in C, while omitting population C^T can lead to uncontrolled bursting of the neurons if separation strength is relatively weak (see [86] for more details).

Some results concerning the capability of our models for pattern retrieval and separation are shown in Figure 4.17. TSI1 is a variant of our base model, where the assemblies in C^S were generated independently from those in C. NSI1 is a variant of NSI where the connection strengths were optimized for pattern separation to unrealistic values. We simulated two conditions: Either with only 10 stored assemblies per population (Fig. 4.17a–d), or with high memory loads $p_1 = 0.25$ for the connections from R to C, and $p_1 = 0.40$ for connections within area C (Fig. 4.17e–h).

The plots show the normalized retrieval quality r_N over probability p_{10} of missing one-entries (Fig. 4.17a,e) and probability p_{01} of additional one-entries (Fig. 4.17b,f) in the address pattern when addressing with a single pattern. The normalized retrieval quality is defined as the transinformation between the stored and the retrieved patterns normalized to perfect retrievals (see eq. B.16 in appendix B.3.1). A value $r_N = 1$ therefore means that a complete pattern was retrieved with no faults.

Then we addressed area C using a superposition of two patterns. Fig. 4.17c,g shows the normalized quality of the retrievals, Fig. 4.17d,h the normalized separation s_N over the average time difference t_{diff} of the spike times of the two addressing assemblies in R. If all addressed assemblies get the same activation, we have $s_N = 0$, and we obtain $s_N = 1$ if only neurons of one assembly get activated (see eq. B.20 in appendix B.3.2).

For low memory load and addressing with a single pattern we obtain good retrieval performance for all models (Fig. 4.17a,b). However, the differences between the models with (TSI, TSI1, PSI and USI) and without (NSI, NSI1) separating inhibitory population C^S (see Fig. 4.15) are revealed when addressing with a superposition of two patterns (Fig. 4.17c,d). While the model variants with inactivated population C^S exhibit pattern separation only for relatively high values of t_{diff}, all the variants including population C^S accomplish pattern separation also for small values of t_{diff}. The differences between the two model groups increase, when the memory load is increased (Fig. 4.17e–h). Here, the models without C^S show generally bad results in both cases addressing with one pattern or with a superposition. In contrast the models with C^S still show acceptable results.

Note that depending on the operation mode of connected associative memories, relatively small values of s_N for a single retrieval are sufficient. For instance, in an oscillatory regime with two reverberating associative memories, the separation is improved from one to the next iteration. This behavior is known as *iterative retrieval* in technical associative memories (cf. [160] for the auto-associative and [171] for the hetero-associative case). The simulation results presented in the next chapter 5 suggest indeed a similar function of synchronous oscillations found in the visual system.

4.7 Implications

4.7.1 Summary of the chapter

In this chapter we have extended the classical Willshaw model (chapter 3) by using spiking neurons. Starting from simple considerations about problems of classical one-step retrieval with the separation of overlayed address patterns (section 4.1 and 4.2) we developed the so-called

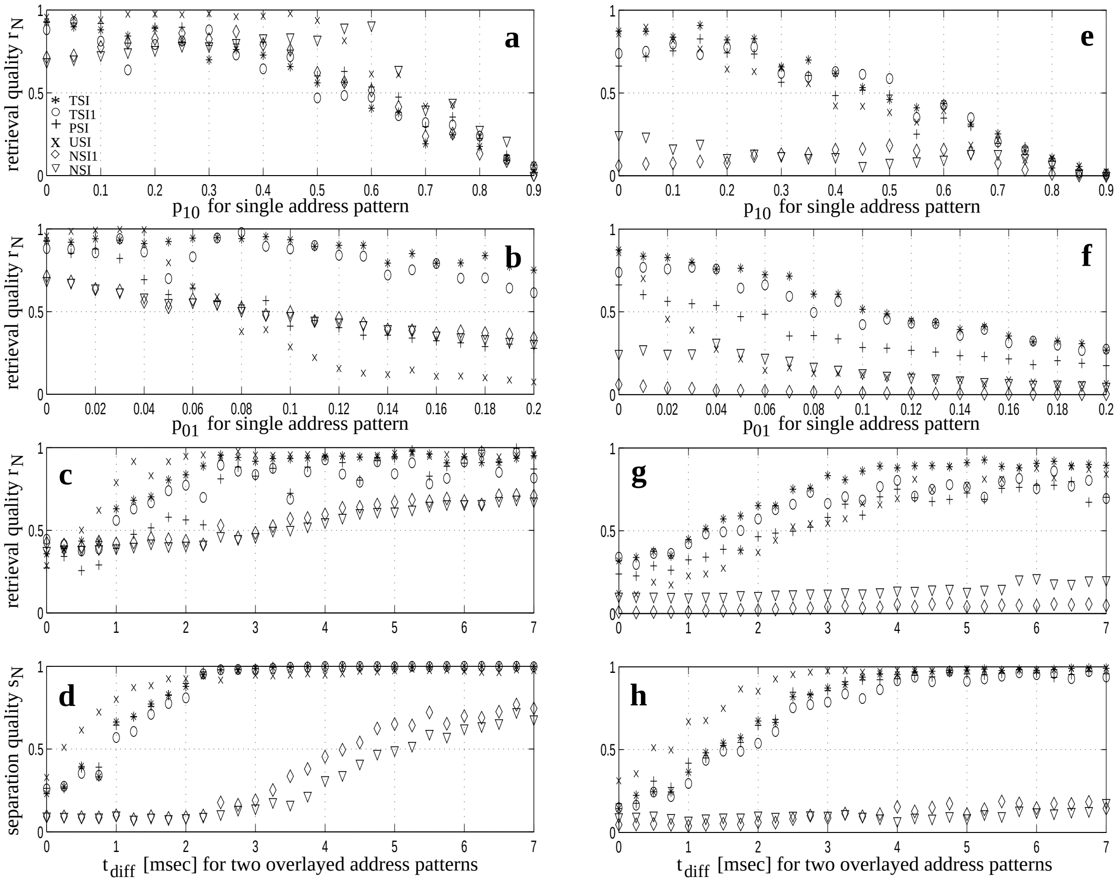


Figure 4.17: Quality of retrieval and pattern separation for the biological spiking associative memory model using 1600 neurons for population C and C^S and 400 neurons for population C^T, and an average pattern size of 160. The following variants of the model (see text) were tested: TSI ($*$), TSI1($\circ$), PSI(+), USI($\times$), NSI(∇) and NSI1($\diamond$). Two different memory conditions were simulated for each variant: Left column (**a-d**) : low memory load (10 stored pattern). Right column (**e-h**): High memory load with intra-areal and inter-areal memory matrixes filled up to 0.4 and 0.25 respectively. **a/e**: Retrieval Quality r_N over probability p_{10} of a missing active element in the single addressing pattern. **b/f**: r_N over probability p_{01} of an additional active element in the single addressing pattern. **c/g**: r_N over average time difference t_{diff} of two address patterns laid over in time. **d/h**: Separation Quality s_N over t_{diff}. For c/d and g/h the same simulations were used respectively. Note the difference in separation performance between the model variants with activated separating inhibition C^S (TSI,TSI1,PSI,USI) and the variants with deactivated C^S (NSI,NSI1).

spike counter model (section 4.3) which implements essentially a time-continuous instantaneous version of the classical Willshaw retrieval strategy (cf. section 3.1.1). The basic idea of the spike counter model is that the *first* spikes during a retrieval can determine the retrieved pattern by immediate auto-associative feedback. This can improve the resistance against noise in the address pattern, and in particular the separation of simultaneously addressed patterns (cf. Fig. 4.2).

In section 4.4 we analyzed the spike counter model with respect to fault tolerance. The analysis for addressing with single patterns compares the potential distributions of the addressed (i.e., correct) neurons to the distribution of the non-addressed (i.e., false) neurons by applying Gaussian approximations (section 4.4.1). For classical one-step retrieval the two Gaussians must be separated completely to obtain perfect retrievals (cf. Fig. 4.1a). In contrast, the spike counter model requires only that the neurons with the *highest* potentials (corresponding to the first spikes) are correct ones, while the two Gaussians are allowed to have significant overlaps (cf. Fig. 4.6). This leads to an improved tolerance against noise in the address pattern. On the one hand the fault tolerance against *missing* one-entries in the address pattern improves only little because the Gaussian of the addressed (correct) neurons collapses to a Dirac impulse in this case, which makes spike counter and one-step retrieval almost equivalent. On the other hand the fault tolerance against *missing* one-entries in the address pattern can be improved massively (see eq. 4.24) at least for a low matrix load $p_1 < 0.5$ corresponding to sparse patterns with logarithmic or sub-logarithmic pattern activity k (cf. section 3.2). In section 4.4.2 it is shown that for ultra-sparse patterns with sub-logarithmic k and $p_1 \to 0$ (where also storage capacity 1 is possible; cf. section 3.2.1) the spike counter model can cope with $O((\ln n)^d)$ false one-entries in the address pattern for any $d > 0$. In section 4.4.3 the analysis is generalized for separation of overlayed patterns where overlaps of the addressed patterns are neglected.

The conducted analysis predicts sometimes implausible high fault tolerance against false ones. In section 4.4.4 the different sources for these misleading results are analyzed, and conditions are derived where the analysis of sections 4.4.1-4.4.3 is valid. One source for improper analytical results is the use of the binomial approximation for the membrane potential distributions (cf. Fig. 4.1). Although we have seen in section 3.6 that the binomial approximation is valid asymptotically for large n and sub-linear address pattern activity z (=k for perfect address pattern), we have also seen that the binomial approximation can be very bad for finite n and in particular for large z where the address pattern contains many false one-entries (cf. Fig. 3.5). In particular it turned out that the binomial approximation can underestimate the true variance of the potentials massively (see Fig. 3.5c,d; cf. eqs. 3.122 and 3.123). In section 4.4.5 we have therefore repeated the analysis of section 4.4.1 using Gaussian approximations with the correct variances (eq. 3.122; cf. Figs. 4.7 and 4.6).

In section 4.5 we have discussed efficient sequential and parallel implementations of the spike counter model (cf. Table 4.1). It turned out that a sequential implementation of the spike counter model is asymptotically (up to a constant factor) as efficient as classical one-step retrieval ($\sim kn$ steps per retrieval; see section 4.5.2; cf. section 3.4). However, the suggested spike counter algorithm cannot be parallelized optimally. In section 4.5.3 we have seen that a parallel implementation requires $\sim k \mathrm{ld}\, n$ steps per retrieval while a parallel implementation of classical one-step retrieval requires only $\sim k$ steps (cf. section 3.5.2). On the other hand this is still quite efficient, and implementing in addition optimal matrix compression (see section 3.2) bears asymptotically no additional costs.

In section 4.6 we have shown that the principles of the technical spike counter model can also be implemented in a biologically realistic architecture. For this purpose the spike counters (see section 4.3) have been mapped to the excitatory and inhibitory synaptic conductances of a more realistic neuron model (see section 2.4). In the subsequent chapter 5 we incorporate the described biological model into a larger model of interacting visual cortical areas in order to investigate spike synchronization, scene segmentation, and binding of distributed representations.

4.7.2 Why spikes?

Why do neurons emit spikes? It is well established that real neurons can emit a spike when the depolarization of the membrane potential at the axon hillock exceeds a certain threshold. The duration of a spike lies in the range of one millisecond, and it is generally believed that spikes are the most important carrier of fast information transmission in biological systems. However, the significance of the precise spike times is controversial.

According to one view-point the only function of spikes is to transmit the information contained in the membrane potential of the neuron over long distances. Direct electrotonic coupling of connected structures would result in heavy loss of information for longer distances. For example, the so-called space constant of dendrites is at most a few millimeters. This means that a signal can be transmitted passively only over a few millimeters, and the signal is weakened gradually to zero all along its way. In contrast a spike can be transmitted along the axon over arbitrary long distances without being weakened. The price to be paid is that a spike is a quite stereotypic almost digital or binary event. In fact a single spike cannot contain significant information about the membrane potential since its shape is almost independent of the degree of depolarization at the axon hillock. Thus information about the membrane potential could be obtained by counting the spikes of the neuron over a longer time interval, i.e. by evaluating its *spike rate*.

According to another view-point the precise timing of a spike on the millisecond-time-scale would contain additional information if related to other spikes of the same and other neurons. For example, the stronger a neuron is excited, the earlier it will emit a spike. In fact the model of a spiking associative memory developed in this chapter makes use of this property of spiking neurons: The spike time contains information about the strength of excitation relative to other neurons. Other neurobiologists and theoreticians claim that even more subtle aspects of the precise spike timing of a neuron population contains significant information about the functional state of the organism. E.g., it has been suggested that spike synchronization within a time window of perhaps 1-5msec may be used for global binding of distributed feature representations (e.g., [41, 61, 168]; see also section 2.2.4). Or it has been suggested that subtle complex spatio-temporal patterns, so-called synfire-chains [2, 3], may also be functionally relevant [197].

In the following we provide a (probably incomplete) list of possible answers why evolution might have invented spikes. Some of the arguments are discussed in more detail below. The rather speculative arguments are signed by a question mark.

C1: Implications of cable theory: Distance-dependent attenuation of electrotonic signals [145], while spikes can be transmitted without loss of information over arbitrary long distances.

T1: High temporal resolution? This corresponds also to short reaction times: After stimulus onset the first spikes at a given cortical location are most relevant [183].

T2: (Global) binding by spike synchronization [191, 192, 41, 61, 168]?

T3: Functional relevance of complex spatio-temporal spike patterns (presumably generated by synfire chains) [2, 3, 5, 197]?

S1: Synaptic repertoire: Synaptic transmission of a spike can elicit numerous effects on the postsynaptic site depending on the receptor distribution: E.g., one spike can at the same time cause modulatory signals and excitatory signals on different time scales (e.g. NMDA vs. AMPA; cf. [38]).

S2: Synaptic plasticity: Detection of coincidence and causes?; Asymmetric learning rules [112, 15, 1, 173, 53].

A1: Separation of overlayed patterns or assemblies (distributed representations): Linear models are not able to separate superpositions.

A2: Attractor fusion: Flat or quasi-linear activation functions of the neurons lead to bad separation (or even fusion) of attractors [70].

A3: Sparseness: Sparse patterns are only stable for steep activation functions of the neurons.

A4: Willshaw distribution: The variance of non-addressed neurons' potentials is quadratic in the number of active input units (see section 3.6.3; cf. Fig. 3.5; cf. eq. 3.122): Thus for noisy membrane potentials assemblies would not be stable for non-steep activation functions of the neurons.

Arguments C1 and T1-T3 have been discussed briefly above. In particular the idea that the first few spikes can determine the functional state of an organism ([183]; see argument T1) fits perfectly with the role of spikes in our model of spiking associative memory (see section 4.3). In chapters 5 and 6 we will develop and discuss a model of reciprocally connected visual cortical areas in order to further investigate the hypothesis of global binding by spike synchronization corresponding to argument T2. Arguments S1 and S2 are experimentally relatively well verified, and at least the truth of argument S2 implies also the truth of argument T1 (and perhaps also T3).

Most closely related to the work described in this chapter are the arguments A1-A3 which presuppose the existence and usefulness of cell assemblies or discrete activity states. Argument A1 states for example that linear associative memory cannot separate superpositions. If the output for addressing with input pattern U^1 is $V^1 := f(U^1)$ and similarly $V^2 := f(U^2)$ for another address pattern U^2, then the output for the superposition $U^1 + U^2$ of the two address patterns will be $f(U^1) + f(U^2)$ per definition of the linearity of f. It is argued that at least it should be *possible* to segment a superposition into its components (see section 4.1). Indeed at least at higher processing stages the brain also processes separately the components of superpositions. For example, if the two eyes see different images the subject does not see a superposition of the two images. Instead one observes *binocular rivalry* where the subject sees the image from either the left or right eye [18], similar to Fig. 2.6 where we also can see either the young girl or the old woman but not a superposition of both (cf. also [46]). In experiments with monkeys this rivalry is also reflected by neural activity [109, 166]. This evidence that the brain requires segmentation of superpositions into its components rules out linear processing, but gives us not yet a clear hint why spikes are useful. This is because also the spike rate of a typical cortical neuron depends non-linearly (usually sigmoidal) on its membrane potential. On the other hand the activation function of many cortical neurons are linear (or almost linear) within a large interval of relevant membrane potentials. This quasi-linearity may cause attractor fusion [70] as stated by argument A2. In contrast, spikes are the most extreme form of a non-linearity (all or nothing) which could be emulated by gradual rate-coded neurons only if the activation function would be very steep (e.g., as it is the case for the Heaviside function). Actually one can argue that networks of spiking neurons and networks of gradual neurons are equivalent (at least with respect to the results presented in this work) if the gradual neurons have a very steep activation function (cf. [70]).

A strong argument for the importance of spikes (or a very steep activation function) is argument A3 stating that rate coding (with a rather flat quasi-linear activation function) would contradict sparse neural assemblies as postulated for example by the Willshaw model. Consider the following example for a population of n neurons and an assembly size of k, where $k \ll n$ for sparse assemblies/patterns. Let us assume that the membrane potentials x of the neurons can take values from the interval $[0;1]$, and that $f(x)$ is the sigmoidal activation function (for example with $f(0) = 0$ and $f(1) = 1$). Let us assume that all k neurons of one assembly are maximally active ($x = 1$), and all other neuron have noisy potentials ν (cf. Fig. 4.1a). If we require the noise-to-signal ratio to be smaller than δ we can write

$$\frac{\text{noise}}{\text{signal}} := \frac{(n-k)p_1 f(\nu)}{kf(1)} \leq \delta \quad (!) \tag{4.79}$$

$$\Leftrightarrow \quad \frac{k}{n-k} \geq \frac{p_1}{\delta} \cdot \frac{f(\nu)}{f(1)} \quad (!). \tag{4.80}$$

This means that for large $f(\nu)$ sparse patterns with small $k/(n-k)$ are not possible for small δ, because then the very few (k) signal inputs would be hidden in the vast amount of many ($p_1(n-k)$) noisy inputs. Obviously, the optimal f would be $f(x) = 0$ for x below a threshold Θ and $f(x) = 1$ (for example) otherwise.

The impact of argument A3 can be understood only in connection with argument A4: In section 3.6 we have seen that the commonly used binomial approximation of the true Willshaw distribution of neuron potentials can be very bad (see Fig. 3.5). In particular the binomial approximation can underestimate massively the variance of the neuron potentials (see section 3.6.3). The true variance grows *quadratic* in the number z of active input elements while the binomial approximation suggests only a *linear* growth (cf. eqs. 3.122 and 3.123; cf. Fig. 3.5c,d). Actually the quadratic growing variance excludes non-sparse assemblies (cf. section 3.2.3 and appendix B.4.3). On the other hand the following self-consistency condition is essential for the stability of an assembly: If only the neurons of the considered assembly are active then the potentials caused by auto-associative feedback must be such that the assembly neurons have larger potentials than the other neurons (e.g., cf. Fig. 4.1a). Now consider gradual neurons with a non-steep activation function and noisy potentials ν of the non-assembly neurons (similarly as argued for A3). Then we have essentially the worst case $z = n$ since all neurons deliver input to other neurons (although the non-assembly neurons only with strength $\approx f(\nu)$ instead of $f(1)$). Since the potential variance (which limits the possible number of assemblies and the storage capacity) grows quadratic in z its largest part will origin from the noisy membrane potential fluctuations of the non-assembly neurons.

Although the analysis of arguments A3 and A4 is quite informal and certainly has to be improved in future work the following conclusions can be drawn: An important function of spikes at least in the context of our model is to *decouple the neurons from noisy membrane potential fluctuations* of the other neurons. It is suggested that in this sense spiking neurons and gradual neurons with a very steep activation function are equivalent.

Chapter 5

Model of visual cortical areas

One of the most fundamental and central questions in systemic neuroscience and brain research is the question of representation of objects or thoughts in the cortex of sufficiently complex animals like monkeys or men. Probably the most reasonable idea on this issue goes back to the psychologist Donald Hebb. Concepts or thoughts are represented in terms of the coincident activation of groups of neurons called *cell-assemblies* ([64]; see section 2.2.2). One immediate argument against this idea which has been put forward many times is the *superposition problem* (cf. section 2.2.3) : if two or more assemblies are activated at the same time, how can they be segmented into the individual assemblies? The answer to this problem is provided by the strong mutual excitation between the neurons belonging to the same assembly, probably acquired by auto-associative Hebbian learning. This has been worked out in several theoretical papers [204, 26, 129, 130, 133, 171] which essentially show that by controlling the total activity within a certain region of the cortex, for example by unspecific inhibition, the superposition problem, i.e. the activation of two or more assemblies at the same time, can be avoided (see section 2.3 and chapters 3 and 4). Furthermore this can be achieved even when a very large number of assemblies (sparse activation patterns) are stored by auto-association.

But is this really a solution? May it not be necessary or at least useful in some situations to activate two or more assemblies at the same time in the same local cortical area and not to confuse them? In a visual scene, for example, there may be several objects present close to each other at the same time: If each of them is represented by an assembly that contains a subassembly in a visual area, and if the whole scene is also represented by an assembly, then these visual sub-assemblies should all be part of the "scene assembly" and therefore be activated at the same time in the same local region of visual cortex. This problem becomes more acute in higher visual areas where the assemblies corresponding to the different objects cannot be assumed to be spatially separated in the cortex due to a topographical visual representation with a sufficiently large magnification factor as in V1 or V2. But even in V1 or V2 neighboring neurons may be activated by different objects, where they are close together or even partially occluding each other. This particular version of the superposition problem has suggested a closer look on the timescale on which coincidence of activity is interpreted. It could be that there is a succession of coincident activity moving from one assembly to the next within a few milliseconds (i.e. on the timescale of single spikes), and also coincident activity of the whole set on the timescale of 100msec or of increased firing rates. This is the strong version of the temporal correlation hypothesis (TCH; see section 2.2.4) [192] which was put forward as a solution to the superposition problem (also called the 'binding problem').

Actually, even a longer time window of up to three seconds [139] may be involved, for example, when we look at a complex picture, or obtain an impression of the room we have just entered, which may require several fixations and involve long-term memorized knowledge. This is a quite common situation, where from the point of view of assembly theory [26, 130] it may be reasonable to assume that a global assembly representing the situation as a whole is activated in the cortex which contains several subassemblies representing several individual objects. These subassemblies are probably only partially activated and in particular it is unlikely that several of them are activated at the same time (narrow time-window) in the same place (local cortical area, within 1-2mm of cortex), because that would violate the constraints on the total local activation. Using threshold control (or eye-movements) we can focus our attention (or our gaze) on each of these objects in turn and activate the corresponding assembly more completely, thereby reducing the activation of other assemblies representing other objects that are part of the same global situation.

The TCH was experimentally supported by the observation of high-frequency gamma (40-60 Hz) oscillations in the visual cortex apparently reflecting global stimulus properties [40, 168]. Evidence for this oscillatory activity was found in EEG or local field potential recordings and also in single-unit auto- or cross-correlograms. While the general idea of interpreting coincidence in assemblies on the timescale of milliseconds or spikes has been pursued by many if not most researchers elaborating the Hebbian assembly concept in the temporal domain [2, 192, 133, 150, 6, 201], the special use of fine-timing made in the context of the visual 'binding problem', also related to the idea of 'phase-coding' (activating different assemblies at different phases of a single underlying oscillation), has been subject to considerable controversies.

In this chapter it is shown that the original Hebbian ideas and the corresponding models based on associative memory almost automatically and naturally solve the "binding problem" by temporal correlation, when they are implemented in biologically realistic networks of spiking neurons. Global cortical assemblies are formed automatically when one assumes Hebbian synapses both in the long-range cortico-cortical connections between different areas and in the excitatory short-range connections within areas. Since most of the cortico-cortical connections appear to be bidirectional, a good starting point is to analyze two bidirectionally connected local areas. Here we essentially interpret our two cortical areas as two local spots of two visual areas, a more peripheral and a more central one, with a high and low spatial magnification factor, respectively. For our neuron-model, we use a spike generation mechanism with a dynamic threshold quite similar to common leaky integrate-and-fire models (see section 2.4). In the following first we describe the model in more detail, and then the simulations and its relation to neurophysiological experiments are summarized.

5.1 Modeling

For simulations we chose a one-point spiking neuron model including refractoriness and habituation as described in section 2.4. The network consists of three areas (R, P and C) each composed of several neuron populations (Figure 5.1). Details of the neuron and network model are also described in [89, 91, 92].

In the retinal area R input patterns corresponding to stimulus objects in the visual field are represented in a 100×100 bitmap. The two cortical areas P (primary visual cortex) and C (central associative area) are both modeled with one excitatory (P and C) and two inhibitory neuron populations (P^S, P^T, C^S, and C^T), where only one of them receives extra-areal input (P^S and C^S). Only the excitatory connections are specific (with respect to orientations in area P, and with respect to the pattern representations in area C) according to Hebbian learning. This is essentially the architecture of the biological version of spiking associative memory described in section 4.6 (cf. Fig. 4.15). The modeling was motivated by the requirement of an efficient threshold control and the separation of distributed representations, especially for the associative area C, and is also biologically plausible ([27]; see sections 2.1.3 and 2.3.3).

5.1.1 Model of the primary visual cortex

Each neuron population in area P has size 100×100 (cf. Fig. 5.1). Connections from R and inside P are modeled corresponding to the subsystem of orientation selective columns in the primary visual cortex (see Fig. 5.2).

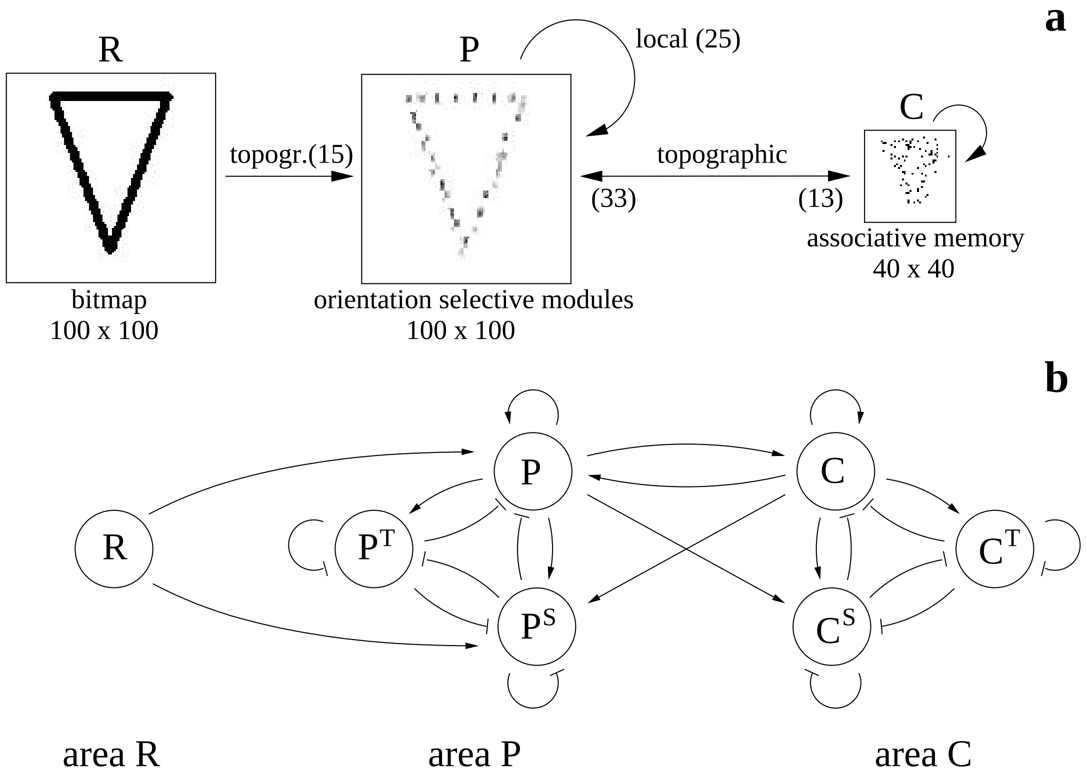


Figure 5.1: Overview of the network model. **a**: Global connection scheme and the representations of a triangle stimulus in the different areas. Retinal area R is simply a bitmap. Primary visual cortical area P is modeled according to the orientation modules of primary visual cortex. Central visual area C represents shapes and is modeled as spiking associative memory (see section 4.6). Numbers in brackets correspond to the size of efferent kernels of individual neurons, e.g. a P-neuron projects to a patch in C with diameter 13 neurons. **b**: Individual neuron populations of the three areas and excitatory (indicated by arrows) and inhibitory connections between them. The structure of areas P and C is similar to the structure of spiking associative memory shown in Fig. 4.15.

The internal excitatory connections from P to populations P and P^S couple specifically neurons which have similar orientation preferences and additionally are near neighbored or collinearly aligned (for details see [91]). This is as expected from Hebbian correlation learning during the presence of stimuli rich of contours, and results in a patchy representation of stimulus objects (Fig. 5.1a): At a certain location in P, only the neurons with orientation preferences best matching the pattern in R get strongly activated, while neighboring cells are suppressed by recurrent inhibition. We have also incorporated in our model temporally and spatially correlated noise as ubiquitously found in experiments (e.g. [190]). This noise, for instance, prevents artificial long-range correlations of spike activity. Due to computational limitations we replaced the two inhibitory populations P^S and P^T in some of the simulations by one gradual inhibitory population.

5.1.2 Reciprocal connection to a central cortical area

Area C (central visual area) is modeled as a fully connected auto-associative network according to Hebbian learning of topographic random representations of stimulus objects (see [89] for details). Populations C and C^S have size 40×40, while population C^T has only size 20×20 (cf. Fig. 5.1b). As can be seen, for example, from the triangle representation in area C (see Fig. 5.1a; cf. Fig. 5.3a),

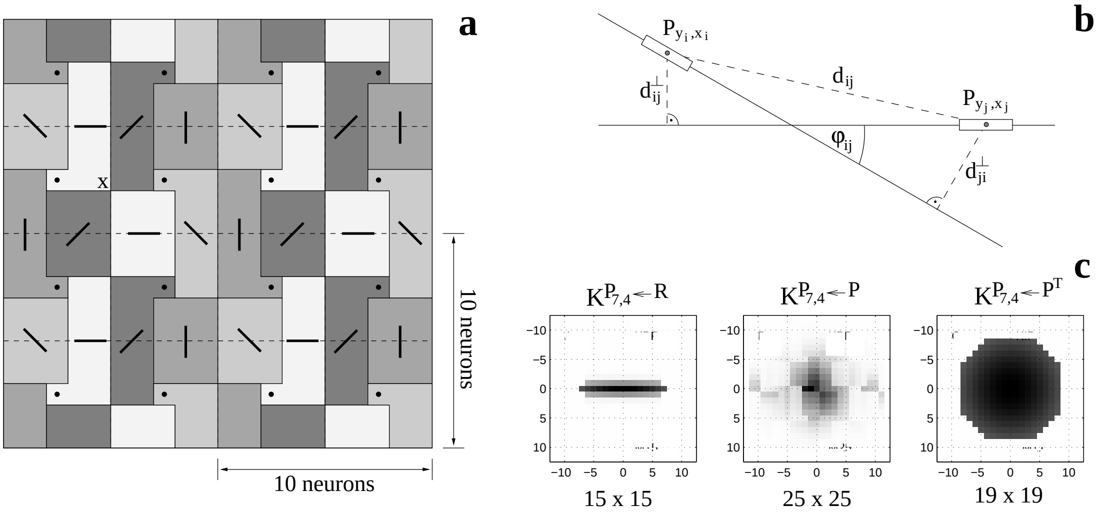


Figure 5.2: Modeling of area P. **a**: Arrangement of orientation selective columns in area P. Only a 20 × 20 clip is shown, whereas area P is composed of 100 identical blocks of size 10 × 10. The *black points* mark axes around which orientation preference rotates for 180°. The *cross* (x) denotes the position of the neurons for which some afferent kernels are shown in (c). **b**: Illustration of the variables determining the strength of the synapse between two P-neurons at positions i and j. Synaptic strength depends mainly on the distance d_{ij} between the neurons, the difference Φ_{ij} in their orientation preference, and the degree of collinearity as measured from $d^{\perp}_{ij}$ and $d^{\perp}_{ji}$ (see text for details). **c**: Plots of some afferent kernels of the neurons at the location corresponding to the *cross* (x) shown in (a) at row seven, column four. For the excitatory neuron, afferents from R (left) and from P (middle) are shown, while for the inhibitory neuron (from P^T) the plot depicts afferents from P (right). Numbers correspond to kernel sizes.

the original stimulus shape is only rawly preserved.

Areas P and C are reciprocally connected by a topographic hetero-association according to Hebbian learning of corresponding stimulus representations in P and C (Fig. 5.3; see [91] for further details).

We use simple shapes as test stimuli, e.g. a triangle, a rectangle or an ellipse (Fig. 5.3a). In different simulations we present single stimuli to focus on the general behavior of the network (e.g., the appearance of oscillations and distance dependent synchronization), or superpositions of several stimuli to investigate competition and segmentation properties of the model. To investigate more global model properties we simulated also a larger variant of the model where areas P and C had sizes 30 × 500 and 15 × 250 and also area C was organized topographically [92].

5.2 Interaction between two cortical areas

5.2.1 Single stimulus

In order to investigate the general behavior of area P, we first simulated the network *without any feedback* from C using the triangle stimulus (see left panels of Figs. 5.4 and 5.5): Activity in both areas then is oscillatory due to the interaction of excitatory and inhibitory pools of neurons. In the primary area P one obtains activated patches along the contour of the stimulus shape (Fig. 5.4c; cf. Figs. 5.1a and 5.3a). Correlation analysis of multi-unit spike activity from different such patches reveals that only activity from neighbored patches is synchronized (modulated correlograms with central peaks) while activity from distant patches is uncorrelated (flat correlograms, cf.Fig.5.5a,c).

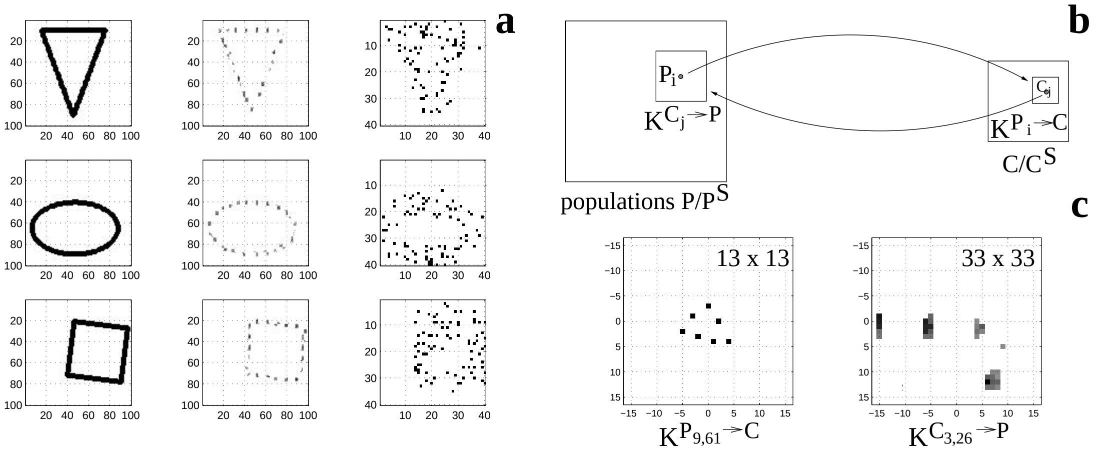


Figure 5.3: Modeling of area C. **a**: Representations of three test stimuli in the three areas R (*first column*), P (*second column*) and C (*third column*). See text for a description of the algorithm generating the representations in C. **b**: Reciprocal connection scheme of areas P and C. Neuron P_i projects to a patch in C/C^S according to efferent kernel $K^{P_i \to C}$. Vice versa, neuron C_j projects back to a patch in P according to efferent kernel $K^{C_j \to P}$. **c**: Plots of efferent kernels of neurons $P_{9,61}$ (*left*) and $C_{3,26}$ (*right*) at roughly corresponding positions in P and C. Compare with the triangle representations in P and C (part (a), *first row*). Numbers in the right upper corners correspond to kernel sizes.

The situation changes if the *complete model* is simulated including the feedback from area C to area P (see right panels of Figs. 5.4 and 5.5). Then also correlograms for activity from distant patches show significant central peaks indicating an enlarged synchronization range (Fig.5.5b,d).

In summary the simulations show that spike synchronization (on a time scale of milliseconds) under realistic noise conditions is only possible for neurons in either directly (neighbored in area P) or indirectly (by divergent feedback from area C) connected cortical regions. This is consistent with neurophysiological results where synchronization seems to be restricted to synaptically connected regions (some millimeters within visual cortex, cf. [40]), but it obviously contradicts the strong version of the TCH: This states that global assemblies of neurons should be synchronized on the time scale of milliseconds despite the lack of direct connections between parts of the assembly. However, as we will see later, global synchronization is indeed possible on a larger time scale of tens to hundreds of milliseconds.

5.2.2 Multiple stimuli

According to the TCH synchronous fast (gamma) oscillations were suggested to solve the binding problem. In this subsection we test our model using a superposition of three test stimuli, a triangle, an ellipse, and a rectangle (cf. Fig. 5.3a). Here the representation of each stimulus in P is a certain set of orientation selective patches (see Fig. 5.6c), and the binding problem is to coordinate activity of patches that belong to one representation in P as well as corresponding representations in P and C. Previous models solved this kind of binding problem for example by phase coding (e.g. [150]). That means, assemblies representing different stimuli are activated during different phases of *one* gamma period. However, there exists little experimental evidence for this kind of coding. Rather, experiments investigating the representations of several simultaneously presented stimuli indicate *uncorrelated* activity between assemblies for the different stimuli corresponding with flat correlograms (e.g., [168]; cf. Fig. 2.8). In case of a phase-coding mechanism the correlograms should be modulated with certain shifts of the central peaks.

Figure 5.6a,b shows simultaneous recordings of spike activity from neurons in P and C. Multi-

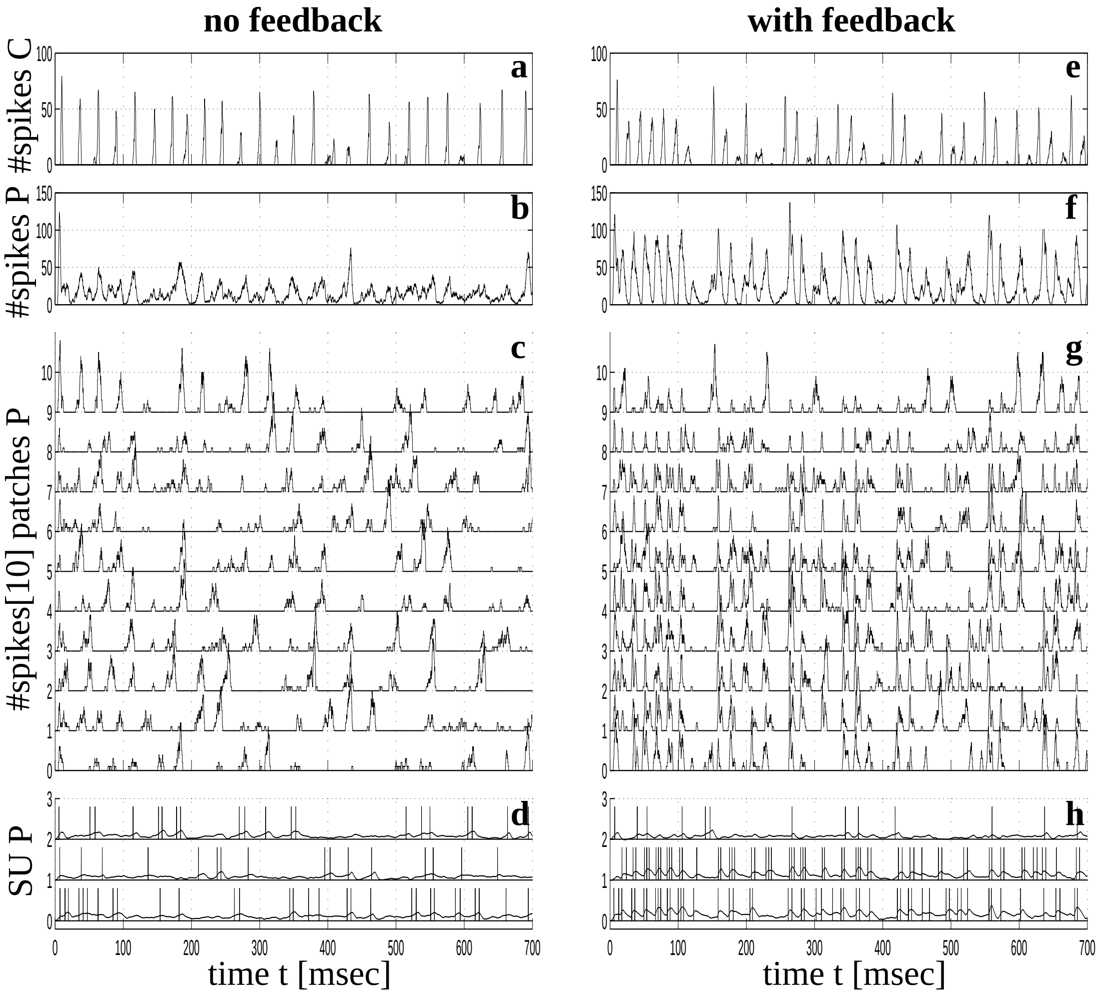


Figure 5.4: Spike trains for single stimulus condition. *Left panels* (**a-d**) correspond to the model with inactivated feedback from area C, *right panels* (**e-h**) correspond to the complete model. **a/e**: Summed spike activity in population C. **b/f**: Summed spike activity in population P. **c/g**: Summed spike records of 10 different activated patches representing the triangle stimulus in population P Ordinate unit corresponds to 10 spikes. **d/h**: Potentials and spikes of single units taken from three activated patches of P. Membrane potential ordinate unit corresponds to 100mV.

unit spike activity is depicted separately for the different (sub)assemblies representing the three stimuli (triangle, ellipse and rectangle). The recordings of one assembly in C (Fig. 5.6a, e.g. the bottom row for the rectangle assembly) show periods of fast oscillatory activity lasting for a few hundred milliseconds alternating with longer periods of essential silence. Comparing the three recordings from C, one observes that only one assembly is in the fast activity state at a time. This results from the structure of the associative memory in area C, which prefers separated activation of the three addressed assemblies (see section 4.6; cf. [89]).

Multi-unit activity of patches belonging to the three assemblies in area P reveals similar results (Fig. 5.6c) as in area C. The recordings show periods of fast and precise (little phase jitter) oscillatory activity (*fast state*) lasting again for a few hundred milliseconds alternating with periods of relatively slow and unordered activity (*slow state*). When looking at different patches of one assembly one observes that activity is synchronized only during the fast state, but asynchronous during the slow state. Moreover, corresponding assemblies in P and C are simultaneously either

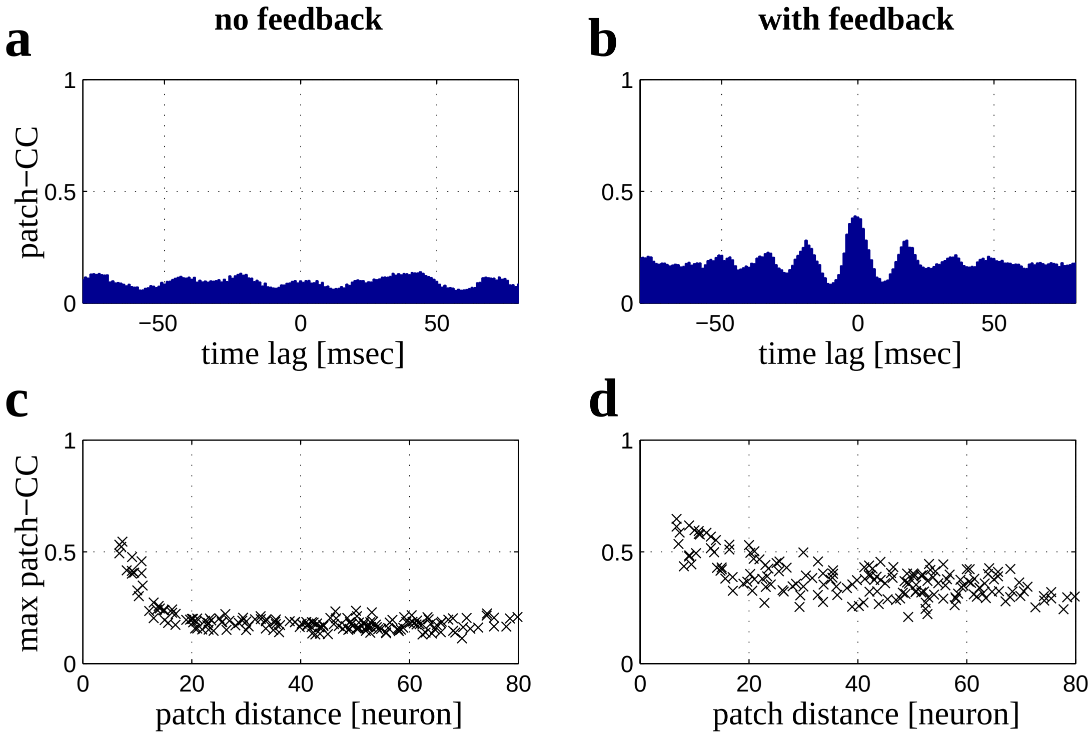


Figure 5.5: Analysis of spike activity from area P when stimulating with a triangle (cf. Figs. 5.4c,g and 5.1a). Plots **a, c** correspond to the model with inactivated feedback from area C, while plots **b, d** correspond to the complete model. **a, b**: Cross correlograms for multi-unit activity from two (distant) activated patches in area P. **c, d**: Maximal correlations (maximized over time lag) vs patch distance for pairs of activated patches in P. Without feedback significant (central) peaks occur only for nearby patch pairs, while for the complete model also distant patch pairs are synchronized.

in the fast or in the slow state. Comparing with the single stimulus conditions, one finds that the slow state is very similar to the results without feedback from the higher area, and that the fast state is very similar to the results of the complete model (cf. Fig. 5.4).

Mean firing rates show a superposition of the assemblies representing the three stimuli (Fig. 5.7a,b). In contrast correlation analysis reveals that spike activity *within* one assembly is synchronized on a timescale of milliseconds (modulated correlograms with central peaks, see Fig. 5.7c for the triangle assembly in area P), while spikes of *different* assemblies are uncorrelated (flat correlograms, see Fig. 5.7d for the cross correlation of triangle and ellipse assemblies in P). This is consistent with findings of specifically modulated and unmodulated correlograms in neurophysiological experiments described in section 2.2.4 using multiple stimulus objects (see Fig. 2.8).

Cross correlations for activities of corresponding assemblies in P and C show modulations with the central peak shifted for about half a period (Fig. 5.7e for the triangle assembly), which occurs similarly for a single stimulus in the complete model (data not shown, cf. [91]). However, fast oscillatory activity in different visual areas is consistently reported to be synchronized with *zero* phase lag. In section 5.3.2 we will discuss this apparent inconsistency in more detail.

Figure 5.7f shows the cross correlations for two distant triangle patches in area P (patches as in Fig. 5.5a,b), but on a larger time scale. While the correlations on the finer time scale looked very similar to Fig. 5.5b, here the correlogram shows one broad central peak indicating non-oscillatory synchronization on a larger time scale. In our simulations these broad peaks result from the switching between two activity states as described above, and they occur centered on

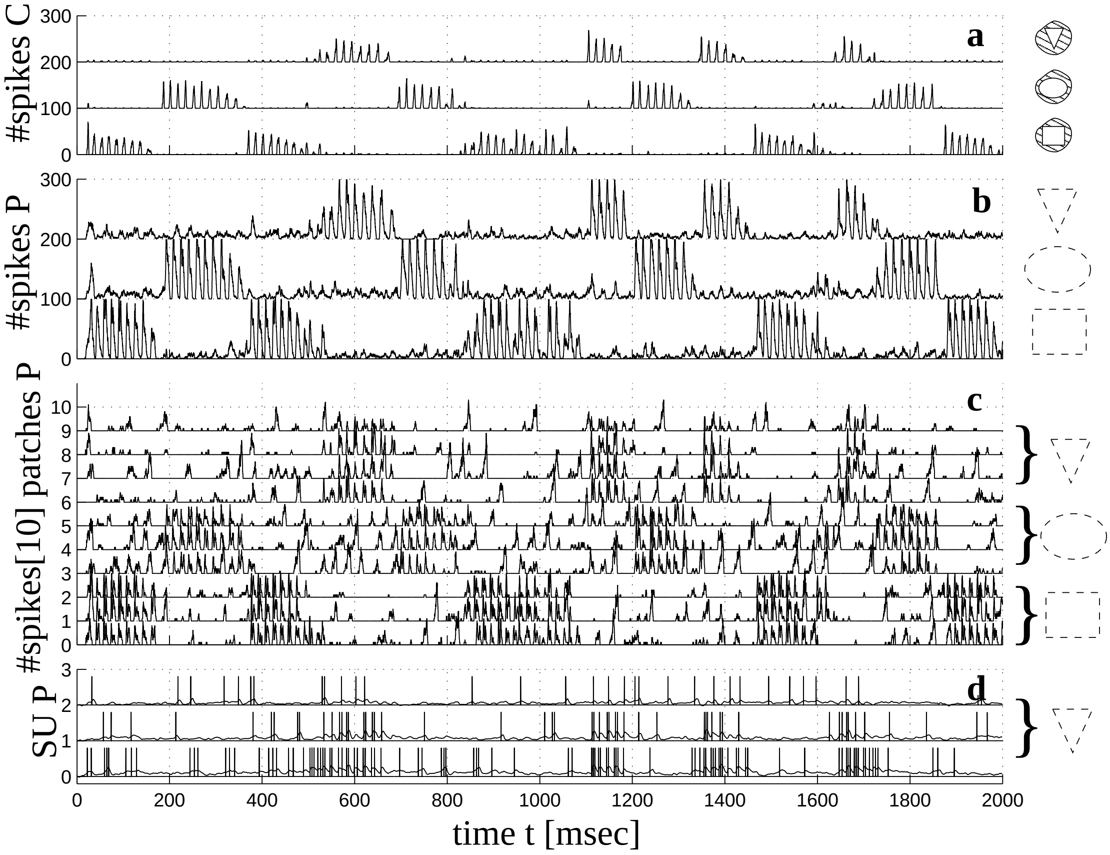


Figure 5.6: Spike trains for a superposition of three stimuli in the complete model. **a**: Summed spike activity in population C for the triangle (top row), ellipse (middle row) and rectangle assembly (bottom row). **b**: Summed spike activity in population P for the triangle (top row), ellipse (middle row) and rectangle assembly (bottom row). **c**: Summed spike records of 10 different activated patches representing the triangle (rows 6–9), the ellipse (rows 3–5) and the rectangle stimulus (rows 0-2) in population P. Ordinate unit corresponds to 10 spikes. **d**: Potentials and spikes of single units taken from three activated patches of P representing the triangle stimulus (same neurons as in Fig. 5.4d,h). For membrane potentials ordinate unit corresponds to 100mV.

the time origin even for activity from different areas. In section 5.3.3 these results are further discussed and related to findings of broad peaks in physiological data.

5.3 Relation to neurophysiological results

5.3.1 Two-bars experiment

In section 5.2.2 we investigated scene segmentation using a superposition of three different stimulus objects (Fig. 5.6). The neuron groups representing one stimulus tended to be simultaneously in the so-called slow state or the fast state, thereby segmenting the superposition. Cross correlations between cell groups coding features corresponding to a common stimulus were modulated, while cross correlations remained flat, if the cell groups coded features corresponding to different stimuli. These results are consistent with experimental findings of specifically modulated and unmodulated correlograms [60, 44, 168, 108] (cf. Fig. 2.8). Figure 5.8 shows the results from a simulation where we used a stimulus configuration in analogy to these experiments (using

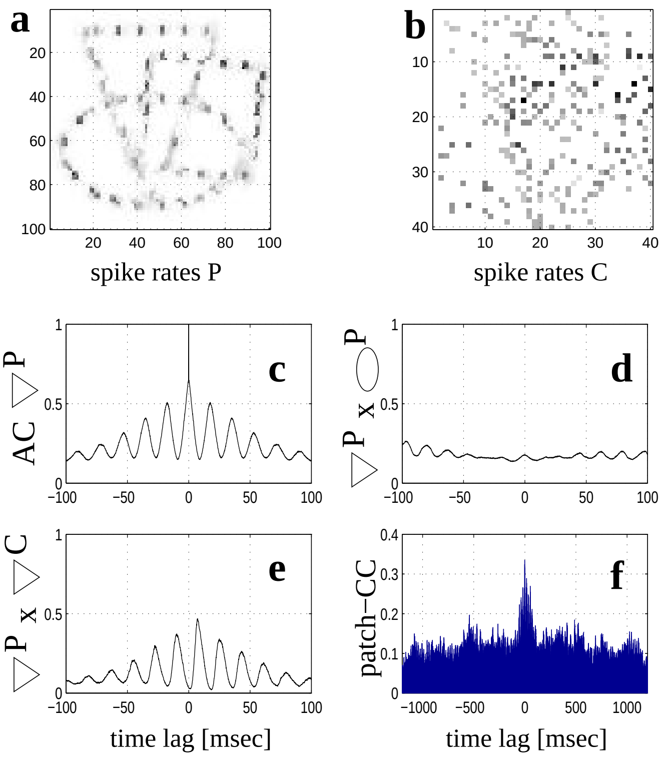


Figure 5.7: Analysis of the spike activity shown in Fig. 5.6 when stimulating with a superposition of three stimuli (triangle, ellipse, rectangle; complete model). **a, b**: Spike rates in population P and C. **c**: Auto-correlogram (AC) for summed spike activity of the triangle assembly (all triangle patches) in population P. **d**: Cross-correlogram (CC) for summed spike activity from the triangle and ellipse assemblies. **e**: CC for the triangle (sub-)assemblies in populations P and C. **f**: CC for the same patch pair as in Fig. 5.5a,b, but on a larger time scale.

static instead of moving stimuli). We recorded simultaneously from two locations testing two stimulus configurations. In the first configuration the two locations were stimulated by one object (Fig. 5.8a,b), while in the second configuration they were stimulated by two different objects (Fig. 5.8c,d). This is reflected in modulated correlograms for the first configuration and flat correlograms for the second.

In earlier proposed *phase coding models* (e.g. [150]) simultaneously presented stimuli are segmented by serially activating the corresponding assemblies in a fixed order within one gamma-period (about 30 msec). However, these models inconsistently would predict correlograms with shifted peaks if the two groups code different entities.

5.3.2 Inter-areal synchronization

Interpreting area C for example as V2 or V4, the finding of anti-phase interactions for the fast oscillatory synchronization between areas P and C (Fig. 5.7e) is inconsistent with experimental results of zero-phase synchronization between connected visual areas (e.g. [41, 43, 45, 52]) which depends apparently on cortico-cortical connections (e.g., [43, 122]).

Further simulations illustrated in Fig. 5.9 (see also [92]; cf. [150, 16, 17]) showed that activity from the two areas is correlated with either zero-phase or anti-phase depending on the inter-areal transmission delays and the oscillation period T. Small delays support zero-phase, medium delays around $T/2$ result in anti-phase, whereas large delays around T support again zero-phase.

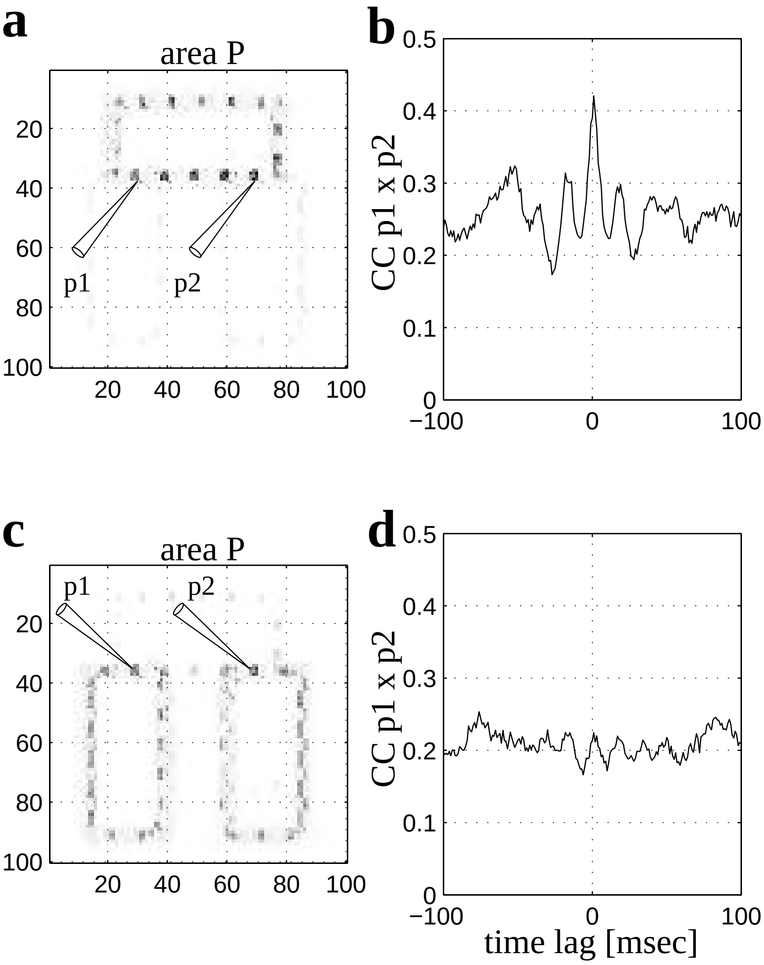


Figure 5.8: Analogy of the 'two-bars-experiments' ([60, 44, 168, 108]; cf. Fig. 2.8). **Top panels**: First stimulus configuration: One single rectangle. **Bottom panels**: Second stimulus configuration: Two different rectangles. **a**: Spike rates in area P (black patches indicate activity) for the first stimulus configuration. Activities of patches p1 and p2 as indicated were used for correlation analysis. **b**: Cross correlation for activities of patches p1 and p2 indicates synchronization (central peak and side-peaks). **c**: Spike rates in P for the second stimulus configuration. Same patches p1 and p2 as in (a,b) were used for correlation analysis. Now the two patches are activated by different objects. **d**: Flat cross correlation for patches p1 and p2 indicates uncorrelated activity.

For the simulations so far we used transmission delays around 7msec (2.5msec - 10msec, see [92] for details), which is about $T/2$ for an oscillation frequency around 60Hz. Using smaller inter-areal delays yields zero-phase correlations as in the neuro-physiological experiments, but then our model does not exhibit the scene segmentation capability as described before, because inhibition cannot be bridged by the small delays. Since experiments rather support broad delay distributions reaching values in the tens of milliseconds [179] we simulated a model variant with a bimodal distribution of inter-areal transmission delays (including very small delays corresponding to fast conducting myelinated axons but also broadly distributed large delays corresponding to slowly conducting unmyelinated axons). We obtained both (near) zero-phase correlations *and* scene segmentation [92]. Moreover, we proved in further simulation studies [97, 98] that spike-time-dependent synaptic plasticity [112, 15, 53, 1] in an oscillatory regime strengthens synapses with delays in the range of one oscillation period such that zero-phase-lag is stabilized even for realistic delay distributions which would otherwise lead to anti-phase correlations.

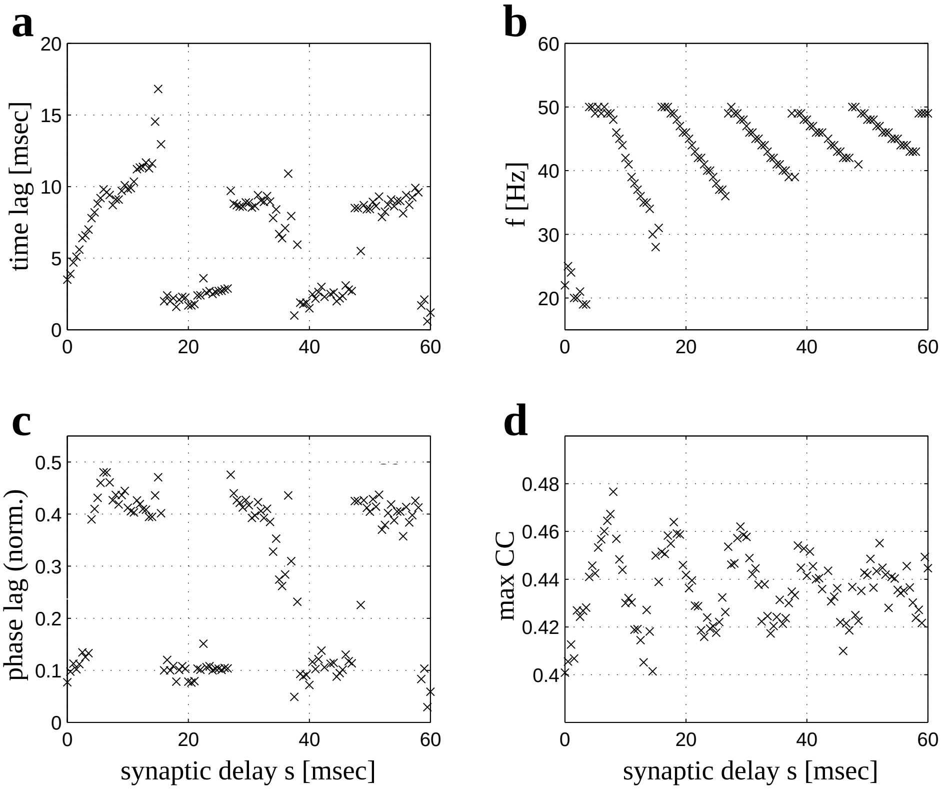


Figure 5.9: Dependences of $P \times C$ cross correlation properties on inter-areal synaptic delay parameter s (see text). **a**: Time lag of the cross correlation peak nearest to zero (absolute values). **b**: Oscillation frequency in populations P and C (measured by Fourier analysis of the correlograms). **c**: Phase lag of cross correlation peak; same values as in (a), but normalized to one oscillation period. **d**: Height of cross correlation peak (normalized values).

5.3.3 Synchronization on larger time scales

Correlation analysis of spike activity from neurophysiological recordings reveal evidence for synchronization on different time scales [123, 127, 126, 125]. Widths of correlogram peaks show a tripartite distribution classified as T(ower) (width at half height < 16 ms), C(astle) (16 − 180 ms) and H(ill) peaks (> 180 ms). Both T and C peaks seem to depend on cortico-cortical connections [122] and thus the broad peaks in our model (cf. Fig. 5.7f) may be related to the C-peaks.

In our model the widths of the broad peaks depend on the duration of the fast state of oscillatory activity (section 5.2.2), and the switching between the slow and fast state is obviously due to habituation of activated neurons. In simulations (see Fig. 5.10; cf. [92]) we manipulated the duration of the fast state by varying the habituation parameters of the neurons. For strong habituation we obtained durations down to 20-50msec where the enhanced periods often degenerated to a single retrieval cycle. For weak habituation the duration of the fast state periods increased up to seconds, and could result in the sustained activation of a single assembly without switching between different assemblies.

The correlograms found in neurophysiological experiments were either flat or showed central peaks, while oscillatory modulations or central troughs were observed rarely. According to our model one should expect oscillatory modulations with central troughs at least for two competing stimuli. To investigate under which conditions our model reproduces the experimentally observed correlograms we used up to seven overlapping stimuli (Fig. 5.11; cf. [92]). While broad

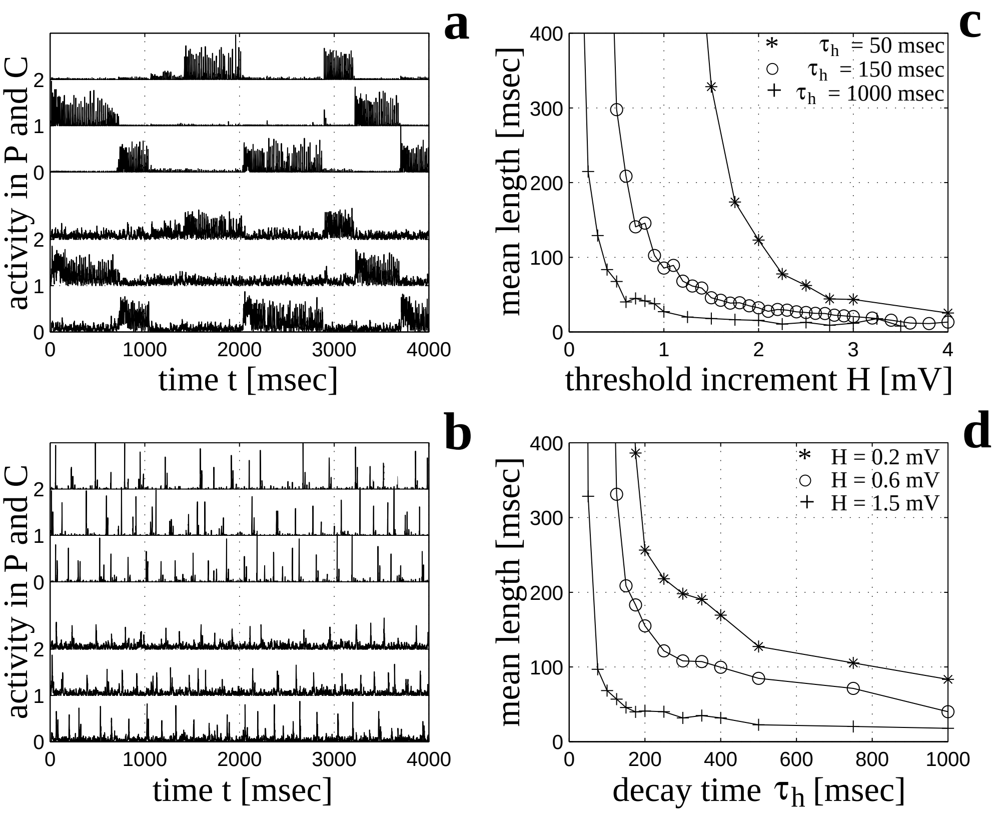


Figure 5.10: Length of the enhanced periods depends on the habituation parameters. **a,b**: Activities in C (top three rows) and P (bottom three rows) for the triangle (top), ellipse (mid), and rectangle (bottom) assemblies. **a**: Example of long enhanced periods ($H = 0.2\,\text{mV}$, $\tau_h = 400\,\text{msec}$, mean length $\approx 500\,\text{msec}$). **b**: Example of extremely short enhanced periods ($H = 3.0\,\text{mV}$,$\tau_h = 150\,\text{msec}$, mean length $\approx$ 20msec). The enhanced periods often degenerate to single retrievals. **c**: Dependence of the mean length of the enhanced periods on the threshold increase H per emitted spike for $\tau_h = 50$ msec ($*$), $\tau_h = 150$ msec ($\circ$, standard value), $\tau_h = 1000$ msec (+). **d**: Dependence of the mean length of the enhanced periods on the decay time constant τ_h of the increased threshold for $H = 0.2$ mV ($*$), $H = 0.6$ mV ($\circ$, standard value), $H = 1.5$ mV (+). Mean lengths were computed for activity in C from 10s trials and averaged over the three activated assemblies.

peaks were absent for single stimuli, and two stimuli induced anti-phasic correlations, we obtained the experimental correlogram peaks if many assemblies are activated at the same time. Then correlograms remain flat for different assemblies and show single broad peaks within an assembly.

A possible conclusion could be that the brain is actually in a 'many-activated-assemblies' regime even without external stimulation, where ongoing activity supposedly wanders permanently through different local assemblies [4, 11, 190, 85, 149]. Indeed, the behavior of our model without stimulation but with enhanced noise level is very similar to the case of many simultaneously presented stimuli. In this view, a stimulus causes the corresponding assembly to spend more time in the fast state as in the other assemblies. This would be in line with recent findings of stimulus dependent two-state fluctuations of membrane potentials [10] and could also explain neurophysiological results described next.

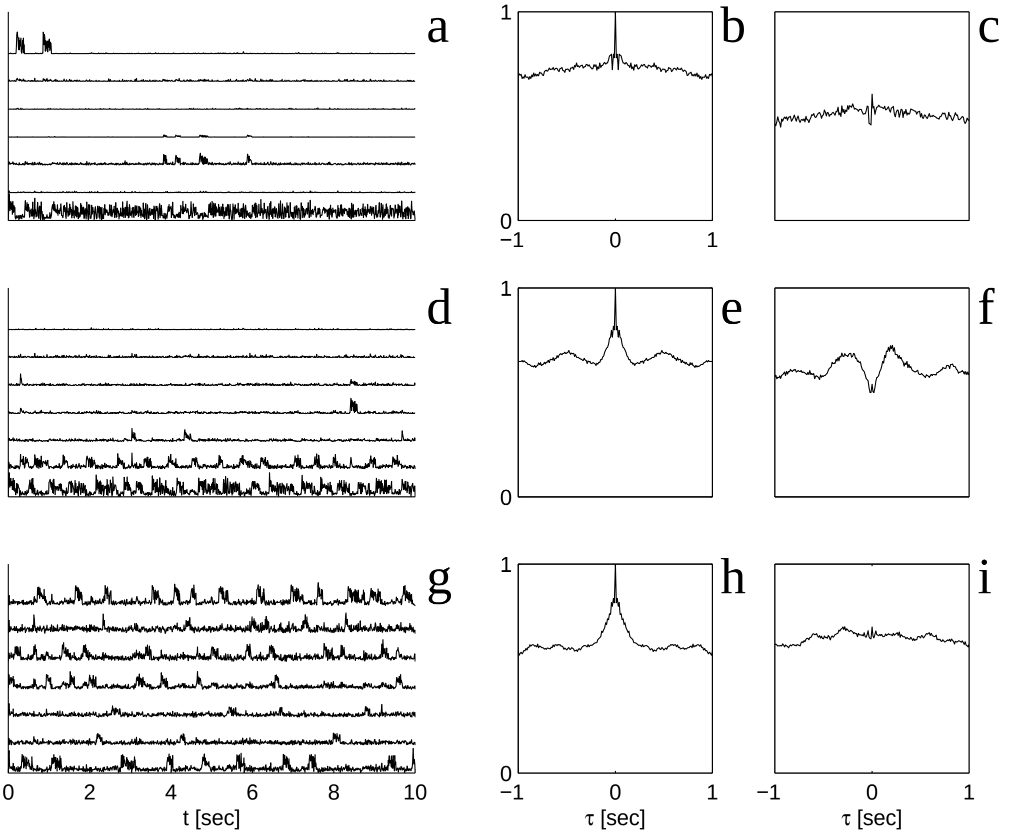


Figure 5.11: Broad peak correlations. **a-c**: Example where one assembly is activated by a stimulus. **a**: Recordings of spike activity for seven assemblies. Only the bottom one is activated. Other assemblies enter spontaneously the fast state. **b**: Auto correlation for the activated assembly. A broad peak is not visible due to tonic activation and absence of competition. **c**: Cross correlation for the activated assembly (most bottom trace in a) and a non-activated assembly (second bottom trace in a) remains flat. **d-f**: Same as a-c but now two assemblies are activated simultaneously by two stimuli resulting in correlograms with oscillatory modulations. **g-i**: Same as a-c but now all seven assemblies are activated simultaneously by seven stimuli resulting in correlograms that are either flat or exhibit single central peaks.

5.3.4 Attention and biased competition

The results from the previous sections suggest an involvement of attentional processes in scene segmentation (cf. [188, 186]). The prevalence of one of the three stimulus objects was switched on a time scale of tens to hundreds of milliseconds (fast and slow state, see section 5.2.2). This may be interpreted as self-generated attention switching serially from one object to the next. For the following simulations we modeled the top-down attentional bias explicitely as additional tonic excitation to the neurons of one selected assembly in area C.

We tried to simulate a scenario as in experiments described in [147] (Fig. 5.12; see also [121, 146, 84, 92]). In this scenario two stimuli in the receptive field of a single neuron were used, a preferred and a poor stimulus. Presenting only the preferred stimulus, the neuron exhibits a strong response, while the response is weak, if only the poor stimulus is used. Interestingly, for the superposition of the two stimuli, the response lies between the responses for single stimuli, indicating competition between the two stimuli. If attention is directed to one of the stimuli, the neuron responds as if only the attended stimulus were present. Thus the effect of attention is similar to a filter that eliminates or weakens unattended stimuli. Another result of the neurophysiological experiments was that the described effect scales with the receptive field size, i.e. the effect was stronger in higher visual areas (e.g. V4 or IT), and weaker in lower visual areas (e.g. V1 or V2).

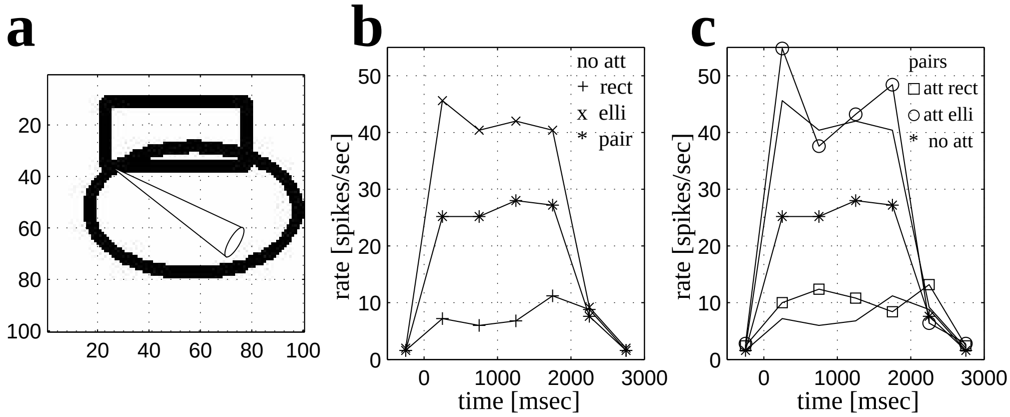


Figure 5.12: Attention and biased competition. Responses of single neurons in P and C when presenting one or two objects. PSTHs (bin size 500 msec) were computed from 5 trials of length 3000 msec. The stimuli were active during the first 2000 msec of each trial. **a**: Stimulus configuration and location of recorded neurons in P. **b**: Responses of a neuron in C when stimulating with the rectangle (+), the ellipse ($\times$), and both the rectangle and ellipse ($*$). The ellipse is the preferred stimulus (response $>$ 40 spikes/sec), the rectangle the poor stimulus (response $<$ 10 spikes/sec). The intermediate response for the pair stimulus indicates competition. **c**: Responses of a neuron in C when stimulating with both the rectangle and ellipse. When attention is directed to the rectangle (*box symbol*) or the ellipse (*circle symbol*) the neuron responds as if the non-attended stimulus were absent.

The results illustrated in Figure 5.12 (see also [92]) show that our model can reproduce experimentally measured neuron responses quite well. It explains the competition and the filter property by two activation states (fast and slow states). For one stimulus a neuron is in the fast state all the time, while with two competing stimuli the neuron is only about half the time in the fast state; attentional bias, however, brings the neuron into the fast state again.

5.3.5 Synchronization on larger space scales

The model as proposed so far should be interpreted as the bidirectional connection of two small patches of cortex. In the real visual system many such structures may be arranged in parallel. To account for more global interactions we simulated a model variant with larger areas (area P size 30×500; area C size 15×250), where area C was modeled as a *topographic* associative memory [92]. While the structure of area P remained unchanged, the connection kernels within area C and between areas P and C were reduced to sizes 31×21 and 15×11, respectively. The kernel sizes in y-direction were larger to model competition in y-direction and topography in x-direction. As stimuli we used three long bars that extended over the whole areas and were represented by three learned assemblies in area C. Fig. 5.13 shows results from a single simulation of the model stimulated with all three bars simultaneously. At time $t = 4$sec attention was directed for three seconds to the upper bar, while attention was absent in the rest of the simulation.

When attention is directed to one of the three bars the whole corresponding assembly enters the fast state, similarly as if only the attended bar were present in the visual field (cf., section 5.3.4). In addition, the globally synchronized oscillations of the smaller model (see section 5.2.2) mutate to waves of activity moving fast along the bar. At a given time the oscillation phase

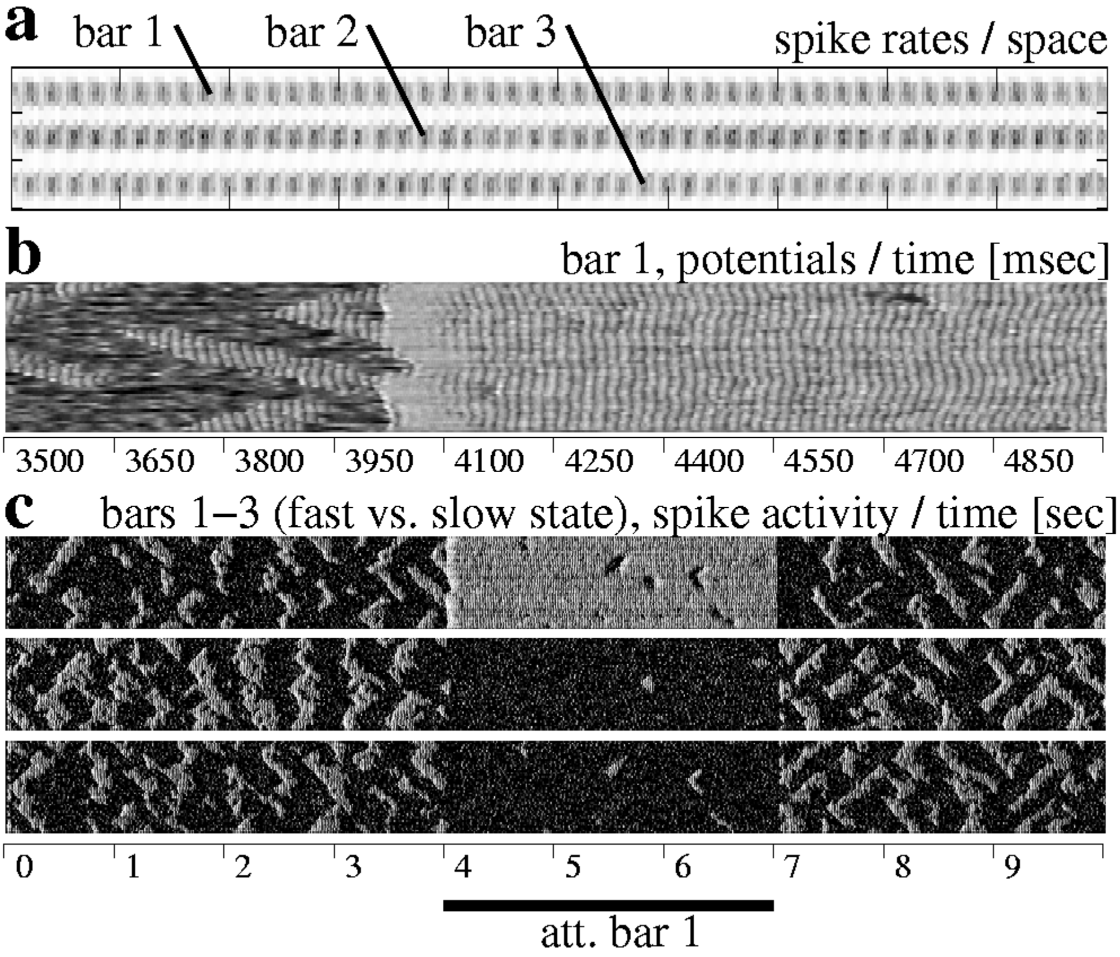


Figure 5.13: Results for a simulation of the larger model stimulated by three bars simultaneously (simulation over 20sec). Between 4sec and 7sec after stimulus onset attention is directed to the (upper) bar 1. **a**: Spike rates in area P. 50 activated patches (black) reflect each of the bar stimuli. **b**: Plot of the membrane potentials vs time for the central neurons of the activated patches in area P for bar 1 (neurons at vertical position 5 and horizontal positions 8,28,...,498; the plot contains one line for each neuron). **c**: Plot of spike activity vs. time for the central neurons of the activated patches for bar 1,2,3 (vertical positions 5,15,25 and horizontal positions 8,18,...,498). White patches indicate strong activation of neurons (fast state). Without attention the fast state is split up between the three bar representations due to competition (slow waves, fluctuating binding). Attention selects one of the bars (static binding).

is a continuous function of the location along the bar representation (Fig.5.13b) reflecting the global shape of the stimulus (cf. [42]). However, fast activity in P and C is synchronized only locally corresponding to the feedback range of intra-areal and inter-areal connections, and correlation analysis reveals that the central peaks of (long-term) correlograms decay with distance [92]. Moreover, short-term correlograms (time window 50msec) show that the central peaks of the long-term correlograms result mainly from averaging over phase-shifts that increase with distance from the recording sites. A similar effect can also be observed for the broad peaks (section 5.3.3).

Without attention the representations of the three bars compete in the higher area C. At a fixed location the situation is very similar to the local model (section 5.2.2): The assembly corresponding to the most salient stimulus wins and enters the fast state. After some time (e.g. some

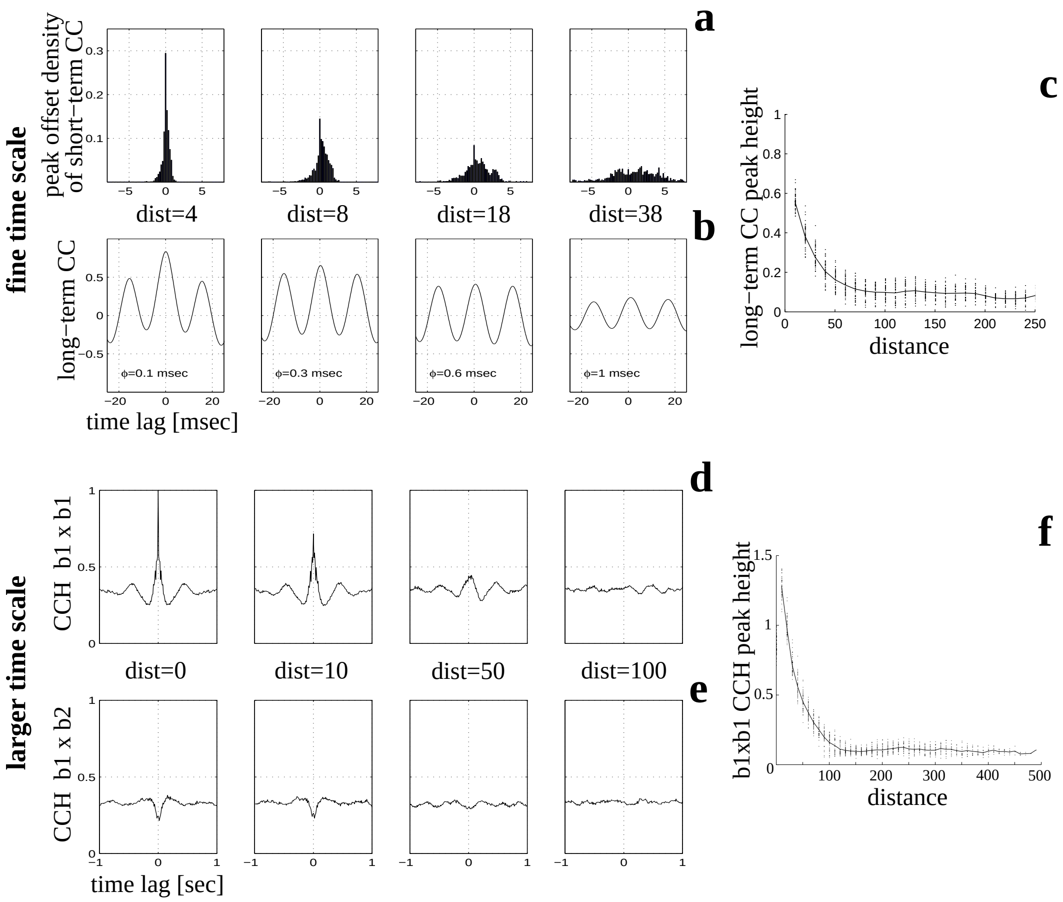


Figure 5.14: Analysis of long-range synchronization of neural activity in the more global model shown in Fig. 5.13 stimulating with three bars. **a-c**: Analysis of activity from single units in activated patches of area P representing the *attended* first bar (cf. top panel of Fig. 5.13c between 4sec and 7sec) on a fine time scale. **a**: Distribution of peak offsets of short-time correlograms (time window 50 msec) for neuron pairs with distances 4,8,18,38 (from left to right). **b**: The corresponding long-time correlograms (time window 10sec) reveal only small peak offsets of $0.1, 0.3, 0.6, 1.0$ msec. **c**: Peak height of the long-time correlograms over neuron distance. **d-f**: Correlation analysis of spike activity from activated patches of area P representing the three bars during absence of attention (cf. Fig. 5.13c between 0sec and 4sec, and also between 4sec and 7sec) on a coarser time scale. Correlograms (bin size 10 msec) are computed from 100sec trials. **d**: Auto and cross correlograms for patch pairs (distances 0,10,50,100) corresponding to bar 1. **e**: Cross correlograms for patch pairs corresponding to bar 1 and bar 2 (horizontal distances 0,10,50,100). **f**: Relative peak height of correlograms over patch distance for patch pairs of bar 1.

hundred milliseconds) the assembly habituates and returns to the slow state, such that another assembly can enter the fast state. However, for the global model it turns out that the assembly switching is a rather local property. At different locations different assemblies can be in the fast state at the same time.

Correlation analysis reveals that synchronization on the larger time scale extends further [92] than synchronization on the finer time scale. Actually it turns out that the synchronization range depends mainly on the duration of the fast state, and is therefore independent of the extension of the underlying synaptic connections. It could therefore be manipulated, e.g., by modulatory synaptic input or attention. This long-range synchronization on a larger scale of space and time might be an important property of the visual system to integrate local features of possibly new objects in the visual field. Although synaptic connectivity is local, global synchronization can be achieved as soon as neighboring local stimulus (sub)representations (e.g., in a topographical area) are compatible with each other.

5.4 Summary of the mechanisms in the model

In the following the main relevant mechanisms in the model are summarized.

5.4.1 Local synchronization

Synchronization within a local cortical patch emerges from the interaction of local excitatory and inhibitory neurons. Typically a few excitatory neurons (the most strongly excited ones) spike first. The excitatory feedback on neighboring neurons launches an avalanche of further excitatory and inhibitory neurons to spike. In the end inhibition prevails because of the larger time constants of inhibitory conductances (cf. τ_{ex} and τ_{in} in Table 2.1 on page 26), non-linear inhibition with an equilibrium potential near the resting potential (cf. E_{ex} and E_{in} in Table 2.1), weaker refractoriness and habituation in inhibitory neurons (cf. τ_h and H in Table 2.1), and the cortical architecture leading to an increasing number of inhibitory neurons firing in a late phase (see Figs. 4.15 and 4.16).

The principle that the most excited neurons fire first and determine by feedback other related neurons to fire, is exactly the mechanism postulated for spiking associative memory (see sections 4.3 and 4.6). The result is a coincident activation of highly connected neuron groups, i.e., of neural assemblies.

5.4.2 Fast oscillations

The interplay of excitation and inhibition leads very naturally to locally synchronized oscillations. Qualitative simulation results (data not shown) suggest that the *amplitude* of the oscillation is determined mainly by delays of the inhibitory feedback loop. E.g, large amplitudes are caused by large transmission delays with a small variance, but also by weak connections from the excitatory onto the inhibitory neurons, or by strong connections from excitatory neurons onto excitatory neurons.

The *period* of the oscillations is also a non-trivial phenomenon. The oscillations become usually faster for stronger extra-areal input. This happens both for constant gradual extra-areal input (as input from area R to area P in our model), but also for synchronized spike input (areas P and C in our model). For the latter case, two reciprocally connected areas can even enter a resonant state where one (or two or even more) waves of spike activity can drive each other (see below).

If the spikes of inhibitory neurons arrive asynchronously at the excitatory neurons the oscillations will come to an end. This happens for example if the variance of the transmission delays are on the order of the oscillation period (i.e., for fast oscillations), or by strong extra-areal input onto inhibitory neurons.

In general the amplitude and period of the oscillation depends in a quite complicated way on the strengths and transmission delays of the connections between the different excitatory and inhibitory neuron populations. Analysis of simplified architectures can be found in [197], where usually the presence of transmission delays and connections from inhibitory neurons onto inhibitory neurons are ignored.

5.4.3 Intra-areal long-range synchronization

Within one (e.g., topographically organized) area synchronization of fast oscillations (gamma range, 30-60Hz) is possible only locally corresponding approximately to the range of intra-areal synaptic connections, or common input from another area. Actually, synchronization as measured from correlogram peak heights decreases continually with distance. Although the peaks of long-term correlograms are usually centered around zero delay this is only an effect of averaging in the relatively large time window. It turns out that for larger distances practically all phase relations can occur in short-time correlations, whereas only in long-term correlations phase 0 prevails while the height of the central peaks decreases to zero with increasing distance. Thus, our results give evidence against the idea of global phase coding in the gamma range (e.g., [150]) where it is assumed that representations to be bound are synchronized *globally* (time window 1-5msec) in a certain phase of the gamma period ($<$ 30msec).

5.4.4 Inter-areal long-range synchronization

Oscillatory activity from distant cortical areas can be synchronized with zero-phase lag if (1) the transmission delays are small [150], (2) if the inter-areal connections with larger delays couple specifically onto inhibitory neurons [23] (which is not very plausible since this would assume inhibition within an assembly), or (3) if synapses of inter-areal connection with delays around multiples of the oscillation period are specifically strengthened by spike-time-dependent synaptic plasticity [15, 53, 112, 97].

Using only small transmission delays with a small variance resulted indeed in zero-phase synchrony in our model. However, then the attention switching described below was absent. Zero-phase synchronization in conjunction with switching can be obtained if many synaptic connections exhibit small delays (e.g., due to myelinated axons) and the remaining connections have broadly distributed delays (e.g., due to unmyelinated axons). In this case we can even expect that a spike-timing-dependent synaptic learning rule as described in neurophysiological experiments will strengthen the connections with delays around multiples of the oscillation period as postulated above in item (3).

Similarly as argued in section 5.3.5, synchronization as defined by long-term correlograms exhibiting central peaks around zero delay does not mean that the neuron populations are actually permanently synchronized as required, for example, by phase-coding models postulating binding by synchronization of gamma oscillations (e.g., [150]). Instead, practically all phase relations occur, and only using a larger time window the different phases average out to zero-phase-lag peaks in the correlograms.

5.4.5 Attentional switching between two activation states

When stimulating with a superposition of several stimuli the result is an activation of all corresponding representations on a large time scale of spike rates (or at least the 3sec window of [139]). On a finer time scale one observes a competition of the representations at locations where they are overlapping, i.e., at locations where different activated local assemblies are controlled by the same inhibitory neurons.

For our model of two reciprocally connected cortical areas we observed two activation states characterized as a *fast state* (corresponding to fast synchronized oscillations) and a *slow state* (slower less synchronized activity). After stimulus onset the most excited local assembly at a given location enters the fast activation state. After some time (e.g., $50 - 500$ msec) the excitatory neurons of the assembly get habituated and enter the slow state, while the most strongly activated one of the remaining assemblies enters the fast state. The switching corresponds to a rather non-oscillatory synchronization on a larger time scale reflected by broader peaks (widths at half height perhaps also $50-500$ msec) in the correlograms. In contrast to the fast gamma oscillations the range of synchronization is quite independent of the underlying synaptic connectivity and can reach arbitrarily far for weak habituation or an attentional bias.

The *two* states of activation result from the reciprocally connections of *two* cortical areas, where an activated assembly in one area can either be supported by a corresponding assembly in the other area or not. If on the other hand a cortical area is reciprocally connected to two (or more) other distant cortical areas then we should be able to distinguish three (or more) activation states corresponding to either no support from another area, support from only one further area, or support from both distant areas.

5.4.6 Binding

In this context binding is defined as the coincident activation of the distributed elements of cells to be bound in the sense of the temporal correlation hypothesis (TCH; see section 2.2.4).

Binding the cells of a *single* local assembly is possible by the local quasi-complete synaptic connectivity, similarly as suggested by the theory of spiking associative memory (see chapter 4, in particular section 4.6) even for a larger cortical area with topographic connections.

Similarly, binding *different* corresponding local assemblies distributed at distant cortical locations (for example within *one* large topographical area, or in *many* different areas) is possible by the *specific* (i.e., Hebbian learned) synaptic connections *between* the local assemblies (either direct or indirect).

Our results do *not* support the *strong version* of the TCH postulating that *global* binding occurs on the fine time scale of the gamma oscillations, i.e., that neurons to be bound have to be synchronized globally within a time window of only $1-5$msec. This kind of synchronization is possible only for neighboring local assemblies where the phase correlations decay gradually with distance: Only nearby cell groups show zero-phase synchronization with central peaks in the correlograms, but distant groups rather uncorrelated activity with flat correlograms (see sections 5.3.2 and 5.4.4).

Our results rather support a *weaker* version of the TCH requiring global synchronization within a larger time window of perhaps $30-300$msec. This type of synchronization is possible over distances larger than connectivity ranges and rather independent of the underlying synaptic connectivity.

In the presence of multiple stimuli the corresponding local assemblies at a given cortical location may compete against each other corresponding to *fluctuating binding* of the distributed representations. The binding can become *static* in the presence of an attentional bias, similar as only one stimulus were present.

In the next chapter we will further discuss possible solutions to the different binding problems in the light of our simulation results.

Chapter 6

Binding in the brain

6.1 Is there a binding problem in the brain?

In the introduction we have argued that it is plausible to assume that representing internal and external states in the brain is accomplished in a *non-trivially* distributed way (see section 2.2.1). This means there is probably no single location or center in the brain where *all* the information necessary for a coherent unified percept is concentrated. Alternative views suggesting localized representations such as the so-called grandmother neuron inevitably lead to a combinatorial explosion of the number of necessary locations (e.g., neurons).

Further there is plenty of evidence from neuroanatomy, neurophysiology, and neuropsychology that the cortex is organized in anatomical and functional processing streams on both global and local levels. For example it is commonly accepted that visual processing in the brain on the global level is segregated into a dorsal ('where') and a ventral ('what') stream. The global streams are further divided on a more local level, such as separate processing of form and color. On the neural level this is reflected by specialized cells. A neuron can be activated by an object moving in a certain direction irrespective of the form or the color. Another neuron may respond to a certain form (e.g., a face) irrespective of its color and location in the visual field. This fits well to the fact that neurological patients suffering, for example, from brain lesions have very specific difficulties depending on the location of the lesion (e.g., [47]). There are patients that can copy the features of a colored form quite good with paper and pencils, but cannot *recognize* the form itself (visual agnosia) [48], or other patients that specifically have problems with the recognition of moving objects [47]. Even on a local level of columns in primary cortical areas (e.g., M1 or V1) where the interpretation of neural responses is still relatively easy, neurophysiological experiments revealed population codes: For example in V1 the tuning of orientation selective cells is relatively broad, and a visual stimulus activates usually a large number of orientation selective cells [184]. Thus the information of the actual orientation of a contour is coded in the activation of a large group of neurons which can be identified with local cell assemblies (see sections 2.2.2). The situation is similar for the prediction of the direction of a hand movement when recording from primary motor cortex [54].

All these arguments are support for the hypothesis that information processing in the brain is distributed in a non-trivial sense (see section 2.2.1) which causes "the" binding problem. Actually there are several distinct binding problems which may be one reason why there is still some controversy on the question if there is actually a problem of binding in the brain (e.g. [56, 163, 148]; see also section 2.2.3). Below we will discuss a classification of different binding problems relevant for our purposes.

6.2 Critique of the strong Temporal Correlation Hypothesis

6.2.1 Postulations of the strong TCH and phase coding

The temporal correlation hypothesis (TCH) postulates that binding of distributed representations in the brain is accomplished by neural activity that is *correlated* in the temporal domain. While the term "correlated activity" is quite general and could, for instance, also mean complex repeating spatio-temporal spike patterns such as synfire chains [2, 3], the predominating interpretation is that of a *synchronization* of spike activity [191, 168].

The specific prediction of the *strong version* of the TCH is a *global* synchronization of elements to be bound within a very *narrow* time window of perhaps 1-5 msec [191, 192]. This idea has been amplified and modified by neurophysiological observations of high-frequency oscillations (gamma range; 30-60Hz) in the visual cortex apparently reflecting global stimulus properties [41, 61, 168]. *Phase-coding models* are popular instantiations of these ideas where it is commonly assumed that one gamma period (15-30msec) can be partitioned into several time slots, and that distributed entities to be bound are activated in one of the time slots (e.g., [150]). Thus multiple objects are represented by the sequential activation of the corresponding cell groups, where the cells representing the same object are activated within the same time slot. Usually the *order* of activation does not change from one oscillation period to the next.

In simulation experiments [150] it has been found that the size of one time slot is about 4-5 msec for realistic neuron parameters (which coincides quite well with the retrieval time in our biological model of associative memory, cf. section 4.6). Thus within one gamma period there would be room for about four to seven time slots. If such models are interpreted at the level of global binding this would mean that up to about seven entities can be integrated to a unified conscious percept. This conclusion tempted several investigators to speculate about a relation to results of psychological experiments where humans can hold up to 7 (± 2) items in working memory (cf. [77, 150]).

6.2.2 Critique of phase coding in the gamma range

There are several arguments why we should be cautious with the interpretation of the simulation results from the phase coding models. First, the architectural relation to the real cerebral cortex is obscure for most phase coding models. Usually a Hopfield-like architecture is used which does not reflect a number of neuroanatomical facts or commonly accepted assumptions (see section 2.3.4; cf. [150]). For example, real neurons are either excitatory or inhibitory while Hopfield neurons (or rather Hopfield *units*) can make both types of synapses, and the sign of a given synapse can even change during learning. This discrepancy is sometimes excused by arguing that a Hopfield unit corresponds actually to a small group of neurons (perhaps 10-100) consisting of both excitatory and inhibitory cells. But this raises new questions concerning the interpretation of a spike within such a cell collection, in particular because phase coding models intend to make a strong point with respect to fine timing of spike activity.

Phase coding within one cortical area modeled as a Hopfield associative memory (e.g., [150]) is apparently based on the following *mechanisms*: (1) Global excitation within one assembly in the area; (2) direct global mutual inhibition between different assemblies; (3) local inhibition such that each Hopfield unit can oscillate independently of the others. When the first neurons of one assembly start firing the fast global excitatory feedback (1) synchronizes the remaining neurons of the same assembly within $4-5$msec, similar to our model (see section 4.6). At the same time the fast global inhibition (2) delays the activation of concurrent assemblies. For this the global inhibition must be weak and/or last only shortly (shorter than the excitation) because other assemblies shall fire in a later time slot of the gamma period, but shall not be suppressed completely. Finally the strong and long lasting local inhibition (3) shapes the gamma oscillation. (1) and (2) are a direct consequence of Hopfield's learning rule. (3) needs an additional neuron population, usually one additional reciprocally connected inhibitory cell per Hopfield unit. Due to the complete connections the described Hopfield phase coding model of a cortical area must be interpreted as a small patch of cortex where dense connections between neurons are possible (e.g., 1 mm^3), similarly as argued for the Willshaw model in section 2.3.3.

Thus Hopfield-like phase coding models for local cortical networks make several strong neuroanatomical and neurophysiological assumptions which are difficult to prove or falsify. Although we cannot exclude by this argumentation that this kind of phase coding occurs in *local* cortical networks it is hard to believe that phase coding as described scales to the *global* level. Although many authors have simulated two (or more) reciprocally connected Hopfield areas, this is usually done just to demonstrate that zero-phase synchronization is possible even between distant areas with considerable delays. For example Ritz et al. [150] simulated a single area using multiple stimuli to demonstrate phase-coding, and two reciprocally connected areas using a single stimulus to demonstrate zero-phase synchronization, but they did *not* simulate *two* reciprocally connected areas using *multiple* stimuli in order to demonstrate global binding by phase coding in the gamma range. I suspect that this would require both (1) small transmission delays, and (2) Hopfield-like inter-areal connections in order to prevent the destruction of the sequential order of corresponding time slots in the two areas. I.e., it would require that the reciprocally inter-areal connections were functionally equivalent to the intra-areal connections (and therefore the two areas would collapse to one single larger area). In particular (2) would be in clear contradiction to neuroanatomical results. It is well known that inter-areal cortico-cortical connections are almost exclusively excitatory (see section 2.1.3), while the Hopfield architecture would require mixed excitation and inhibition. The argument that the required long-range inhibition could be provided indirectly if long-range excitation activates local inhibition polysynaptically is also not convincing because this would even worsen the transmission delay problem (1). However, it will probably require further simulations to confirm these suspicions.

The probably most convincing argument against gamma range phase coding is lacking or even contradicting evidence in neurophysiological experiments. In the so-called two-bars-experiments ([44, 168, 108]; see Figs. 2.8 and 5.8) spike activity is recorded simultaneously from two neuron groups at two neighboring locations with overlapping receptive fields. When stimulating with a *single* coherently moving bar the correlograms show central peaks corresponding to synchronized spike activity. However, if the two receptive fields are stimulated with *two* bars moving in different directions the correlograms remain flat whereas phase coding models would rather predict shifted peaks.

On the other hand there are indeed experiments where correlograms exhibit shifted peaks which have been interpreted as evidence supporting the strong version of the TCH and even phase coding. For example in [106] simultaneous recordings from two neighboring neuron groups in two different orientation columns of primary visual cortex with different orientation preferences but largely overlapping receptive fields revealed correlograms with systematic phase shifts of a few milliseconds. But here the stimulus was a *single* bar where phase-coding models predict no phase-shifts. A better interpretation for these results would be that the most excited neurons best matching the orientation of the bar stimulus fire a few milliseconds earlier than less excited neurons with non-matching orientation preference. This is exactly what happens also in our model of spiking associative memory (see section 4.6): The first spikes correspond to the most excited neurons and determine the result of the retrieval (see Fig. 4.16; cf. [199, 183]).

Finally, there is a principle problem in the interpretation of neurophysiologically observed synchronization especially over longer distances. Usually the experimenters record from one or more stimulated neurons. The length of recordings used for computing the correlograms are usually several hundred milliseconds and involves in addition averaging over several trials. Thus the interpretation of central peaks in such correlograms must be interpreted as an average effect. Indeed some authors stress that zero-phase (unlagged) synchronization occurs only on average, in particular for more distant recording sites [41, 52, 42]. This is consistent with simulation results from our model. Here gamma synchronization as revealed by central peaks in correlograms can reach some millimeters, where the peak height decreases gradually with increasing distance of the recorded neuron groups. However, analyses using time windows of only one or two gamma cycles reveal that most of the time the neuron groups do not fire in synchrony (see section 5.3.5; cf. Figs. 5.13 and 5.14). Only when averaging over a larger time window (of hundred milliseconds or more) the various phases average out to zero-phase. This effect happens for long-range synchronization within one area (Figs. 5.13 and 5.14; cf. [41, 42]), but also for inter-areal synchronization (Fig. 5.9d; cf. [52]). A good indicator of this effect are the decreasing peak heights with increasing distance or transmission delays. Using very simple oscillators this effect is analyzed further in [99]. Even for the model of Ritz et al. this effect can be seen in the correlograms (see Fig. 6d in [150]).

The latter argumentation implies that synchronization over longer distances (e.g., $\geq 1-2$mm) as found in neurophysiological experiments reveals to be a phenomenon of a larger time window of perhaps hundred milliseconds. On the global level this excludes phase coding in the gamma range.

6.2.3 Functional relevance of fine timing

The argumentation above points out that the strong TCH and in particular the postulation of *global* phase coding in the gamma range are probably *not* true. This conclusion has been drawn because of *lacking neurophysiological evidence*, and because *plausible models* (such as our visual model described in chapter 5) incorporating anatomical constraints such as transmission delays and noise *cannot establish* global synchronization within the time window suggested by the strong TCH.

Even if global synchronization on the spike time scale were possible (i.e., if our models were invalid), there still would be the question about the functional relevance. How could the brain read out or make use of the global synchronization within a very narrow time slot of all cortical neurons representing a coherent entity? It is difficult to find an answer (but also to exclude the existence of an answer).

We can assume that making use of synchronization of two cortical locations A and B at a third location C would require at least that the synchronization of A and B can be detected at C. If we assume that the type of synchronization observed in some neurophysiological experiments (e.g., [168]) would really occur *globally* there is still the problem that detection of synchrony will require a larger time window of perhaps hundred milliseconds (similar to the neurophysiologists computing correlograms by averaging over multiple trials each having lengths of tens to hundreds of milliseconds; cf., the argumentation in sections 5.3.2, 5.3.5, 5.4.4, and 5.4.3). But then there is no advantage over binding by synchronization on a larger time scale as suggested by our model.

Note that our argumentation contradicts the use of global gamma synchronization or phase coding for binding, but not the use of fine timing of spike activity in general. Our results rather suggest that spike synchronization is functionally very relevant in local or directly connected cortical networks. In our model of spiking associative memory (see chapter 4) the precise temporal order of the extra-areal input spikes is the crucial factor deciding which assembly gets activated. And this property is passed on to the output spikes: The first spikes are usually the most relevant spikes (see Fig. 4.16; cf. [183]). Also for our model of reciprocally connected cortical areas (see chapter 5) pattern separation in spiking associative memory plays an important role to segment a scene of overlaying objects, whereas the reciprocal connections between the areas stabilize the activation of single representations in a way similar to iterative retrieval in more technical models [160, 170, 171].

Moreover, *local* fine timing within the range of direct synaptic connections is also important for synaptic plasticity. In neurophysiological experiments it has been shown that the order of presynaptic and postsynaptic spikes is decisive if a synapse is potentiated or depressed, where the relevant time window is below 10msec [112, 15, 1, 53]. It has also been suggested that this so-called spike-timing-dependent synaptic plasticity (STDP) plays also a role in synchronization of distant cortical areas connected by reciprocally connections with large transmission delays [97, 98, 100].

Although we argued that *global synchronization* in the gamma range is improbable, the occurrence and functional relevance of complex *global spatio-temporal spike patterns* (e.g., synfire chains) is more likely [2, 3].

6.2.4 Different time and space scales for binding by synchronization

There is considerable controversy about the significance of fine timing of spikes, and about the relevant time window for binding by synchronization. For binding of *global* representations, different models suggest different time windows for synchronization, e.g. a fraction of a gamma cycle (1-5msec; e.g. [150]), one gamma cycle (about 30msec; e.g. [199, 201]), or even several gamma cycles as in our visual model ([91, 92]; cf. chapter 5).

The TCH basically states that binding is accomplished by neuronal activity that is correlated in time. For the time window of conscious experience (about 3sec [139]) there can be no dispute about the truth of a weak form of the TCH (cf. section 2.2.4): Actually, our brain can construct only *one* coherent conscious percept at a time; when looking at Fig. 2.6 we can perceive either the young girl or the old woman, but not a superposition of both. Thus the real dispute about the TCH is rather about which time scale is used for solving which kind of binding problem.

We have argued that binding is also necessary *within* one conscious interpretation of a picture. The introspectively experienced switching between the old and young woman in Fig. 2.6 is similar to the activation switching in our visual model (chapter 5; see Fig. 5.6), but differs with two important respects: First, the switching in our model is not between *different incongruent* interpretations as in Fig. 2.6, but between the representations of different constituents of *one coherent* scene. Second, the time scale of the switching in our model is much finer (at least one order of magnitude) than in the bistable figure. Thus a reasonable conclusion would be that global *synchronization* in the brain is possible within a time window of 30-300msec in order to bind globally distributed local representations, while conscious perception of the whole scene requires the *integration* of one or more such sets of synchronized local representations within a time window of perhaps 300msec - 3sec.

It is important here to distinguish between two kinds of time windows, one for *synchronization*, and another one for perceptual *integration*. The time window for synchronization would be required to express *unity* while in the larger time window for integration *relations* between *different* unified entities can be expressed. Hierarchically, the integration time window of a lower stage can be identified with the synchronization time window of a higher stage. For example for phase-coding models as discussed above (e.g. [150]; see section 6.2.2) the synchronization window is 1-5 msec and the integration window is one gamma cycle (about 30msec). In our visual model (chapter 5; see Fig. 5.6) the synchronization window is perhaps 30-300msec, and the integration window for representing a scene containing three shapes perhaps 100msec-1sec.

The results and arguments presented in this work suggest a raw classification of relevant time and space scales for binding as illustrated by Table 6.1. The time window for local synchronization is 1-5msec corresponding to the duration of a retrieval in our model of spiking associative memory interpreted as a model of a local cortical network (see section 2.3.3). The integration window for local representations is perhaps 30-300msec corresponding to the interval between two retrievals (about one gamma cycle) or even the time necessary for iterative retrievals [160, 171]. The integration window for local representations coincides with the time window for global synchronization, whereas the time window for global integration corresponds to the time window for conscious experience (up to 3sec; [139]). This integration time window corresponds to a synchronization time window on a further "spatial" level for the representation of multiple interpretations. However, the representation of multiple interpretations is apparently not implicitly integrated in the brain. For this we need additionally explicit conscious reflection and memory.

time window	local	global	(mult. interpr.)
1-5msec	synchronization		
30-300msec	integration	synchronization	
300msec-3sec		integration	synchronization

Table 6.1: Different time windows for binding on different spatial (or representational) levels. The representation of multiple interpretations is apparently not integrated in the brain: At any time we can see only *one* of the possible interpretations of a scene (cf. Fig.2.6).

The classification of time and space scales suggested by this work is consistent with several theoretical and experimental results. (1) The division of space into a local and global level reflects basic results of neuroanatomy concerning a fundamental difference between local cortical networks with a very dense connectivity within a volume of perhaps 1mm^3 and global cortical networks involving patchy cortico-cortical connections ([27, 3]; see also section 2.1.3). (2) The division of time into three levels reflects neurophysiological observations of a tri-partite distribution of peak widths in correlograms computed from spike activity [123, 127, 126, 125]. (3) Based on psychological experiments Pöppel suggested a bi-partite division of time: A narrow time window of 30msec being the subjective "horizon of simultaneity" or the "temporal quantum" for behavior, and another broader time window of 3sec for a coherent conscious experience [139, 141, 140]. Actually our tri-partition just extends Pöppels bi-partition by one intermediate level which can be interpreted as a further structuring of the 3sec time window. (4) The suggested classification would even be consistent with alternative models such as gamma phase coding at a *local* level, although the argumentation in section 6.2.2 rather excludes gamma phase coding on a *global* level.

6.3 Sketch of a more global model of binding in cortical areas

6.3.1 Classification of different binding problems

To investigate synchronization of distributed neural assemblies we have proposed a model of two interacting cortical areas which is compatible with several lines of experimentally observed phenomena such as synchronization on a fast time scale [40, 168] or a slow time scale [123], ongoing activity in optical recordings [190], two-state fluctuations of membrane potentials [10], and attentional or biased competition [147].

Neural assemblies are distributed representations that actually need binding of their components, probably by synchronization of spike activity. Before we discuss possible solutions to the binding problem (BP) in the context of our simulations, note that the BP occurs at different levels of difficulty.

BP0 Binding between neurons in *local* cortical network.

BP1 Binding between *distant* neurons representing externally correlated or co-occurring features.

BP2 Binding across different, often unrelated feature dimensions (such as form and color) within one assembly.

BP3 Dynamic binding (e.g. role assignment) relating different assemblies, where possibly fast synaptic plasticity is required.

6.3.2 Possible solutions

The simplest binding problems are BP0 and BP1. At least BP0 is well understood by the application of the theory of associative memory (see chapters 3 and 4). BP1 is to bind together corresponding subrepresentations like the features of a complex object and perhaps also an abstract representation of the whole object in a higher area. This problem is relatively simple because we can assume that specific synaptic connections exist between the subrepresentations. So far, our model accounts only for this simplest BP: The features are the orientation patches in the peripheral area P, and the abstract representation are the object assemblies in the central area C. Another instructive example is the famous picture of the dalmatian dog composed of a chaos of black patches, (Fig. 2.5; cf. [40]). Here one can observe that BP1 occurs in two subforms: It is rather difficult if the abstract representation does not (yet) exist as it is difficult to see the dog for the first time. In contrast, if recognized only once, the dog is much easier to recognize the next time. So we can assume that actually an abstract representation of this picture was allocated in our brains that facilitates the binding of the black patches. However, we are also capable of recognizing the dalmatian dog if we see the picture for the first time, although it will take some time. We believe that binding in our larger model (Fig. 5.13) — where no global representations of the bars exist — is similar to this process (although much easier). A refined coding strategy would not require an extra code for each object. For example different objects could be segmented by local properties alone, such as continuously aligned contours, cf., e.g. [124]. A process like the slow waves in our simulations could help to segment a visual scene, although on a relatively large time scale.

A more difficult problem is BP2 that occurs if binding across different feature dimensions (such as form and color, cf. Fig. 6.1) is required. In this case we cannot always expect specific synaptic connections between the different processing streams. For example artificial objects in the visual field can occur with arbitrary colors, and combinatory coding each of n forms and each of m colors would require $n \times m$ representations. Separating the feature dimensions, however, requires only n+m representations plus an adequate binding mechanism. Thus, it would be very inefficient and therefore surprising if the brain would not take advantage of the possibility of separating different feature dimensions.

Figure 6.1 sketches a more global model to solve at least BP1 and BP2 that follows from our simulations. Binding between subassemblies of two connected areas, i.e. BP1, is accomplished by reciprocal excitation leading to the fast synchronized activation state. Local assemblies in one area are competing with each other and only the winning assembly is routed to the next higher processing stage. The range of competition is larger for higher areas. The bias of external excitation (e.g., attention) can select the subassemblies to be processed globally. While the assembly corresponding to a "non-winning" (non-attended) object is suppressed in a high area, pre-attentive processing may still be possible in a lower area where the corresponding assemblies do not compete due to spatial separation. Binding across different processing streams may be possible due to the indirect reciprocal connections via the primary areas and/or bottom-up synchronization mechanisms, e.g., locally correlated random fluctuations [72], or a localized 'searchlight' [35].

An even more serious problem is BP3 which occurs if dynamic binding is required between different entities. This occurs for example if one tries to understand the sentence "Tom beats Fritz". Here one can assume the existence of previously learned representations of the involved persons (Tom and Fritz), and also representations of the meaning of "to beat" which involves representations of a beater and a victim, but no prejudicial bias of the role of Tom and Fritz. Here the binding problem is to assign these roles dynamically to Tom and Fritz. One solution to BP3 could be short-term plasticity that would temporarily connect the representations 'Tom' with 'beater', and similarly 'Fritz' with 'victim'. Although this fast synaptic plasticity (on a time scale below a second) was proposed already long ago [192] it is experimentally not easy to measure and its existence is still uncertain today. So the question arises if and how BP3 can be solved without fast synaptic plasticity.

Note that the structure of BP3 is very similar to BP2. BP2 involves for example understanding a visual scene containing a red apple and a green pear where temporary associations between 'red' and 'apple' and between 'green' and 'pear' are involved similarly as for BP3. As we discussed above BP2 can be solved by a topographical representation where basic feature representations of the apple and the pear are still connected with the correct color representations due to spatial separation. This spatial separation principle could also work for BP3 which would reveal it as a pseudo-problem. A possible solution is the following: "Tom beats Fritz" must be translated into a *topographical generic feature representation* where a generic feature representation of 'Tom' is connected with a generic feature representation of 'beater', and the same for 'Fritz' and 'victim'. This could happen for example by visual imagination of a visual scene where Tom beats Fritz. However, since we seem to understand "Tom beats Fritz" even without visual imagination there may be involved also non-visual topographical representations. Although it is true that no synaptic plasticity is required by this solution for BP3 there is still a need for a kind of "working memory" to preserve e.g. an imagined scene by sustained activity.

Another solution to BP3 not requiring sustained activity or fast synaptic plasticity suggested

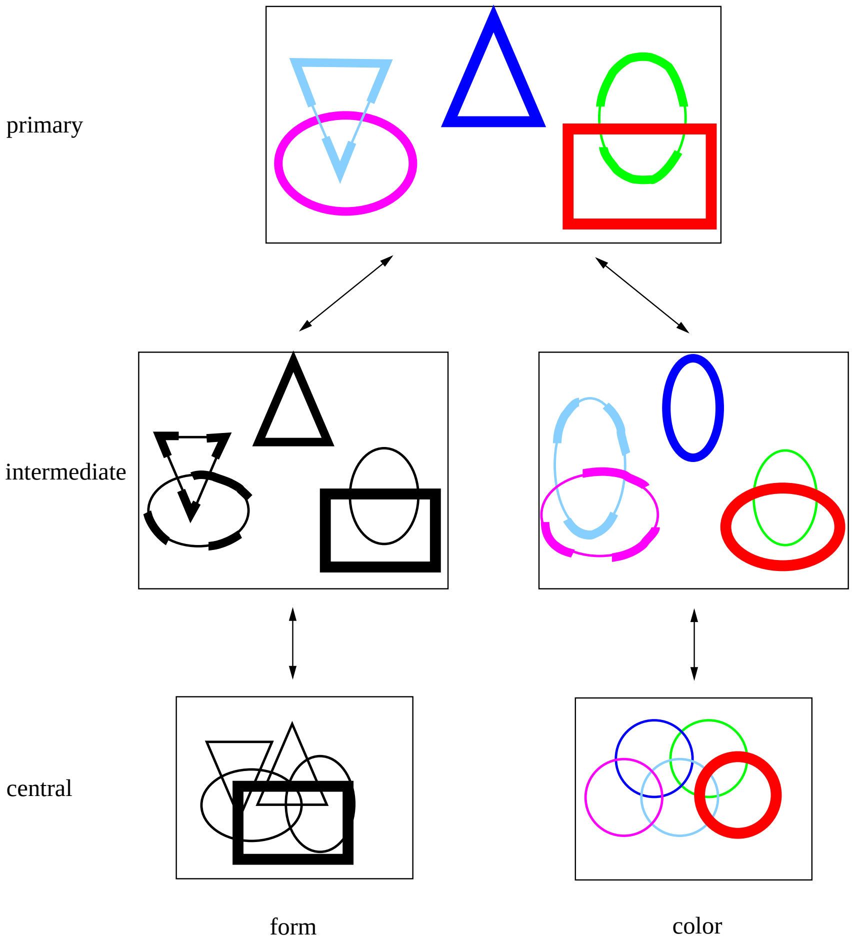


Figure 6.1: Suggested binding by "slow waves" in a global model of three stages and two processing streams (form and color). Thick lines correspond to the fast enhanced activity state, thin lines to the slow state. In the topographically organized primary area objects separated in space do not compete with each other and are therefore in the fast state all the time. Competition, i.e., switching between different assemblies, occurs only at places where objects are overlapping. In the smaller topographically organized intermediate area nearby objects compete, and in the central area there is global competition. In the central area the "winning" (e.g., attended) assembly forces by feedback the corresponding assemblies in the peripheral areas to the fast state, and suppresses overlapping competing assemblies. Binding over different streams is accomplished indirectly via the primary area due to the reciprocal connections.

by our simulations would rely on common habituation levels of corresponding representations. For example hearing "Tom beats ..." would first activate two assemblies 'Tom' and 'beater' at two different cortical locations for some time. Subsequently hearing "... beats Fritz" would similarly activate the representations 'victim' and 'Fritz' in the same two areas representing persons and roles, respectively. After suppressing activity in the two areas binding information would still be preserved in the habituation variables. If the two areas are unspecifically activated (assuming that all 4 assemblies are equally primed) one would obtain again 'Tom' and 'beater' because they are less habituated than 'Fritz' and 'victim'.

Fast but decaying synaptic plasticity would improve binding in any case. How well the proposed binding mechanisms operate remains to be seen in further future work.

Chapter 7

Conclusions

This thesis mainly contributes to the theory of *neural associative memory* [174, 204, 104, 129, 69, 105, 134, 29, 160, 171], and to the theory of *cell assemblies* in the brain [64, 26, 130, 192, 133]. The results of this work are relevant for technical applications, e.g., for information retrieval, but also for brain theory and perhaps also for (the connectionist tradition of) artificial intelligence.

On the one hand, neural associative memory models provide efficient methods for *information retrieval*. This is true in particular for the classical binary associative nets proposed by Steinbuch [174], and later analyzed by Willshaw et al. [204] and Palm [129] which is in the following referred to as the **Willshaw model**. It delivers an algorithm useful in technical applications for rapid and fault tolerant access to stored pattern information, where performance can be superior to classical algorithms of computer science like search-trees, look-up-tables, or hash-tables. Here the focus of this work is on optimal exploitation of the physical memory, while maintaining a high retrieval speed, and good fault tolerance. In particular, the role of spikes in the separation of superimposed patterns has been investigated by the development and analysis of a spiking variant of the Willshaw model, the so-called spike counter model.

On the other hand, neural associative memory can be seen as a model for local networks in the cerebral cortex [27], and is therefore closely related to **assembly theory**. Here the auto-associatively stored patterns can be identified with Hebbian learned cell assemblies, i.e., discrete neuron groups presumedly representing internal and external entities (e.g., thoughts and objects) in the brain. A biological version of the spike counter associative memory has been developed and simulated as a model for a small patch of cortex using a more realistic spiking neuron model. Similar as for the technical associative memory our focus is on the separation of patterns overlapping in space and time. On a more global level these questions are related to the feature binding problem and to phenomena like spike synchronization and fast oscillations observed in neurophysiological experiments. In particular, this work investigated predictions of the so-called temporal correlation hypothesis [192, 168] which postulates, for example, that binding in the brain is accomplished by global synchronization of spike activity of the neurons coding the entities to be bound (e.g., the features of a common object). For this purpose a larger model of two visual cortical areas has been simulated, one of them a primary area similar to V1, the other one a more central area modeled as spiking associative memory containing learned representations of the stimulus objects.

7.1 Contributions to the theory of Willshaw associative memory

In **chapters 3 and 4** the classical theory of Willshaw associative memory is extended in three directions: (1) Compression of the memory matrix, (2) using spiking neurons, and (3) improving the analysis technique.

One classical result of the analysis of the binary Willshaw model is that for sparse patterns with $k = \operatorname{ld} n$ (where n is the length of the binary pattern vectors, and k is number of one-entries) a high storage capacity of $\ln 2 \approx 0.7$ is possible asymptotically for $n \to \infty$ which is much higher than for alternative (e.g., Hopfield-like) models. This means that the Willshaw model can store asymptotically about 0.7 bit of information *per synapse*. Since *binary* synapses are used this means also that the storage capacity normalized to the required *physical memory* is $\ln 2$, i.e., the classical Willshaw model exploits about 70% of the physical memory required for representing the memory matrix. So far it has been believed that it would be impossible to reach capacity 1 (or at least exceed $\ln 2$) for truly distributed storage [134].

One surprising result of this work is that asymptotically *storage capacity 1* (normalized to the required physical memory) is possible for the Willshaw model if the binary memory matrix is optimally compressed by applying Huffman or Golomb coding (see section 3.2). While for classical parameters ($k = \mathrm{ld}\, n$) the memory matrix is incompressible since the memory load (probability of an active synapse) is $p_1 = 0.5$, it turns out that storage capacity 1 is possible for *all* sub-linear but non-logarithmic pattern activities $k(n)$ corresponding to sparse memory matrices with $p_1 \to 0$ (sub-logarithmic k) or $p_1 \to 1$ (super-logarithmic k). Further, it turns out that even non-optimal matrix compression (e.g., simply storing the indices of sparse matrix entries) can yield storage capacity 1 (although only for parameters where $\leq n$ patterns can be stored).

With respect to *fault tolerance* a very simple and general result has been derived: Requiring fault tolerance against $(1 - \lambda)k$ missing ones in the address pattern (i.e., using only a fraction $\lambda \in (0; 1]$ of the k ones) decreases the storage capacity by a factor of about λ (see section 3.3.1). For example, if we want to use address patterns where half of the original ones are missing this would decrease the asymptotic storage capacity for optimal matrix compression from 1 to 0.5.

It has been shown that using efficiently Golomb coding (see appendix A.3.2) for optimal matrix compression not only increases the storage capacity, but generally even decreases the time t_{ret} required for a retrieval, at least for sequential implementations (see section 3.4). Indeed for *sequential* implementations it turned out that matrix compression improves not only the *absolute* retrieval time t_{ret} but also the information flow f_I which normalizes to the amount of retrieved information, and the retrieval efficiency E which additionally normalizes to the total amount C_A of stored information. Although quite efficient *parallel* implementations for matrix compression are possible (see section 3.5.2), it turns out that parallel implementations of the classical Willshaw model *without* matrix compression remain superior with respect to t_{ret}, f_I, and E.

The classical analysis of the Willshaw model is based on a binomial approximation of the error probabilities. It is assumed that the active synapses are generated independently, and that for addressing with z one entries the membrane potential X_R (number of synaptic inputs) of an unrelated (non-addressed) neuron can be described by a binomial random variable $X_R \sim B_{z,p_1}$. Actually the synapses are not independent since storing one pattern (or pattern pair) generates k^2 synapses at a time. The true potential distribution has previously been described in [29] and is denoted in this work as the *Willshaw distribution* $W_{z,n,k,M}$ where M is the number of stored patterns (see section 3.6). We have seen (see Fig. 3.5) that the binomial approximation can differ quantitatively and qualitatively from the Willshaw distribution even for very large n. In particular the true Willshaw distribution of membrane potentials can exhibit oscillatory modulations and a much larger variance than the corresponding binomial approximation. This has severe consequences for the classical analysis which therefore overestimates storage capacity and fault tolerance for finite neuron numbers n. In sections 3.6.2 and 3.6.3 it has been analyzed for conditions where the binomial approximation is justified.

Chapter 4 investigated the use of spiking neurons for associative memory. Simple considerations about the separation of overlaying patterns led to the development of a *spike counter model* implementing a time-continuous version of the so-called Willshaw retrieval strategy of the classical Willshaw model (see section 4.3; cf. section 3.1.1) . The spike counter model is superior to the classical model since it requires for a good retrieval only that the *first spikes* are correct (cf. [183]). In contrast one-step retrieval in the classical model requires for a good retrieval that the less excited addressed neurons must still have a larger potential than the strongest excited non-addressed (false) neurons (Fig. 4.1). In the spike counter model the first spikes determine the retrieval result (Fig. 4.2). If the first spikes belong to the (correct) addressed neurons instantaneous feedback will pop-out the correct pattern and suppress the false (non-addressed) neurons.

This effect works best for a low matrix load $p_1 \leq 0.5$ when addressing with a superposition of several overlayed patterns (see section 4.4.4). Here the classical one-step retrieval usually yields also a superposition of many patterns whereas the spike counter model can segregate the individual components contained in the address pattern.

In contrast to the classical one-step retrieval the spike counter model can be seen as a clique-detector where the patterns correspond to cliques if the one-entries in the memory matrix are interpreted as the edges of a graph (in the auto-associative case). Thus, by choosing appropriate parameters (internal feedback much stronger than external input; balanced excitation and inhibition) the spike counter model supports the storage of patterns with different pattern activities k and the autonomous detection of failed retrievals (see section 4.3).

In section 4.4 the spike counter model has been analyzed with respect to fault tolerance against false one-entries in the address patterns. Analysis using the classical binomial approximation technique suggests a surprisingly high fault tolerance, in particular for ultra-sparse patterns with sub-logarithmic $k(n)$ (corresponding to $p_1 \to 0$). Even if the memory matrix is filled up according to high-fidelity with $\lambda = 1$ for classical one-step retrieval (see section 4.4.2), i.e., without requiring fault tolerance against missing ones, we can expect good retrieval even for $\kappa \cdot k$ false ones where κ can be polynomial in $\ln n$ (cf. eq. 4.34). Since this analysis relies on the binomial approximation the fault tolerance is certainly overestimated for finite n. However, the analysis is asymptotically correct for sublinear $k(n)$. For finite n the analysis has been repeated in section 4.4.5 using the true Willshaw distribution and the refined analysis technique derived in section 3.6. This resulted in an implicit equation for the maximal possible κ. The true values are smaller than the estimates of the binomial approximation but for ultra-sparse patterns with $k < \mathrm{ld}\, n$ still very large values $\kappa \gg 1$ are obtained. In section 4.5.4 these theoretical results are compared to numerical simulations of classical one-step retrieval and the spike counter model.

In section 4.5 it is shown that the spike counter model can be implemented efficiently (see Table 4.1). For the classical optimum $k = \mathrm{ld}\, n$ both classical one-step retrieval and the spike counter model require $\sim n \mathrm{ld}\, n$ steps for a sequential implementation. However, for a parallel implementation the spike counter model requires already $\sim (\mathrm{ld}\, n)^2$ steps while classical one-step retrieval needs only $\mathrm{ld}\, n$ steps.

There is a remarkable coincidence of arguments favoring *spiking* associative memory, sparse synaptic connectivity with $p_1 \to 0$ corresponding to ultra-sparse patterns with sub-logarithmic $k(n)$, and optimal matrix compression using Golomb coding. (1) For ultra-sparse patterns with $p_1 \to 0$ storage capacity 1 is approached most quickly, i.e., the largest storage capacities can be obtained for finite n (see Fig. 3.2d). (2) For ultra-sparse patterns still a very large number of patterns can be stored (e.g., $\sim n^{2-1/k}$ for constant k). This is almost as much as for the classical optimum $k = c \mathrm{ld}\, n$ (cf. Table 3.1). (3) The time required for a retrieval is particularly short for ultra-sparse patterns. The efficiency measures t_{ret}, f_I, and E are better for smaller k (see sections 3.4 and 3.5). Note for example from eq. 3.57 that for constant k a retrieval can be implemented sequentially in sub-linear time. (4) The fault tolerance of spiking associative memory against false one-entries in the address pattern is particularly good for ultra-sparse patterns (see eq. 4.34). And correspondingly, (5) spiking associative memory enables best pattern separation for small $p_1 \to 0$ as obtained for ultra-sparse patterns (cf. section 4.4.4).

7.2 Contributions to the theory of cell assemblies

The theory of Willshaw associative memory in **chapters 3 and 4** is closely related to the postulation of local cell assemblies in a small patch of cortex. The binary memory matrix of Willshaw associative memory can be seen as an idealized model of the neural connectivity below perhaps a square millimeter of cortex surface (see section 2.3.3). Thus the classical results concerning sparseness, storage capacity, and number of storable patterns apply at least qualitatively to local cortical networks if they actually contain local assemblies.

Interestingly, the results of this work suggest a method to test the postulate of local cell assemblies in a neurophysiological experiment measuring the variance $\mathrm{Var}(X_R)$ of the postsynaptic potentials X_R of a given neuron when unspecifically stimulating a large number z of neurons (see section 3.6.5). The experiment is based on results in section 3.6.3 where it is shown that $\mathrm{Var}(X_R) \sim z^2$ (see eq. 3.122) if the active synapses are generated according to learning of local Hebbian cell assemblies (corresponding to the Willshaw distribution), while $\mathrm{Var}(X_R) \sim z$ (cf. eq. 3.123) if the synapses are generated independently (corresponding to the binomial approximation). For large z the difference should be large enough to be measurable in neurophysiological experiments even if we take into account additional sources of variance (like different synaptic strengths, incomplete connections, delays, noise, etc.). It may be even possible to estimate the mean size k of a local assembly at a given cortical location (see eq. 3.128).

In section 4.6 a biological version of spiking associative memory has been derived from the spike counter model (as a continuous version of the classical Willshaw model) using a more realistic spiking neuron model, and incorporating principles of local cortical architecture (see section 2.1.3). This model gives a plausible role to excitatory and inhibitory conductances comparable to the counters of the simpler spike counter model. The model consists of three neuron populations, one of them excitatory, and two inhibitory. Different versions of the model have been investigated with respect to retrieval quality and separation of overlaid patterns. In order to allow a performance comparable to the technical spike counter model it turned out that the relevant property of the model is a balanced activation of excitatory and inhibitory conductances. This requires that at least a subgroup of the inhibitory neurons (i.e., one of the two inhibitory populations, the so-called “separating inhibition”, see section 4.6; cf. Fig. 4.15) is active in synchrony with the excitatory neurons. This can be achieved by inhibitory neurons receiving extra-areal (hetero-associative) address input similar to the excitatory population. In contrast, previously suggested simpler models [199], where inhibitory neurons receive only local excitation, are generally not robust against realistic transmission delays.

With realistic neuron parameters (see section 2.4; see Table 2.1) a retrieval occurs within 5 msec (see Fig. 4.16). This means that the spikes of a local assembly are synchronized within a time window of about 5 msec. When addressing with a superposition of several overlayed patterns the first spikes determine which assembly gets activated. I.e., similar to the technical spike counter model the delayed feedback of the first spike cause a pop-out of the corresponding assembly, while other assemblies are suppressed. This demonstrates that subtle temporal structure in neural signals such as spike synchronization on a millisecond-time-scale can guide information processing in local cortical networks, despite the relatively large membrane time constants of the neurons. However, our model did not exhibit phase coding as described in other work using Hopfield architectures (e.g., [150]) involving sequential activation within even smaller time windows of all assemblies addressed by the input superposition.

In **chapter 5** a model of two reciprocally connected visual cortical areas has been proposed in order to investigate on a more global level, binding, synchronization and the temporal correlation hypothesis (TCH). The lower area is modeled similar to the orientation selective subsystem of primary visual cortex V1, the other more central visual area using the biological variant of spiking associative memory as described in chapter 4. In the lower area objects in the visual field are represented by distributed activity patches corresponding to local orientations, while the higher area contains Hebbian learned cell assemblies representing the objects. The two areas are connected by reciprocal connections associating the representations in the higher area with certain patterns of activated patches in the lower area corresponding to the objects.

The model has been simulated using either a single stimulus or a superposition of several stimuli. The simulations for a single stimulus showed that the feedback from the higher area can induce fast oscillations in the gamma range (30-60Hz), and increase the range of synchronization. For multiple stimuli the overlaying patterns are separated in the associative memory similarly as described in chapter 4. In contrast to models with only uni-directional connections (e.g. [199]) the reciprocal connections induce stable attentional activation states where corresponding assemblies representing the same one object get activated in the two areas while the assemblies representing the remaining objects are suppressed. Due to habituation of the excitatory neurons the attentional state lasts only tens to hundreds of milliseconds and then switches to the next assembly. Correspondingly, for neurons representing only one of the objects two activation states can be observed: The attentional or fast state was accompanied with stronger spike activity and fast oscillations synchronized over longer distances, while the unattended or slow state exhibited weaker activity, slower oscillations, and a confined synchronization range.

These observations can be interpreted as *binding* of corresponding representations within and between areas: The distributed feature representations in the lower area were bound to the object representation in the higher area by synchronization of spike activity similar as suggested by the temporal correlation hypothesis. This process is also related to phase coding as suggested in previous models (e.g., [150]) but differs with respect to several important aspects. Most importantly, binding occurs on a slightly larger time scale than suggested by the strong version of the TCH. Common phase-coding models [150] assume that within one gamma period (15-30msec) all subassemblies representing the scene are activated separately in a sequence. Furthermore, phase coding models usually assume that the order in the sequence does not (or only slowly) change between different gamma cycles. In contrast, in the model proposed in this work already the temporal ignition of the fast state along the cortico-cortical connections lasts 15-30 msec (or even longer), the duration of the fast state of one assembly can be about 50-500 msec, and the representation of a whole scene would require perhaps 100msec-1sec or even the whole time window of conscious experience as postulated in [139] (about 3 sec).

In order to further investigate synchronization on different time scales also a larger variant of the model has been simulated where also the higher area was modeled as a *topographical* associative memory, i.e., *all* the connections within and between the two areas were only local. Stimulating with elongated bars we found that zero-phase synchronization on the fine time scale corresponding to the gamma range can result from averaging over a larger time window when computing spike correlograms, an effect which occurs apparently also in neurophysiological experiments [41, 52, 42]. This means that zero-phase correlograms with central (un-lagged) peaks do not necessarily indicate that the neurons are really synchronized within a time window required by phase-coding. Actually the neurons fire most of the time in asynchrony, and only when evaluating a larger time window this averages out to zero phase. Thus a global cell assembly distributed over the cortex cannot be expected to be globally synchronized within a time window of 1-5msec.

The results of this work rather suggest that the relevant time window for global binding by synchronization is at least one order of magnitude longer than suggested by the strong version of the TCH, perhaps 50-500msec. Common phase-coding models [150] also contradict even elementary neurophysiological experiments. For example, they would rather predict shifted peaks in correlograms of neurons representing different entities, while the experimental correlograms remain flat [60, 44, 168, 108]. In contrast our model seems to be consistent with various phenomena observed in neurophysiological experiments (see section 5.3) such as synchronization on a fast time scale [40, 168] or a slow time scale [123], ongoing activity in optical recordings [190], two-state fluctuations of membrane potentials [10], and attentional or biased competition [147].

In **chapter 6** binding problems (BP) of different levels of difficulty are classified and discussed in the light of the simulation experiments presented in this work. Binding within a local cortical area (BP0) can occur due to local synaptic auto-associative connections as described and analyzed in chapters 3 and 4 within a time window of $1-10$ msec. Binding of corresponding assemblies in different areas (BP1) can occur due to the reciprocal connections between the areas. If two areas are directly connected binding of the most relevant assemblies requires perhaps $10-30$ msec, but binding of multiple competing representations can require several hundred milliseconds. Binding across different processing streams (BP2) corresponding to independent feature dimensions (like form, color, or location) can occur indirectly via primary topographically organized areas where the different streams are still represented at the same cortical location. The presented model even suggests solutions for some forms of the dynamic binding problems (BP3) requiring neither fast synaptic plasticity nor sustained activity. There are also more difficult binding problems beyond BP3 which certainly require further assumptions about the functional architecture of the brain (e.g., multiple instantiations, etc.; cf. [165]). How well the suggested solutions for BP2 and BP3 work remains to be seen in future studies.

Chapter 8

Zusammenfassung (in German)

Diese Arbeit leistet hauptsächlich Beiträge zur Theorie der *Neuronalen Assoziativspeicher* [174, 204, 104, 129, 69, 105, 134, 29, 160, 171] und zur Theorie der *Neuronalen Assemblies* des Gehirns [64, 26, 130, 192, 133]. Die erzielten Ergebnisse haben eine Relevanz einerseits für technische Anwendungen im Bereich Information Retrieval, aber auch für die Hirnforschung und für die neurowissenschaftliche Künstliche Intelligenz.

Hauptergebnisse mit Relevanz für Information Retrieval

Neuronale Assoziativspeicher liefern effiziente Methoden für *Information Retrieval*, d.h. für den assoziativen Zugriff auf große Mengen in Form von Mustervektoren gespeicherter Information. Insbesondere von dem von Steinbuch [174] vorgeschlagenen und später von Willshaw et al. [204] und Palm [129] analysierten klassischen binären assoziativen Netzwerk, welches im folgenden als *Willshaw-Modell* bezeichnet wird, läßt sich ein Algorithmus ableiten, der in technischen Anwendungen einen schnellen und fehlertoleranten Zugriff auf Information erlaubt, und der klassischen in der Informatik eingesetzten Verfahren (etwa Suchbäumen, Look-Up Tabellen oder Hash-Tabellen) überlegen sein kann. In Kapitel 3 und 4 wurde die klassische **Theorie der Willshaw-Assoziativspeicher** in drei Richtungen erweitert: (1) Komprimierung der Speichermatrix, (2) Verwendung von spikenden Neuronen, und (3) Verbesserung der Analysetechnik.

Für das Willshaw-Modell ist seit langem bekannt, daß für spärliche Muster mit $k = \mathrm{ld}\, n$ (wobei n die Länge des binären Mustervektors und k die Anzahl der Einsen ist) asymptotisch eine hohe Speicherkapazität von $\ln 2 \approx 0.7$ erreichbar ist, was ein viel höherer Wert als bei alternativen, etwa Hopfield-artigen Modellen ist. Bisher wurde angenommen, daß es für echt verteilte Speicherung unmöglich sei, Speicherkapazität 1 zu erreichen oder wenigsten $\ln 2$ zu übertreffen.

Ein überraschendes Resultat dieser Arbeit ist aber, daß **Speicherkapazität 1** (normalisiert zum belegten physikalischen Speicher) asymptotisch tatsächlich möglich ist, wenn man die Speichermatrix auf optimale Weise komprimiert, etwa durch Anwendung einer Huffman- oder Golomb-Kodierung (siehe Abschnitt 3.2). Obwohl für klassich logarithmische Musteraktivität $k = \mathrm{ld}\, n$ die Speichermatrix nicht komprimierbar ist da hier die Matrixbeladung $p_1 = 0.5$ ist, ergibt sich Speicherkapazität 1 für *alle anderen* sub-linearen $k(n)$ wo entweder $p_1 \to 0$ oder $p_1 \to 1$.

Diese Befunde ergeben vor allem für technische Anwendungen des Willshaw-Modells eine Reihe von Vorteilen (aber auch für die Simulation biologischer Modelle). (1) Der physikalische Speicher wird viel besser ausgenutzt. (2) Für *alle* sub-linearen $k(n)$ (und nicht nur für $k = \mathrm{ld}\, n$) kann man hohe Speicherkapazität erreichen, was das Finden von geeigneten Kodierungen erleichtert. (3) Es ergibt sich ein sehr einfacher Zusammenhang zwischen fehlenden Adreß-Einsen und der Speicherkapazität: Wenn man mit nur $\lambda \cdot k$ der Einsen adressiert ($\lambda \in [0; 1]$) so verringert sich die Speicherkapazität um Faktor λ. (4) Matrixkomprimierung mit einer Golomb-Kodierung verbessert nicht nur die Speicherkapazität, sondern für sequentielle Implementierungen auch die Auslesegeschwindigkeit (selbst wenn man zur ausgelesenen oder insgesamt gespeicherten Information normalisiert). Auch für parallele Architekturen existieren effiziente Implementierungen, obwohl hier das klassische unkomprimierte Modell überlegen bleibt.

In Kapitel 4 wurde der Nutzen **spikender Neurone** für Assoziativspeicher untersucht. Einfache Überlegungen über die Trennung überlagerter Muster führten zur Entwicklung, Analyse und Implementierung des sogenannten **Spike-Zähler-Modells** (*spike counter model*) als einer zeitkontinuierlichen Variante des Willshaw-Modells. Das Spike-Zähler-Modell bietet Vorteile, da für ein gutes Ausleseergebnis nur die Korrektheit der Neurone mit den *höchsten* Potentialen (die zuerst feuern) verlangt wird. In diesem Fall wird die instantane Rückkopplung zu einem Pop-Out des richtigen Musters und gleichzeitig zu einer Unterdrückung der falschen (nicht adressierten) Muster führen. Dies funktioniert am besten für geringe Matrixbeladung $p_1 \leq 0.5$ wenn mit einer Überlagerung mehrerer Muster adressiert wird. Im Vergleich zum klassischen Einschrittverfahren bietet das Spike-Zähler-Modell eine Reihe von weiteren Vorteilen: (1) Es lassen sich Muster mit unterschiedlicher Musteraktivtäten k speichern. (2) Fehlerhaft verlaufende Auslesevorgänge können frühzeitig erkannt und abgebrochen werden. (3) Insbesondere für ultra-spärliche Muster mit sub-logarithmischem $k(n)$ und $p_1 \to 0$ erhält man eine hohe Fehlertoleranz gegen $\kappa \cdot k$ falsche Einsen im Adreßmuster, wobei κ polynomiell in $\ln n$ wachsen darf (Gl. 4.34). (4) Das Spike-Zähler-Modell kann sequentiell und parallel effizient implementiert werden, auch in Kombination mit Matrixkompression.

Die bisherige Analyse des Willshaw-Modells beruht auf einer Binomialapproximation der neuronalen Potentialverteilung von der man u.a. die Fehlerwahrscheinlichkeit p_{01} (Gl. 3.6) ableitet. In dieser Arbeit wurde gezeigt, daß diese Approximation der sogenannten **Willshaw-Verteilung** unter Umständen sehr schlecht sein kann (Abschnitt 3.6). Die tatsächliche Willshaw-Potentialverteilung kann oszillatorische Modulationen und eine viel größere Varianz als die Binomialapproximation aufweisen, was zu einer Überschätzung von Speicherkapazität und Fehlertoleranz führen kann (Fig. 3.5). Für die Varianz konnte eine Näherungsformel hergeleitet werden (Gl. 3.122), mit deren Hilfe die Fehlertoleranz des klassischen Willshaw-Modell und des Spike-Zähler-Modell genauer vorhergesagt werden kann (siehe Abschnitt 4.4.5).

Hauptergebnisse mit Relevanz zur Hirnforschung

Das Willshaw-Modell steht in enger Beziehung zur **Assembly-Theorie** der Gehirnforschung, da sich neuronale Assoziativspeicher auch als Modell für lokale Netzwerke in der Großhirnrinde interpretieren lassen [27]. Hierbei können die autoassoziativ gespeicherten Muster mit Hebb'sch gelernten Zellen-Assemblies identifiziert werden, d.h. mit diskreten Neurongruppen von denen angenommen wird, daß sie der Repräsentation interner oder externer Entitäten im Gehirn dienen (etwa von Gedanken oder Objekten). Leider konnten die postulierten Zellen-Assemblies noch nicht im Gehirn nachgewiesen werden, da man für einen direkten Nachweis eine sehr große Zahl Neurone gleichzeitig ableiten müßte, was technisch sehr schwierig ist.

Interessanterweise kann man aber aus der eben beschriebenen Diskrepanz zwischen Willshaw-Verteilung und Binomialapproximation eine Methode für die Hirnforschung ableiten, wie man die **Hypothese lokaler Assemblies in einem neurophysiologischen Experiment überprüfen** könnte (siehe Abschnitt 3.6.5). Das Experiment beruht darauf, daß Binomialapproximation und tatsächliche Willshaw-Verteilung bei zufälliger Stimulierung von z Adreßneuronen unterschiedlich große Varianzen in den Membranpotentialen vorhersagen. Bei unabhängig generierten Synapsen (entsprechend der Binomialapproximation) würde die Varianz nur *linear* in z wachsen, während sie bei Hebb' sch gelernten Synapsen (entsprechend der Willshaw-Verteilung) *quadratisch* in z wachsen müßte. Weiter erscheint es möglich, die mittlere Größe k einer lokalen Assembly in einer gegebenen kortikalen Region abzuschätzen (Gl. 3.128).

Ein weiteres Problem der Hirnforschung (aber auch der neurowissenschaftlichen Künstlichen Intelligenz [130]) ist das sogenannte Bindungsproblem, d.h. zu erklären wie aus der Aktivität über das ganze Hirn verteilter Zellgruppen etwa kohärente Wahrnehmungen erzeugt werden. Als mögliche Lösung wurde die sogenannten zeitlichen Korrelationshypothese (*temporal correlation hypothesis*, TCH; [192, 168]) vorgeschlagen. Ein wichtiges Postulat der TCH ist etwa, daß Bindung im Gehirn durch eine globale Spikesynchronisation der die zu bindenden Entitäten kodierenden Neurone erreicht wird (z.B. die Merkmale eines gemeinsamen Objektes).

Um Bindung und Synchronisation in *lokalen* kortikalen Netzwerken zu untersuchen wurde in Abschnitt 4.6 vom Spike-Zähler-Modell unter Verwendung realistischerer Neurone ein **biologisches Modell eines spikenden Assoziativspeichers** abgeleitet. Unterschiedliche Varianten des Modells wurden in Bezug auf Auslesequalität und Separation überlagerter Muster untersucht, wobei eine Performanz vergleichbar mit der des technischen Spike-Zähler-Modells erreicht werden konnte. Mit realistischen Neuronparametern (Tab. 2.1) dauert ein Auslesevorgang ungefähr 5 msec. Wie beim Spike-Zähler-Modell verursacht die verzögerte Rückkopplung der ersten Spikes einen "Pop-Out" der entsprechenden Assembly, während andere Assemblies unterdrückt werden. Diese Ergebnisse zeigen wie feine zeitliche Struktur in neuronalen Signalen die Informationsverarbeitung in lokalen kortikalen Netzwerken trotz relativ großer neuronaler Membran-Zeitkonstanten steuern kann.

In Kapitel 5 wurde ein Modell zweier **reziprok verbundener visueller kortikaler Areale** entwickelt um auf einer *globaleren* Ebene **Bindung, Synchronisation und Vorhersagen der TCH** zu untersuchen. Das niedrigere Areal wurde ähnlich zum orientierungsselektiven Subsystem des primären visuellen Kortex V1 modelliert, das andere eher zentrale visuelle Areal als die biologische Variante eines spikenden Assoziativspeichers. Im primären Areal werden Objekte im visuellen Feld durch lokalen Orientierungen entsprechenden verteilte Aktivitätsflecken repräsentiert, im zentralen Areal durch Hebb'sch gelernte Assemblies.

Simulationen ergaben, daß die Rückkopplung vom zentralen zum primären Areal schnelle Oszillationen im Gamma-Bereich (30-60Hz) induzieren und den Synchronisationsbereich der neuronalen Aktivität vergrößern kann. Bei mehreren Stimuli werden überlagerte Muster ähnlich wie in den spikenden Assoziativspeichern separiert. Die reziproken Verbindungen verursachen stabile "attentive" Aktivierungszustände, wobei in den beiden Arealen entsprechende Sub-Assemblies aktiviert werden, und die anderen Assemblies unterdrückt werden. Wegen Habituation der exzitatorischen Neurone halten diese attentiven Zustände nur einige zehn bis einige hundert Millisekunden lang an und schalten dann auf eine andere Assembly um.

Die beschriebenen Beobachtungen konnten als *Bindung* zusammengehöriger Repräsentationen in und zwischen kortikalen Arealen interpretiert werden. Im Unterschied zu Vorhersagen der starken TCH (z.B. Phasenkodierungen wie in [150]) findet *globale* Bindung auf einer gröberen Zeitskala statt. Tatsächlich ergibt sich Synchronisation zwischen weit entfernten Neuronen nur aus der Mittelung über größere Zeitfenster (vielleicht 50-500 msec). Die Repräsentation einer ganzen Szene (mit mehreren Objekten) braucht vielleicht 100 msec-1 sec, oder sogar das ganze Zeitfenster der bewußten Wahrnehmung (etwa 3 sec; [139]). Im Gegensatz zu alternativen Arbeiten scheint das in dieser Arbeit untersuchte Modell aber mit vielen neurophysiologisch beobachtbaren Phänomenen konsistent zu sein (etwa [40, 168, 123, 10, 147]).

In Kapitel 6 wurden **verschiedene Bindungsprobleme** unterschiedlicher Schwierigkeitsgrade im Zusammenhang mit den in dieser Arbeit beschriebenen Simulationen diskutiert und klassifiziert. Während die durchgeführten Simulationen für einige einfachere Bindungsprobleme schon plausible Lösungen darstellen, konnten die Ergebnisse auch hinsichtlich schwierigerer Bindungsprobleme (z.B. dem der Rollen-Zuteilung) extrapoliert werden. Insbesondere erscheinen für einige dynamischen Bindungungsprobleme neue Lösungsmöglichkeiten, welche keine schnelle synaptische Plastizität erfordern.

Appendix

Appendix A

Information theory

A.1 Basic information theory

A.1.1 Information of random variables

Let X be a random variable on $\Omega = \{\omega_1, \omega_2, ...\}$, and $p_i := \mathrm{pr}[X = \omega_i]$ the probability that X has value $\omega_i \in \Omega$. Then the **information** of X is defined as

$$I(X) = \sum_{i\in\Omega} -p_i \mathrm{ld}\, p_i. \tag{A.1}$$

$I(X)$ can be thought of as the minimal number of yes/no-questions that are necessary on average to guess the actual value of X, or, in other words, the mean codeword length (in bit) of an optimal code for outcomes of X (cf. section A.3.1).

Let Y be a second random variable on Ω, and $p_{i\wedge j} := \mathrm{pr}[X = \omega_i \wedge Y = \omega_j]$ the compound probability of X and Y, and $p_{ij} := \mathrm{pr}[X = \omega_i | Y = \omega_j]$ the conditional probability of X given Y. Then we can similarly define the **conditional information** $I(X|Y)$ of X given Y as

$$I(X|Y) = \sum_{i,j\in\Omega} -p_{i\wedge j} \mathrm{ld}\, p_{ij}, \tag{A.2}$$

which can be interpreted as the minimal number of yes/no-questions to guess X, *if Y is already known*. Correspondingly, it can be shown that

$$I(X, Y) = I(Y) + I(X|Y) \leq I(X) + I(Y), \tag{A.3}$$

where equality holds if X and Y are independent. This relation is also called the *subadditivity of information*.

A.1.2 Transinformation and transinformation rate

The **transinformation** or **mutual information** between two random variables X and Y is defined as

$$T(X; Y) = I(X) + I(Y) - I(X, Y) = I(X) - I(X|Y). \tag{A.4}$$

and can be thought of as the information contained in Y about X (or vice versa), or correspondingly, the number of yes/no-questions that can be *saved* due to knowledge of Y when guessing X.

Furthermore, we define the **transinformation rate** between two processes $(X^\mu)_{\mu=1,2,...}$ and $(Y^\mu)_{\mu=1,2,...}$ as

$$T((X^\mu)_\mu; (Y^\mu)_\mu) = \lim_{M\to\infty} \frac{1}{M} \cdot T(X^1, ..., X^M; Y^1, ..., Y^M). \tag{A.5}$$

A.1.3 Channels and channel capacity

The information transmission from a stationary source process $(X^\mu)_{\mu=1,2,\ldots}$ to a target $(Y^\mu)_{\mu=1,2,\ldots}$ can be described by a **channel** which can be identified with a **transition probability** $P : \Omega^{\mathbb{N}} \hookrightarrow \Omega^{\mathbb{N}}$ assigning for each outcome of $(X^\mu)_\mu$ a probability measure for $(Y^\mu)_\mu$ on $\Omega^{\mathbb{N}}$. If Ω is finite and P is memory-free (i.e. the outcome of Y^μ depends only on X^μ), then P can simply be identified with a matrix of transition probabilities

$$p_{ij} = \mathrm{pr}[Y^\mu = j | X^\mu = i]. \tag{A.6}$$

Finally we define the **channel capacity** C_P for a channel $P : \Omega^{\mathbb{N}} \hookrightarrow \Omega^{\mathbb{N}}$ as the maximal achievable transinformation rate when varying the statistics of the source process $(X^\mu)_{\mu=1,2,\ldots}$. For our purposes it is sufficient to consider memory-free channels where the components X^μ of the source process are i.i.d. (identically and independently distributed) according to a probability measure p_X. Then we can simply write

$$C_P = \sup_{p_X} T(X;Y). \tag{A.7}$$

A.2 Binary channels

In the following we consider the binary case $\Omega = \{0,1\}$. For a random variable X on Ω with $p_1 := \mathrm{pr}[X = 1]$ the information I(X) equals

$$I(p_1) := -p_1 \cdot \mathrm{ld}\, p_1 - (1-p_1) \cdot \mathrm{ld}\,(1-p_1) \tag{A.8}$$

$$\approx \begin{cases} -p_1 \cdot \mathrm{ld}\, p_1 & , \quad p_1 \ll 0.5 \\ -(1-p_1) \cdot \mathrm{ld}\,(1-p_1)\,, & 1-p_1 \ll 0.5 \end{cases} \tag{A.9}$$

Note the symmetry $I(p_1) = I(1-p_1)$, and that $I(p_1) \to 0$ for $p_1 \to 0$ (and $p_1 \to 1$). Occasionally we will need also the derivative of $I(p_1)$,

$$I'(p_1) := \frac{dI(p_1)}{dp_1} = -\mathrm{ld}\, p_1 + \mathrm{ld}\,(1-p_1). \tag{A.10}$$

A binary channel without memory is already determined by the two error probabilities p_{01} (false one) and p_{10} (false zero). For two binary random variables X and Y where Y is the result of transmitting X over a binary channel we can write

$$I(Y) = I_Y(p_1, p_{01}, p_{10}) := I\,(p_1\,(1-p_{10}) + (1-p_1)\,p_{01}) \tag{A.11}$$

$$I(Y|X) = I_{Y|X}(p_1, p_{01}, p_{10}) := p_1 \cdot I(p_{10}) + (1-p_1) \cdot I(p_{01}) \tag{A.12}$$

$$T(X;Y) = T(p_1, p_{01}, p_{10}) := I_Y(p_1, p_{01}, p_{10}) - I_{Y|X}(p_1, p_{01}, p_{10}). \tag{A.13}$$

For the analysis of Willshaw associative memory we are especially interested in the case $p_{10} = 0$ and small $p_{01} \ll 0.5$ (cf. appendix B.1). Applying linearization of $I(p_1)$ in p_1 we obtain

$$T(p_1, p_{01}, 0) = I\,(p_1 + (1-p_1)p_{01}) - (1-p_1) \cdot I(p_{01}) \tag{A.14}$$

$$\approx I(p_1) + I'(p_1) \cdot (1-p_1) \cdot p_{01} - (1-p_1) \cdot I(p_{01}) \tag{A.15}$$

$$\approx (1-e_I) \cdot I(p_1). \tag{A.16}$$

For the relative error e_I of approximating $T(p_1, p_{01}, p_{10})$ by $I(p_1)$ we can write

$$e_I := \frac{I(p_1) - T(p_1, p_{01}, p_{10})}{I(p_1)} \tag{A.17}$$

$$\approx (1-p_1)\frac{I(p_{01}) - I'(p_1) \cdot p_{01}}{I(p_1)}. \tag{A.18}$$

If we additionally assume $p_1 \ll 0.5$ we have approximately $1 - p_1 \approx 1$, $I(p_1) \approx -p_1 \mathrm{ld}\, p_1$ (see eq.A.9), and $I'(p_1) \approx -\mathrm{ld}\, p_1$ (see eq. A.10). Thus we obtain

$$e_I \approx \frac{I(p_{01})}{I(p_1)} - \frac{p_{01}}{p_1}. \tag{A.19}$$

A.3 Optimal compression of sparse bit streams

In chapter 3 we investigate the Willshaw model of associative memory. Here, information about pattern pairs is stored in a binary matrix. We find that it is advantageous if sparse matrices are compressed optimally with respect to information theory. Optimal compression of a bit stream (e.g., a row of the Willshaw memory matrix) can be achieved by Huffman or Golomb coding [74, 57, 164, 34]. These strategies are described in the following in more detail.

A.3.1 Huffman codes

As mentioned in section A.1.1 it is known that the expected length l of any binary code for a random variable $X \in \Omega$ is at least $I(X)$. More exactly, for any *optimal code* we have an expected length l with

$$I(X) \leq l \leq I(X) + 1. \tag{A.20}$$

Correspondingly, for coding the outcome of n i.i.d. random variables, i.e., $X = (X_1, X_2, ..., X_n)$, an optimal code will require mean length $l/n \to I(X_1)$ per symbol for large n.

Compressing sparse binary vectors (e.g., the matrix rows of the Willshaw memory matrix) can in principle be done by using Huffman codes which are known to be optimal [74, 34].

When compressing a bit stream where the sparse elements occur with probability p_1 one would proceed as follows: Divide the bit stream in compartments of length n. Then we have an alphabet Σ containing 2^n elements, i.e., all the binary vectors of length n. The probability of a vector containing exactly k ones would be $p_1^k \cdot (1 - p_1)^{n-k}$. Thus we can construct a Huffman code that yields optimal compression for this alphabet. In the ideal case the code would decrease the required physical memory by a factor of $I(p_1)$.

However, the compression factor would be limited to $1/n$ (each Huffman coded vector has at least size 1 bit). Therefore, for very sparse bit streams (e.g., $p_1 \to 0$) we would need very large n for optimal compression, and therefore also a very large (exponential size!) code tree to represent the Huffman code.

Thus, Huffman coding is not adequate for compressing very sparse bit streams. Instead we prefer Golomb codes described in the following.

A.3.2 Golomb codes

Golomb coding can be used for source coding of geometrically distributed random variables [57]. A random variable X is geometrically distributed if $\mathrm{pr}[X = k] = p_0^k \cdot p_1$ for $k \in \mathbb{N}$, $p_1 \in (0; 1)$, and $p_0 := 1 - p_1$. Thus, for a bit stream where the sparse elements are generated with probability p_1, the random variable X would code the so-called *run-length* between two sparse elements.

The basic idea of Golomb coding is to code the m most probable values of X (i.e., 1,2,..,m) by themselves (as $\mathrm{ld}\, m$-bit binary vectors) following a 0. For this m should be a power of 2. E.g., for $m = 4$, the value $k = 3$ would be coded by 011, the value $k = 1$ by 001. For the next m values the same codewords are used, but following an additional 1. E.g., for $m = 4$, the value $k = 7$ is coded by 1011, $k = 11$ is coded by 11011, and so on. Table A.1 shows the Golomb codewords for $m = 2$, $m = 4$, and $m = 8$.

Although Golomb coding is possible for any m [57], coding and decoding are especially simple if m is a power of 2: The codeword of k is k DIV m 1's followed by a 0 and the binary representation of k MOD m. Decoding is similarly simple: Start at the beginning (left end) of the word, and count the number A of 1's preceeding the first 0. Then the correct decoding of the word is $m \cdot A + R$ where $R \in \{0, 1, ..., m - 1\}$ is the number represented by the $\mathrm{ld}\, m$ bits following the first 0. Thus coding and decoding can be performed using only a constant number of additions and bit shifts. If the values of X are limited by n (which is true for practical applications such as compressing matrix rows, cf. section 3.2) then bit shifts and additions can be implemented by parallel circuitry of depth $\sim \ln \ln n$ (e.g., [33]). It may be necessary to generate (when coding)

k	codeword $m = 2$	codeword $m = 4$	codeword $m = 8$
0	00	000	0000
1	01	001	0001
2	100	010	0010
3	101	011	0011
4	1100	1000	0100
5	1101	1001	0101
6	11100	1010	0110
7	11101	1011	0111
8	111100	11000	10000
9	111101	11001	10001
10	1111100	11010	10010

Table A.1: Golomb codewords of integers $k \in \mathbb{N}$ for parameters $m = 2, 4, 8$.

and count (when decoding) up to n/m prefix 1's. However, m can usually be chosen such that the average number of prefix ones remains constant in n.

Let us assume $g := \mathrm{pr}[X \leq m] = (1 - p_1)^m$. Then the codeword of X has length $\mathrm{ld}\, m + 1 + i$ with probability $g^i \cdot (1 - g)$ where i is the number of prefix 1's. We compare the performance of Golomb coding to Huffman coding when compressing the outcome of n independent binary random variables, i.e., a binary random vector of length n, where each component is 1 independently with probability p_1 (e.g., a row of the Willshaw memory matrix, cf. section 3.2). The expected lengths for the Huffman and Golomb coded vector are l^H and l^G, respectively, and for large n we obtain

$$l^H/n = I(p_1) := -p_1 \mathrm{ld}\, p_1 - (1 - p_1)\mathrm{ld}\, (1 - p_1) \tag{A.21}$$

$$\approx -p_1 \cdot \mathrm{ld}\, p_1 \tag{A.22}$$

$$l^G/n := p_1 \cdot \sum_{i=0}^{\infty} g^i \cdot (1 - g) \cdot (\mathrm{ld}\, m + 1 + i) \tag{A.23}$$

$$= p_1 \cdot (1 - g) \cdot \left((\mathrm{ld}\, m + 1) \cdot \sum_{i=0}^{\infty} g^i + \sum_{i=0}^{\infty} i \cdot g^i \right) \tag{A.24}$$

$$= p_1 \cdot (\mathrm{ld}\, m + 1 + \frac{g}{1 - g}) \tag{A.25}$$

$$= p_1 \cdot \left(-\mathrm{ld}\, \ln \frac{1}{1 - p_1} + \mathrm{ld}\, \ln \frac{1}{g} + 1 + \frac{g}{1 - g} \right) \tag{A.26}$$

$$\approx -p_1 \mathrm{ld}\, p_1, \tag{A.27}$$

where we used $\sum_{i=0}^{\infty} g^i = 1/(1 - g)$, $\sum_{i=0}^{\infty} i g^i = g/(1 - g)^2$, and $\mathrm{ld}\, m = -\mathrm{ld}\, (-\ln(1 - p_1)/ - \ln g)$. The approximations follow from $-\ln(1 - p_1) \approx p_1$ which is valid for $p_1 \to 0$ (while g remains fixed). This shows that for $p_1 \to 0$ and any fixed $g \in (0; 1)$ (or even $0 < a < g < b < 1$ for fixed a and b) Golomb coding is asymptotically as good as Huffman coding and therefore optimal.

The minimum of the overhead term $f(g) := 1 + \mathrm{ld}\, \ln(1/g) + g/(1 - g)$ is obtained for $g_{\mathrm{opt}} \approx 0.513 \approx 1/2$ (Fig.A.1). Thus for a given p_1 we can choose a Golomb code with $m = \ln g_{\mathrm{opt}} / \ln(1 - p_1)$, or $m = 2^{[\mathrm{ld}\, (\ln(g_{\mathrm{opt}}) / \ln(1 - p_1))]}$ if we want m to be a power of 2.

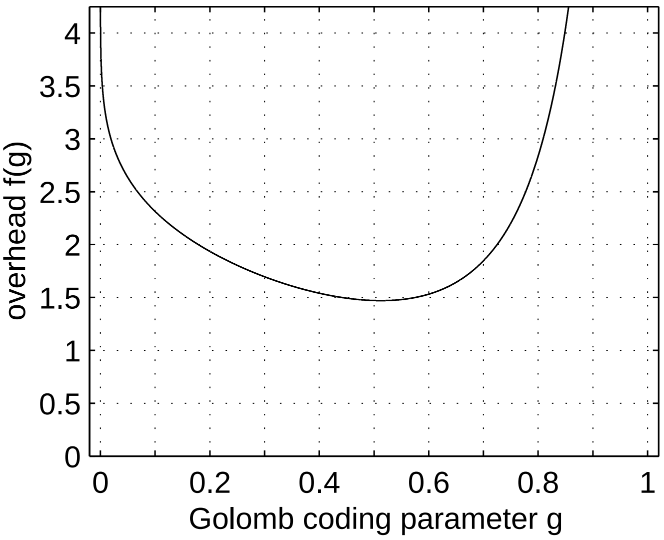


Figure A.1: Memory overhead $f(g) := 1 + \mathrm{ld}\ \ln(1/g) + g/(1-g)$ for Golomb coding with parameter m where $g = (1 - p_1)^m$ (see text). For small p_1 Golomb coding requires factor $f(g)/ - \mathrm{ld}\, p_1$ more memory than Huffman coding. The factor vanishes for $p_1 \to 0$ and fixed g (or $g \in (a; b)$ for fixed $0 < a < b < 1$). Minimal memory overhead occurs for $g \approx 0.513 \approx 0.5$.

Appendix B

Analysis of associative memory

B.1 Information-theoretical storage capacity

Associative memories are used to store and retrieve a finite number M of patterns (see chapter 3). This can be interpreted as transmitting the original source patterns $v^1, v^2, .., v^M$ over a **memory channel**, where at the target site (possibly modified) patterns $\hat{v}^1, \hat{v}^2, .., \hat{v}^M$ are received. Correspondingly, to obtain the channel capacity of eq.A.7 for given M, we have to maximize the transinformation between the source and target patterns with respect to the statistics of the source patterns. Thus we can define the *absolute* **storage capacity**,

$$C_A \quad := \quad \sup T(v^1, v^2, .., v^M; \hat{v}^1, \hat{v}^2, .., \hat{v}^M). \tag{B.1}$$

For neural associative memory it is reasonable to argue that the maximal transinformation occurs for independently generated random patterns. First, the information of the source patterns is maximal for independent random patterns. Second, if the patterns are correlated the retrieval error probabilities seem to be increased (and correspondingly the transinformation decreased). Further we can assume that the retrieval result $\hat{v}^\mu$ depends only on the corresponding original pattern v^μ. Then the storage capacity is simply

$$C_A \quad \approx \quad M \cdot \sup T(v^\mu, \hat{v}^\mu). \tag{B.2}$$

In the binary Willshaw model binary patterns $v^\mu \in \{0,1\}^n$ are stored. A further reasonable assumption is that a high storage capacity can occur only if all the patterns have about the same pattern activity, i.e., the the same number of one-entries. Thus we will assume that all patterns are binary random vectors of length n containing exactly k ones. We might further assume that $\hat{y}_i^\mu$ depends approximately only on y_i^μ, i.e., that the components of the patterns are transmitted over the channel independently of each other. Then one can apply the theory of binary channels (sect. A.2) and one obtains with eq.A.13

$$C_A(n,k,M) \quad \approx \quad M \cdot n \cdot T(k/n, p_{01}, p_{10}) \tag{B.3}$$

where $k/n := p_1$ is the fraction of ones in an original pattern, and p_{01} and p_{10} are the error probabilities of the binary channel. For the Willshaw model we can define further

$$C_A(n,k) \quad := \quad \max_M C_A(n,k,M) \tag{B.4}$$

$$C_A(n) \quad := \quad \max_k C_A(n,k). \tag{B.5}$$

It is useful to normalize C_A with respect to the number of physical memory units (usually bits) necessary to implement the associative memory. By doing so we finally obtain the *normalized* **storage capacity** C. For example, the classical Willshaw associative memory needs n^2 binary synapses and thus $C := C_A/n^2$.

As a prerequisite one usually demands low error probabilities p_{01} and p_{10} for the retrieved patterns (although larger error probabilities will increase C_A). For binary associative memory, the Willshaw retrieval strategy leads automatically to $p_{10} = 0$ (see sect. 3.1.1). By further demanding a so-called *high-fidelity requirement*

$$p_{01} \leq \epsilon \cdot \frac{k}{n} \tag{B.6}$$

for a small positive ϵ (cf. sect. 3.1.2) we obtain $T(k/n, p_{01}, p_{10}) = (1 - e_I) \cdot I(k/n)$ for a small relative error e_I (see eq. A.18)

$$e_I \leq (1 - \frac{k}{n}) \frac{I(\epsilon\frac{k}{n}) - I'(\frac{k}{n}) \cdot \epsilon\frac{k}{n}}{I(\frac{k}{n})}. \tag{B.7}$$

For sparse patterns with small $k/n \ll 0.5$ we obtain from eq. A.19

$$\begin{aligned} e_I &\leq \frac{I(\epsilon\frac{k}{n})}{I(\frac{k}{n})} - \epsilon \approx \epsilon \cdot \frac{\mathrm{ld}\, \epsilon}{\mathrm{ld}\, (\frac{k}{n})} && \text{(B.8)} \\ &\approx \frac{I(\epsilon)}{-\mathrm{ld}\, (\frac{k}{n})} && \text{(B.9)} \\ &\leq \left\{ \begin{array}{ll} I(\epsilon) \,, & \text{in any case} \\ \epsilon \quad , & \text{k/n} \leq \epsilon \end{array} \right. . && \text{(B.10)} \end{aligned}$$

Note that typically sparse patterns with $k/n \ll 1/100$ are used. Thus requiring for example $\epsilon = 0.01$ implies that the relative error is smaller than one percent. This justifies rewriting eq.B.3

$$C_A(n, k, M) \approx M \cdot n \cdot I(\frac{k}{n}) \approx M \cdot k \cdot \mathrm{ld}\, (\frac{n}{k}). \tag{B.11}$$

Note that for sufficiently small ϵ the latter approximation even underestimates C_A of eq.B.2 since then $T(v^\mu, \hat{v}^\mu) \approx I(v^\mu) = \mathrm{ld}\, \binom{n}{k} > k\mathrm{ld}\, (n/k)$.

B.2 Auto association and hetero association

In a sense auto-association and hetero-association are equivalent. Auto-association can be interpreted as a special case of hetero-association: When we have stored auto-associatively patterns $u^1, u^2, ..., u^M$ (each a binary vector of length n with exactly k ones) and perform a retrieval using an address pattern $\tilde{u}^\mu$ containing λk of the k ones of an original pattern u^μ and additionally κk false ones, then the analysis of one-step retrieval is almost identical to the analysis of hetero-association for M pattern pairs with the same parameters n, k (see chapter 3). But note that the diagonal of the memory matrix has different statistical properties for auto- and hetero-association.

On the other hand, hetero-association can be interpreted as a special case of auto-association [129] because storing hetero-associatively the association of pattern pairs $u^\mu \to v^\mu$ is equivalent to storing auto-associatively the concatenation (u^μ, v^μ), and using for the retrieval only a quarter of the memory matrix.

However, if we think of pure auto-association of patterns $u^1, u^2, .., u^M$ using the whole memory matrix we must keep in mind that an address pattern $\tilde{u}^\mu$ already contains information about u^μ. This amount of information must be subtracted from the information of the retrieval result $\hat{u}^\mu$ about the original pattern u^μ. For estimating the quality of auto-associative memories we define therefore in analogy to the storage capacity of eq.B.1 the (absolute) **completion capacity**

$$\begin{aligned} C_{C,A} &:= \sup T(u^1, u^2, .., u^M; \hat{u}^1, \hat{u}^2, .., \hat{u}^M) - T(u^1, u^2, .., u^M; \tilde{u}^1, \tilde{u}^2, .., \tilde{u}^M) && \text{(B.12)} \\ &\approx M \cdot n \cdot (T(k/n, p_{01}, p_{10}) - T(k/n, \tilde{p}_{01}, \tilde{p}_{10}))\,, && \text{(B.13)} \end{aligned}$$

where the latter approximation is for the binary Willshaw model, with k/n being the fraction of ones in a pattern, $p_{01/10}$ being the error probabilities in the retrieved patterns, and $\tilde{p}_{01/10}$ being the error probabilities in the address patterns. Note that *bidirectional* hetero-associative retrievals [107, 171] should also be analyzed using the completion capacity since retrievals initiated (for example) in the first pattern area by a pattern $\tilde{u}^{\mu}$ will result in retrieved patterns $\hat{u}^{\mu}$ and $\hat{v}^{\mu}$ in both pattern areas.

B.3 Retrieval and separation quality

B.3.1 Retrieval quality

To judge the quality of a single retrieval in the Willshaw associative memory model with n neurons where each patterns contains k ones, we can estimate the error probabilities p_{01} and p_{10} from the number of false ones and false zeros in the retrieved pattern, i.e.,

(B.14)
$$\hat{p}_{01} := \frac{\text{\#false ones}}{n-k}$$

(B.15)
$$\hat{p}_{10} := \frac{\text{\#false zeros}}{k}.$$

With eq.A.13 we can estimate the **normalized retrieval quality**

(B.16)
$$r_N := \frac{T(k/n, \hat{p}_{01}, \hat{p}_{10})}{T(k/n, 0, 0)} \in [0; 1].$$

For a perfect retrieval (without any false ones or zeros) we obtain $r_N = 1$. The advantage of r_N over the error probabilities is that we have only a single measure for the retrieval.

B.3.2 Separation quality

While the normalized retrieval quality is sufficient for the classical scenario of addressing the associative memory using a single address pattern (plus perhaps noise), we may need an additional measure if we address with a superposition involving N address patterns. Let $A_1, A_2, ..., A_N \subseteq \{1, 2, ..., n\}$ be the addressed assemblies (i.e., patterns defined as subsets of the neurons). Furthermore, let $S \subseteq \{1, 2, ..., n\}$ be the set of neurons activated by the retrieval (i.e., the ones in the retrieved pattern). For $i = 1, 2, ..., N$ let

$$\hat{A}_i := S \cap A_i$$

be the active neurons of assembly i and let

$$B := \cup_{i=1}^{n} \hat{A}_i$$

be the set of addressed neurons that are active. The *winner* of the retrieval is the addressed assembly A_w with the most active neurons, i.e.

$$w := \arg\max_{i=1}^{n} \#\hat{A}_i.$$

For the judgment of a retrieval when addressing with several patterns we define three straightforward measures. First, we define the *completeness*

(B.17)
$$v := \frac{\#\hat{A}_w}{\#A_w} \in [0; 1]$$

denoting the fraction of active units in the winner assembly. For $N = 1$ we simply have $v = 1 - \hat{p}_{10}$ (cf. eq.B.15). Second, we define the *fault value*

$$f := \frac{\#(S - B)}{\#S} \in [0; 1] \tag{B.18}$$

as the fraction of non-addressed active units, i.e., the relative number of active units that do not belong to one of the addressed assemblies. For $N = 1$ we simply have $f = p_{01}$ (cf. eq.B.14). Third, we can define the *separation*

$$s := \frac{\#\hat{A}_w}{\#B} \in [\frac{1}{N}; 1] \tag{B.19}$$

as the relative number of active addressed units that belong to the winner assembly. If all active addressed units belong to the winner assembly we have $s = 1$. If all addressed assemblies get equally activated we have $s = 1/N$. Thus we can further introduce the *normalized separation*

$$s_N := \frac{s - \frac{1}{N}}{1 - \frac{1}{N}} \in [0; 1] \tag{B.20}$$

exhibiting values between zero and one. While we still have $s_N = 1$ if all active units belong to the winner assemblies, now we have $s_N = 0$ for an equal activation of all addressed assemblies.

B.4 Further analysis of Willshaw associative memory

B.4.1 Derivatives of memory load, storable patterns, and storage capacity

From the classical binomial analysis of the Willshaw model in section 3.1.2 we obtained approximative formulae for the (maximal) memory load p_1 (eq.3.9), the (maximal) number M of storable patterns (eq.3.10), and the storage capacity C (eq.3.12),

$$p_1 = (\epsilon k/n)^{1/k} \tag{B.21}$$

$$M = -\frac{n^2}{k^2} \ln(1 - p_1) \tag{B.22}$$

$$C = -\ln(1 - p_1) \frac{\mathrm{ld}\,(n/k)}{k}. \tag{B.23}$$

For a given number of neurons n it is useful to find an optimal k in order to maximize the storage capacity or the number of storable patterns. For this we have to know the derivatives

$$p_1{}' := \frac{dp_1}{dk} = \frac{p_1}{k^2}(1 + \ln \frac{n}{\epsilon k}) \tag{B.24}$$

$$= \frac{p_1}{k^2}(1 - k \ln p_1) \tag{B.25}$$

$$M' := \frac{dM}{dk} = \frac{n^2}{k^2} \left(\frac{p_1{}'}{1 - p_1} + \frac{2 \ln(1 - p_1)}{k} \right) \tag{B.26}$$

$$C' := \frac{dC}{dk} = \frac{p_1{}'}{1 - p_1} \cdot \frac{\mathrm{ld}\,(n/k)}{k} + \ln(1 - p_1) \frac{1/\ln 2 + \mathrm{ld}\,(n/k)}{k^2}. \tag{B.27}$$

Note that these formulae are only valid if $k \ll n$, or, asymptotically, if k is sub-linear in n (see sections 3.1.2 and 3.6).

B.4.2 Optimal storage capacity for moderately sparse patterns

In section 3.2 it is shown that matrix compression can lead to storage capacity 1 asymptotically if we compress the memory matrix. In general we have a compression method with a compression factor $i_{\text{zip}}(p_1)$ such that we can improve the storage capacity from C to C^{zip}. With $i_{\text{zip}}{}' := di_{\text{zip}}/dp_1$ and the definitions of section B.4.1 we can write

$$C^{\text{zip}} := \frac{C}{i_{\text{zip}}} \tag{B.28}$$

$$C^{\text{zip}\prime} := \frac{dC^{\text{zip}}}{dk} = \frac{C' i_{\text{zip}} - C i_{\text{zip}}{}' p_1{}'}{i_{\text{zip}}{}^2}. \tag{B.29}$$

Local extrema of $C^{\text{zip}}(k)$ are obtained for $C^{\text{zip}\prime}(k) = 0$, or equivalently

$$C' i_{\text{zip}} - C i_{\text{zip}}{}' p_1{}' = 0 \quad (!). \tag{B.30}$$

As can be seen in Fig. 3.2b, both the storage capacity C^{cmpr} for optimal compression (see eq.3.15) and the storage capacity C^{sprs} for sparse matrix representation (see eq.3.14) exhibit local maxima for moderately sparse patterns with super-logarithmic, but still sub-linear pattern size $k(n)$ (see section 3.2). In the following we compute the corresponding optimal pattern activities $k_{\text{opt}}^{\text{cmpr}}$ and $k_{\text{opt}}^{\text{sprs}}$.

Sparse matrix representation

For the storage capacity C^{sprs} corresponding to sparse matrix representation (see eq.3.14) we have for moderately sparse patterns with $p_1 \to 1$ (cf. section 3.2.2)

$$i_{\text{zip}} = (1 - p_1) \ln n \tag{B.31}$$

$$i_{\text{zip}}{}' := \frac{di_{\text{zip}}}{dp_1} = -\ln n. \tag{B.32}$$

In order to compute the optimal pattern activity $k_{\text{opt}}^{\text{sprs}}$ we can write eq.B.30 with the definitions in section B.4.1 equivalently as

$$\ln \frac{n}{k} \left(\frac{k p_1{}'}{1 - p_1} (1 - \frac{1}{\ln(1 - p_1)}) - 1 \right) - 1 = 0 \quad (!). \tag{B.33}$$

With eq.3.22 we can infer the approximations

$$\frac{p_1{}'}{1 - p_1} \approx \frac{\frac{1}{k^2}(1 + \ln \frac{n}{\epsilon k})}{\frac{1}{k} \ln \frac{n}{\epsilon k}} = \frac{1}{k}(1 + \frac{1}{\ln \frac{n}{\epsilon k}}) \tag{B.34}$$

$$-\ln(1 - p_1) \approx \ln k - \ln \ln \frac{n}{\epsilon k}. \tag{B.35}$$

Inserting in eq.B.33 yields

$$\ln(n/k) \left((1 + \frac{1}{\ln \frac{n}{\epsilon k}}) \cdot (1 - \frac{1}{\ln k - \ln \ln \frac{n}{\epsilon k}}) - 1 \right) - 1 = 0 \quad (!). \tag{B.36}$$

With the substitutions

$$\lambda := \ln \frac{n}{k} \Leftrightarrow \ln k = \ln n - \lambda \tag{B.37}$$

$$\alpha := \ln n \tag{B.38}$$

$$\beta := -\ln \epsilon \tag{B.39}$$

$$\gamma := \ln \ln \frac{n}{\epsilon k} \tag{B.40}$$

we can rewrite eq.B.36 as

$$\lambda \left((1 + \frac{1}{-\lambda + \beta}) \cdot (1 + \frac{1}{-\lambda + \alpha - \gamma}) - 1 \right) - 1 = 0 \quad (!) \tag{B.41}$$

$$\Leftrightarrow \quad \lambda^2 + \lambda(1 + 2\beta) - \beta(\alpha - \gamma). \tag{B.42}$$

The relevant solution of this quadratic equation is

$$\lambda_1 = \frac{1}{2}\sqrt{1+4\beta^2+4\beta(\alpha-\gamma+1)} - \frac{1+2\beta}{2} \tag{B.43}$$

$$\approx \sqrt{\alpha\beta+\beta^2} - \frac{1+2\beta}{2} \tag{B.44}$$

$$\approx \sqrt{\alpha\beta}, \tag{B.45}$$

where the approximations are valid for large n. From eq.B.37 we obtain finally

$$k_{\text{opt}}^{\text{sprs}} = n \cdot e^{-\lambda_1} \tag{B.46}$$

$$\sim n \cdot e^{0.5-\ln\epsilon-\sqrt{(\ln\epsilon)^2-(\ln n)\ln\epsilon}} \tag{B.47}$$

$$\sim n \cdot (e^{\sqrt{-\ln\epsilon}})^{-\sqrt{\ln n}}. \tag{B.48}$$

Note that the approximations are very raw and deliver small relative errors only logarithmically for large n. On the other hand eq.B.46 contains k implicitly in γ. To make use of eq.B.46 we can start with eq.B.47 to obtain an initial value for γ, and then iteratively apply eqs.B.40 and B.46. Figure B.1 illustrates $k_{\text{opt}}^{\text{sprs}}(n)$ and $C^{\text{sprs}}(k_{\text{opt}}^{\text{sprs}})$ using different approximations.

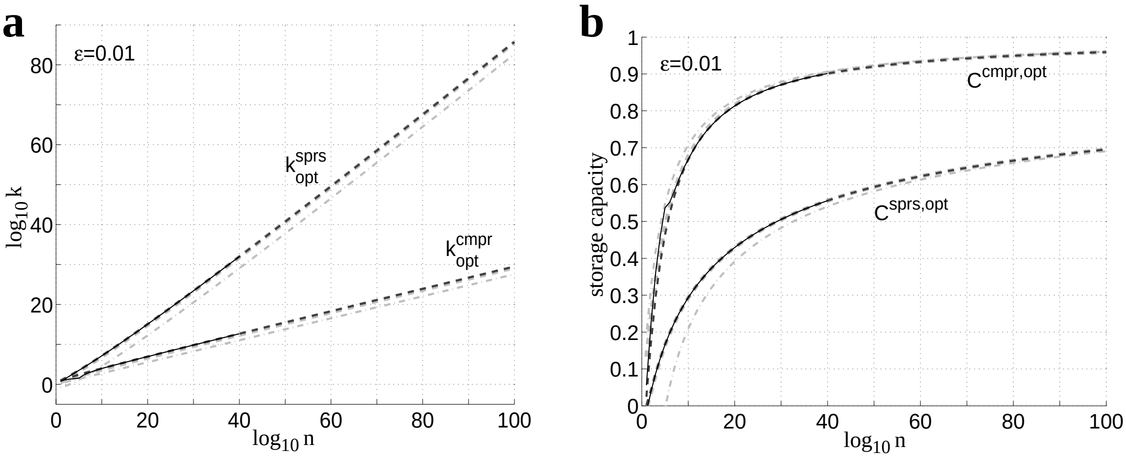


Figure B.1: Maximal storage capacities for moderately sparse patterns with super-logarithmic pattern activity $k(n)$ occur at $k_{\text{opt}}^{\text{sprs}}(n)$ for sparse matrix representation and $k_{\text{opt}}^{\text{cmpr}}(n)$ for optimal matrix compression. The plots show exact values (solid), and approximations according to eqs.B.48/B.61 (dash-dotted, gray), and eqs.B.47/B.60 (dashed, gray), and eqs.B.46/B.59 after 5 iterations (dashed, black) for high-fidelity parameter $\epsilon = 0.01$. **a** : $\log_{10} k_{\text{opt}}^{\text{sprs}}$ and $\log_{10} k_{\text{opt}}^{\text{cmpr}}$ vs. $\log_{10} n$. **b** : $C(k_{\text{opt}}^{\text{sprs}})$ and $C(k_{\text{opt}}^{\text{cmpr}})$ vs. $\log_{10} n$.

Optimal matrix compression

For the storage capacity C^{cmpr} corresponding to optimal matrix compression (see eq.3.15) we have for moderately sparse patterns with $p_1 \to 1$ (cf. section 3.2.2)

$$i_{\text{zip}} = I(p_1) = -p_1 \text{ld}\, p_1 - (1-p_1)\text{ld}\,(1-p_1) \tag{B.49}$$

$$i_{\text{zip}}' := \frac{di_{\text{zip}}}{dp_1} = I(p_1)' = -\text{ld}\, p_1 + \text{ld}\,(1-p_1). \tag{B.50}$$

In order to compute the optimal pattern activity $k_{\text{opt}}^{\text{cmpr}}$ we can write eq.B.30 with the definitions in section B.4.1 equivalently as

$$\ln\frac{n}{k}\left(\frac{kp_1'}{1-p_1}\left(\frac{1}{-\ln(1-p_1)} - \frac{(1-p_1)I(p_1)'}{I(p_1)}\right) - 1\right) - 1 = 0 \quad (!). \tag{B.51}$$

With $p_1 \approx 1$ and $1 - p_1 \approx -\ln p_1$ we can approximate

$$\frac{(1-p_1)I(p_1)'}{I(p_1)} = \frac{-\ln p_1 + \ln(1-p_1)}{\frac{-p_1 \ln p_1}{(1-p_1)} - \ln(1-p_1)} \approx \frac{\ln(1-p_1)}{1-\ln(1-p_1)}. \tag{B.52}$$

Inserting in eq.B.51 yields

$$\ln \frac{n}{k}\left(\frac{kp_1{}'}{1-p_1}(1+\frac{1}{-\ln(1-p_1)(1-\ln(1-p_1)})-1\right)-1 = 0 \quad (!). \tag{B.53}$$

With the approximations eqs.B.34 and B.35, and the substitutions eqs.B.37-B.40 we obtain

$$\lambda\left((1+\frac{1}{\lambda+\beta})\cdot(1+\frac{1}{(\lambda+\alpha-\gamma)\cdot(1-\lambda+\alpha-\gamma)})-1\right)-1=0 \quad (!) \tag{B.54}$$

$$\Leftrightarrow \quad \lambda^2(\beta-1)-\lambda(2\beta(\alpha-\gamma+1)+1)+\beta(\alpha-\gamma)(\alpha-\gamma+1)=0 \quad (!). \tag{B.55}$$

The relevant solution of this quadratic equation is

$$\lambda_1 = \frac{2\beta(\alpha-\gamma+1)+1}{2(\beta-1)} - \frac{\sqrt{(2\beta(\alpha-\gamma+1)+1)^2-4(\beta-1)\beta(\alpha-\gamma)(\alpha-\gamma+1)}}{2(\beta-1)} \tag{B.56}$$

$$\approx (\alpha-\ln\alpha)\cdot\frac{\beta-\sqrt{\beta}}{\beta-1} \tag{B.57}$$

$$\approx \alpha\cdot\frac{\beta-\sqrt{\beta}}{\beta-1} \tag{B.58}$$

where the approximation is valid for large n. From eq.B.37 we obtain finally

$$k_{\mathrm{opt}}^{\mathrm{cmpr}} = n\cdot e^{-\lambda_1} \tag{B.59}$$

$$\approx n\cdot(\frac{n}{\ln n})^{-\frac{-\ln\epsilon-\sqrt{-\ln\epsilon}}{-\ln\epsilon-1}} \tag{B.60}$$

$$\approx n^{1-\frac{-\ln\epsilon-\sqrt{-\ln\epsilon}}{-\ln\epsilon-1}}. \tag{B.61}$$

Note that the approximations are very raw and deliver small relative errors only logarithmically for large n. On the other hand eq.B.59 contains k implicitly in γ. To make use of eq.B.59 we can start with eq.B.60 to obtain an initial value for γ, and then iteratively apply eqs.B.40 and B.59. Figure B.1 illustrates $k_{\mathrm{opt}}^{\mathrm{cmpr}}(n)$ and $C^{\mathrm{cmpr}}(k_{\mathrm{opt}}^{\mathrm{cmpr}})$ using different approximations.

B.4.3 For non-sparse patterns asymptotic storage capacity is generally zero

In section 3.2.3 we have seen for non-sparse linear pattern activity $k = cn$ with $0 < c \ll 1$ that the classical binomial analysis (see section 3.1.2, in particular eq.3.6) can still yield asymptotic storage capacities in the range of 1 for compressed memory matrices. However, in section 3.6.3 we have shown that the classical binomial analysis is not valid for linear k. The analysis showed that for $k/n = c$ the variance of the non-addressed neurons potentials is truly quadratic in the address pattern size z (typically $z = k$) whereas the binomial approximation (cf. eqs.3.122 and 3.123) yields variances that are only linear in z.

Our original high-fidelity criterion requires for the probability of a false one in the retrieval result (i) $p_{01} \to 0$ and (ii) $p_{01}/(k/n) \approx 0$ (see section 3.1.2, cf. [134]). For sub-linear k it was sufficient to require $p_{01} \leq \epsilon k/n$ for a constant $0 < \epsilon \ll 1$, whereas for linear $k = cn$ we have to require $\epsilon \to 0$ in order to satisfy (i). However, the estimations of capacity (similar as performed in sections 3.1.2 and 3.2.3 using illegally eq.3.6) are still valid for a *weaker* high-fidelity criterion with small *constant* ϵ and $p_{01} \leq \epsilon c$ (see appendix B.1). In the following we will demonstrate by taking into account the true variance of neuron potentials (section 3.6.3) that the actual asymptotic storage capacity for linear $k = cn$ is generally zero, even if the memory matrix is optimally compressed. We will see that for linear $k = cn$ and *any* reasonable fidelity criterion only a constant number of patterns can be stored asymptotically.

For address pattern size $z = k = cn$ (perfect address pattern) we obtain for the expectation and variance of the potential X_R of a non-addressed neuron from eq. 3.101 and eq. 3.119 using eq. 3.4

$$
\begin{align}
E(X_R) &= k \cdot p_1 \tag{B.62}\\
\mathrm{Var}(X_R) &:= E({X_R}^2) - (E(X_R))^2 \tag{B.63}\\
&\approx k^2 - k(2k-1)(1-(k/n)^2)^M + k(k-1)e^{-2Mc^2(1-c/2)} - k^2 \cdot {p_1}^2 \tag{B.64}\\
&\approx k^2(1-{p_1}^2) - k(2k-1)(1-p_1) + k(k-1)(1-p_1)^d \tag{B.65}\\
&= \left((1-p_1)^d - (1-p_1)^2\right) k^2 + \left(1 - p_1 - (1-p_1)^d\right) k \tag{B.66}\\
&= a^2k^2 + bk = a^2k^2(1 + b/k), \tag{B.67}
\end{align}
$$

where $d := -2c^2(1-c/2)/\ln(1-c^2)$ is a constant with $0 < d < 2$, and $a^2 := (1-p_1)^d - (1-p_1)^2$, and $b := 1 - p_1 - (1-p_1)^d$. If we assume storing only few patterns such that we have only *constant* $0 < p_1 < 1$ for $n \to \infty$ (instead of $p_1 \to 1$ as suggested by the binomial analysis for high-fidelity; cf. eq.3.9) then a and b are also constant and we have $\sigma := \sqrt{\mathrm{Var}(X_R)} \approx ak \to \infty$. In analogy to the approximation of binomials by Gaussians we hypothesize that also the Willshaw distribution $W_{k,n,k,M}$ can be approximated by a Gaussian $N_{\mu,\sigma}$ with corresponding expectation $\mu = kp_1$ and variance σ^2. For the error probability $p_{01} \approx \mathrm{pr}[N_{\mu,\sigma} \geq k]$ that a given non-addressed ('false') neuron exceeds the threshold k (Willshaw strategy, see section 3.1.1) we can write

$$
p_{01} \approx 1 - G\left(\frac{k-\mu}{\sigma}\right) \approx 1 - G\left(\frac{1-p_1}{a}\right) = 1 - G\left(1/\sqrt{\frac{1}{(1-p_1)^{2-d}} - 1}\right) \tag{B.68}
$$

where G is the Gaussian error function defined in eq. 4.9. Thus for constant matrix load p_1 we have already a constant positive error probability p_{01}. For $p_1 \approx 1$ (as suggested by the classical binomial analysis and high-fidelity) we obtain truly $p_{01} \approx 1 - G(0) = 0.5$ asymptotically for $n \to \infty$ which contradicts *any* reasonable fidelity criterion. Since p_{01} is monotonically increasing in p_1 we have also $p_{01} = 0.5$ in the limit $p_1 \to 1$ for $n \to \infty$.

Thus any reasonable fidelity requirement will forbid $p_1 \to 1$ asymptotically for $n \to \infty$. However, for constant $p_1 < 1$ we can store asymptotically only a constant number of patterns $M = -\ln(1-p_1)/\ln(1-c^2)$ (see eq. 3.29), and therefore we obtain $C = MI(c)/n \to 0$ (see eq. 3.30). Since $I(p_1)$ is also constant we have asymptotically $C^{\mathrm{cmpr}} := C/I(p_1) \to 0$.

Appendix C

Simulation tool Felix++

All simulations described in this work have been implemented using the *Felix* or *Felix*++ simulation tools. Originally the C based simulation tool *Felix* has been developed by Thomas Wennekers at the University of Ulm [197] as a universal simulation environment for physical and, in particular, neural systems. The development of Felix was motivated by the need for a fast implementation of multi-layer one- or two-dimensional neural structures such as neuron populations. For this purpose, Felix provides elementary algorithms for single-cell dynamics, inter-layer connections, and learning. Additionally, there exist also libraries for non-neural applications, e.g., for general dynamical systems and elementary image processing.

Simulations can be observed and influenced online via the X11/XView-based *graphical user interface* (GUI) of Felix. The Felix GUI provides elements such as *switches* for conditional execution of code fragments, *sliders* for online-manipulation of simulation parameters (like connection strengths, time constants, etc.), and *graphs* for the online observation of the states of a simulated system in xy-plots or gray-scale images (see [197, 88] for more details).

During this work the simulation tool Felix++ has been developed as a C++ based object-oriented extension of Felix. Felix++ provides additionally classes for neuron models, n - dimensional connections, pattern generation, and data recording. Current installations of Felix++ are running on PC/Linux as well as on 64bit-SunFire/Solaris9 systems. In the following the architecture of Felix++ is briefly sketched (for more details see [88]).

C.1 Basic architecture of Felix++

Essentially Felix++ is a collection of C++ libraries supporting fast development of neural networks in C++ [178, 180]. Thus Felix++ comprises a number of modules each consisting of a header (with the suffix ".h") and a corpus (with the suffix ".cpp" for Felix++/C++ or ".c" for Felix/C). The header files contain declarations of classes, types, and algorithms, whereas in the corpus files the declarations are implemented. Figure C.1 illustrates the architecture of Felix++ by classifying all the modules of Felix++ and Felix in a hierarchy.

C.1.1 The core modules of Felix++

The *core* of Felix++ contains the most important modules required by all other Felix++ modules.

- **F2_types.h/cpp** declares some elementary type conventions and some global objects.
- **F2_time.h/cpp** declares classes for time, for example to evaluate the time necessary for computing a simulation.
- **F2_random.h/cpp** provides several different random number generators (see [142]).

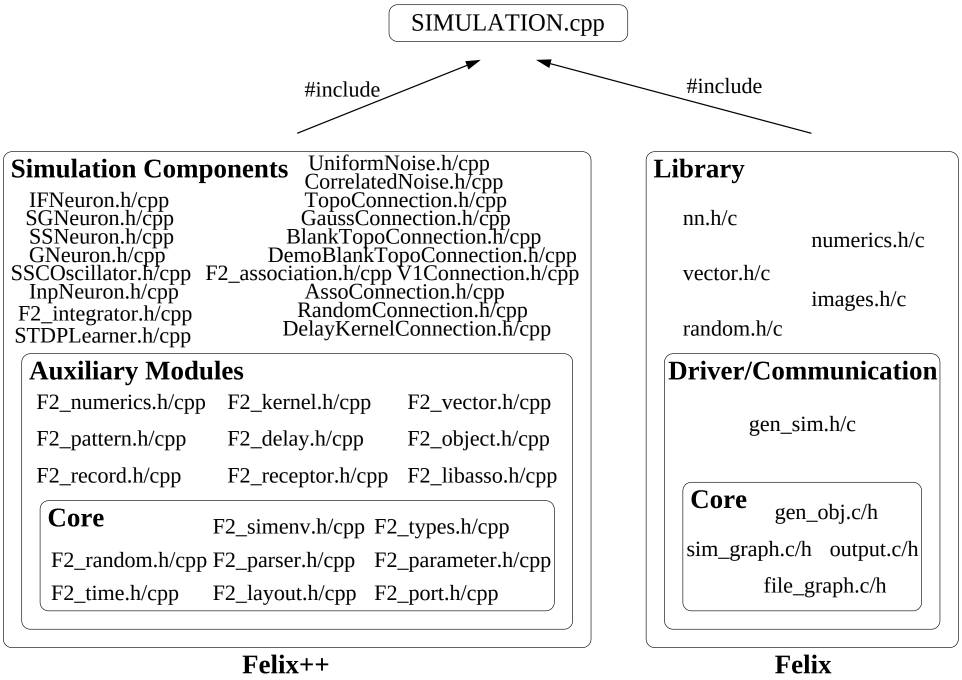


Figure C.1: Architecture of Felix++: A simulation is a C++ program that includes headers of Felix and Felix++. Felix++ contains core modules (e.g., *F2_simenv.h/cpp*; cf. Fig. C.2), auxiliary modules, and modules for simulation components such as neuron populations and connections between neuron populations (cf. Fig. C.3). The Felix modules implement a graphical user interface and elementary algorithms (see [197] for more details).

- **F2_layout.h/cpp** declares so-called *layouts*. A layout can be used to define the topology of a vector (or in terms of C++, an array). For example a population of 1000 neurons can be arranged as a 10 × 10 × 10 *cuboid*. Apart from cuboid layouts also *ellipsoid* layouts are defined which are useful in particular for saving memory when modeling isotropic local connectivity (in three dimensions, for example, an ellipsoidal kernel saves almost 50 percent of the memory required by a cuboid kernel).

- **F2_parameter.h/cpp** declares classes for simulation parameters. For example, the membrane time constant field of a neuron class is usually declared as such a parameter. This allows conveniently parsing and online reparsing of the parameters from a parameter file as well as online manipulation via sliders of the Felix GUI (including updating of other dependent parameters). The parameters declared in this module are essentially arrays equipped with a cuboid layout (see above). Therefore they can be used not only for single parameter values, but as well for multi-dimensional parameter collections such as parameter vectors or matrices.

- **F2_parser.h/cpp** provides the classes for parsing parameters from a parameter file. Usually, in Felix++ a component class (e.g., a neuron class) is designed in such a way that a construction of an object is paralleled by parsing the corresponding parameters from a file. Furthermore, during the simulation the parameter file can be modified and reparsed by pressing the *reparse-button*.

- **F2_port.h/cpp** declares interfaces for the communication between different simulation components, so-called ports. For example, a neuron class may contain an output port representing the spikes of the neuron, and an input port representing synaptic input to the neuron. Correspondingly, the constructor of a connection component class requires as parameters the output port of a neuron population and the input port of another neuron population such that the spikes from the first population can be propagated to the dendrites of the second population.

- **F2_simenv.h/cpp** declares the simulation environment class `TSimulationEnvironment` and the component base class `TComponent` as well as some base classes for special components such as neurons (`TNeuron`) and connections (`TConnection`). This module should be included by *any* simulation program using Felix++. The simulation environment is essentially a container for the simulation components (see below for more details; cf. Fig. C.2), but provides also additional infrastructure such as look-up tables (for example for Gaussians), random number generators, and much more. Usually, the construction of a simulation component requires a `TSimulationEnvironment` as an argument, such that the component is automatically inserted. After construction of all the components, calls to methods `allocate()` will allocate memory shared by multiple components (for example when integrating differential equations via `TIntegrator` objects; see below). Before starting the simulation all the components can be initialized by calling the `init()` method of the simulation environment. Similarly, during the simulation a call to the `step()` method will compute one simulation step.

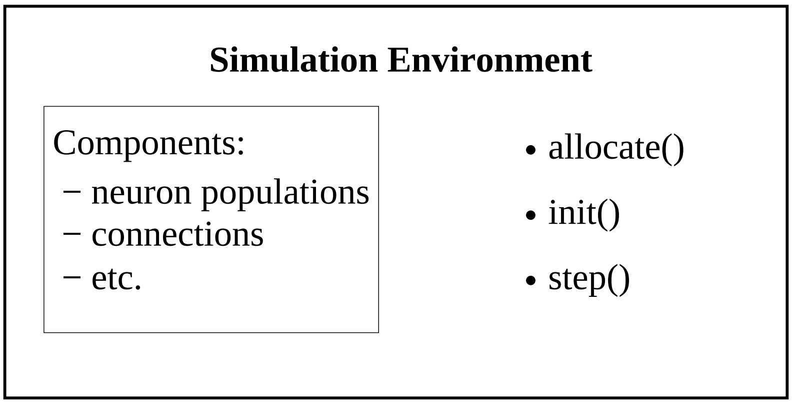


Figure C.2: The simulation environment object (of class `TSimulationEnvironment`) is essentially a container object containing all the simulation components such as neuron populations or connections. The components are inserted during construction. Before starting a simulation a call to method `allocate()` is necessary to allocate memory. A call to `init()` initializes the components, and each call to `step()` results in the computation of one simulation step.

C.1.2 Auxiliary modules of Felix++

Besides the core modules there are a number of auxiliary modules that provide additional functionality required by only some of the Felix++ component modules, and perhaps also by the programmer developing a simulation.

- **F2_numerics.h/cpp** provides a number of useful constants (e.g., π, e, and $\ln 2$) and functions (e.g., density function of Binomials or Gaussians, information and transinformation functions for binary random variables, etc.). Further declarations provide classes for look-up-tables and interpolation.

- **F2_kernel.h/cpp** declares classes for kernels that can be used, for example, for implementing synaptic connections. Kernels are essentially arrays (e.g., of synaptic weights or delays) that have been assigned a topology via layouts (see *F2_layout.h/cpp*). The classes defined in this module enable, for example, the coordination of a neuron population (layout) to a set of kernels. This happens in a rather flexible manner such that each neuron can be assigned individually a kernel index, where also certain regularities of kernel arrangements can be exploited (such as the regularities occurring for the orientation modules in our visual model; cf. Fig. 5.2a).

- **F2_vector.h/cpp** provides basic vector functionality. This module also declares classes for numerical vector parameters (cf. *F2_parameter.h/cpp*).
- **F2_pattern.h/cpp** implements classes for various types of patterns. From the pattern base type (`TMPattern`) which corresponds simply to a multi-dimensional array there are derived specialized pattern types such as binary patterns (`TMbPattern`), sparse binary patterns (`TMsbPattern`), sparse patterns (`TMsPattern`), or sparse binary spatio-temporal patterns (`TMsbSTPattern`). Additionally, further auxiliary classes have been implemented in order to facilitate the use of patterns. For example, pattern container classes are declared (`TMPatternStock` and derivatives of `TMPatternGroup`) for convenient construction and parsing of pattern groups from parameter files. The `TMPatternRanking` class can be used for analyzing neural activity with respect to a set of patterns (i.e., to determine the pattern in the set that is most similar to the neural activity pattern). Similarly, the `TMPatternHistogram` class can be used to create pattern-specific histograms of state variables (as used, for example, for the threshold distance histograms in Fig. 4.16d-h).
- **F2_delay.h/cpp** provides classes based on the definitions in *F2_kernel.h/cpp* for efficient implementation of synaptic delays.
- **F2_object.h/cpp** declares classes for generating stimulus objects. Further classes can be used to put static or moving objects in space (derivatives of `TMSpace`), or to project the stimulus configuration onto a two-dimensional surface (derivatives of `TMSpaceRepresentation`). In the visual model of chapter 5 these classes have been used in order to project a visual scene of several stimulus objects onto the retinal area R (see Figs. 5.1a and 5.3a).
- **F2_record.h/cpp** provides the infrastructure for efficient recording of simulation data.
- **F2_receptor.h/cpp** declares classes for the efficient implementation of various types of receptors for synaptic transmitters. Derivatives of class `TMReceptorPort` can be used, for example, to implement certain transmitter-dependent synaptic conductances. For our neuron model described in section 2.4 we used class `TMOffDynamics` for implementation of excitatory AMPA currents (conductance g_{ex}; cf. eq. 2.1) and inhibitory GABA-A currents (conductance g_{in}; cf. eq. 2.2). More complex receptor dynamics are implemented by classes `TMOnOffDynamicsRP` and `TMSimpleNMDARP` where the latter can be used to model NMDA receptor dependent currents (cf. [101]). Actually neuron classes such as `TSSNeuron` (which we have used for our biological simulations) or `TGNeuron` use the receptor port classes provided by this module. These models can be equipped with an arbitrary configuration of different receptor ports which can be specified in the parameter file (see below code fragment C.5).
- **F2_libasso.h/cpp** encapsulates the C-library for associative memory implemented by Friedrich Sommer (cf. [171]).

C.1.3 Component classes of Felix++

Based on the core and auxiliary modules there exists already a large number of simulation components. Figure C.3 illustrates the class hierarchy of the Felix++ simulation components. They can be divided into the following component base classes derived from `TComponent`:

- `TNeuron` (defined in module *F2_simenv.h/cpp*) is the base class for all neuron classes. Currently there are implementations for gradual neurons (`TSGNeuron` in module *SGNeuron.h/cpp* and `TGNeuron` in module *GNeuron.h/cpp*), spiking neurons (`TIFNeuron` in module

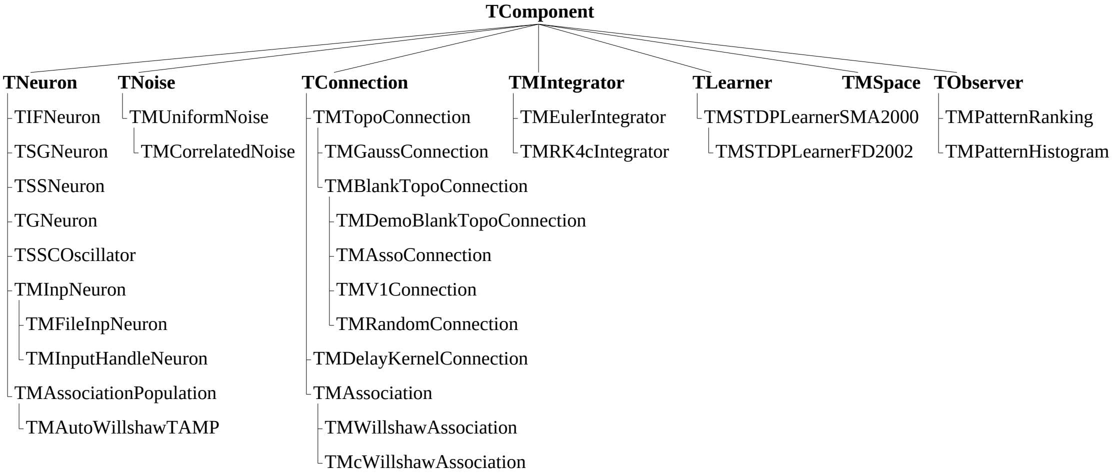


Figure C.3: The class hierarchy for the currently implemented simulation components of Felix++. From the base class *TComponent* specialized sub-classes are derived for neuron populations (`TNeuron`), noise generators (`TMNoise`), connections between neuron populations (`TConnection`), integration of differential equations (`TMIntegrator`), synaptic plasticity (`TLearner`), representations of stimulus space (`TMSpace`), and on-line observation of the simulation state (`TObserver`).

IFNeuron.h/cpp and `TSSNeuron` in module *SSNeuron.h/cpp*), and oscillators (`TSSCOscillator` in module *SSCOscillator.h/cpp*) which all can be used for biological modeling. Additionally, there are classes adequate for technical implementations of associative memory (`TMAssociationPopulation` and `TMAutoWillshawTAMP` in module *F2_association.h/cpp*). For the simulations of biological models in section 4.6 and chapter 5 we used the simple spiking neuron class `TSSNeuron`, whereas for the implementation of Willshaw associative memory and the spike counter model in section 4.5.4 the technical associative memory population class `TMAutoWillshawTAMP` has been used.

- `TMNoise` (defined in module *F2_simenv.h/cpp*) is the base class for noise populations. A noise population provides random numbers generated according to a certain distribution for another component object such as a neuron population. Derivatives of type `TMUniformNoise` (defined in module *UniformNoise.h/cpp*) provide uniformly distributed random numbers with a certain power (or variance). While this type generates independent random numbers in each simulation step the random numbers generated by derivatives of type `TMCorrelatedNoise` can be correlated in space and time. The standard noise type for neuron populations such as `TSSNeuron` (or for synaptic noise in connections; see module *F2_receptor.h/cpp*) is `TMUniformNoise`, whereas for the primary visual area P (see chapter 5) correlated noise of type `TMCorrelatedNoise` has been used.

- `TConnection` (defined in module *F2_simenv.h/cpp*) is the base class for connections between neuron populations (or more exactly, between ports; see module *F2_port.h/cpp*). The function of derivatives of this type is to propagate information from an output port to an input port, for example, to propagate the spikes from the output port of one neuron population through the network to the input port of another neuron population. The most important derived type for biological modeling is `TMTopoConnection` (defined in module *TopoConnection.h/cpp*). This generic type is the base class for many further derived classes. Here a synapse is defined by two state values: A synaptic weight, and a synaptic delay. Correspondingly, an object of type `TMTopoConnection` essentially contains two kernel arrays of type `TKernel` (defined in *F2_kernel.h/cpp*) representing weights and delays. The kernel classes can be applied in a very flexible manner allowing implementation of full, sparse, topographical schemes in multiple dimensions. Additionally, efficient algorithms are implemented for several special cases (e.g., for non-sparse bit-packed binary topographical connectivity). The derivatives of `TMTopoConnection` merely specify *how* the synaptic weight and delay kernels are generated. For example, class `TMGaussConnection` (defined in module *GaussConnection.h/cpp*) implements simple topographical connections with Gaussian kernels. A further derived class `TMBlankTopoConnection` (defined in module *BlankTopoConnection.h/cpp*) provides an interface to `TMTopoConnection` in order to allow a more convenient derivation of further connection classes. While `TMDemoBlankTopoConnection` (defined in module *DemoBlankTopoConnection.h/cpp*) is merely a demonstration how to derive from `TMBlankTopoConnection`, also a number of important connection classes have been derived. `TMAssoConnection` (defined in module *AssoConnection.h/cpp*) can be used to implement fully connected or multi-dimensional topographically confined associative connections, for example, of the Willshaw type. `TMV1Connection` (defined in module *V1Connection.h/cpp*) is a specialized connection scheme for the primary visual cortex as used for the simulations in chapter 5 (see Fig. 5.2). And `TMRandomConnection` (defined in module *RandomConnection.h/cpp*) can be used conveniently for implementing connections with random connectivity. Another class derived directly from `TConnection` is `TMDelayKernelConnection` (defined in module *DelayKernelConnection.h/cpp*) which provides a much simpler and faster scheme for delayed connections than `TMTopoConnection`. For implementation of technical associative memory derivatives from class `TMAssociation` (defined in module *F2_association.h/cpp*) can be used, such as `TMWillshawAssociation` and `TMcWillshawAssociation` for the Willshaw model, where the latter implements compression of the binary memory matrix.

- `TMIntegrator` (defined in module *F2_integrator.h/cpp*) is the base class for numerical integration of differential equations. Derived classes (defined in the same module) are `TMEulerIntegrator` which implements a simple first order Euler method, and `TMRK4cIntegrator` which implements the fourth order Runge-Kutta method with constant step size (see [142]). Normally, these integrator objects are used by some of the neuron classes (e.g., `TSSNeuron`).

- `TLearner` (defined in module *F2_simenv.h/cpp*) is the base class for plasticity of synaptic connections. Currently, two derivatives are implemented in the module *STDPLearner.h/cpp*. `TMSTDPLearnerSMA2000` implements a model of spike-timing dependent synaptic plasticity (STDP) described by Song, Miller, and Abbott [173], while `TMSTDPLearnerFD2002` implements an extended model suggested by Froemke and Dan [53]. Both classes are interfaced with connection classes via the kernel classes defined in *F2_kernel.h/cpp*. Therefore it is easy to endow connections (e.g., derived from `TMTopoConnection`) with synaptic plasticity.

- `TMSpace` (defined in module *F2_object.h/cpp*) is the base class for the definition of a space for stimulus objects (see above module *F2_object.h/cpp*).

- `TObserver` (defined in module *F2_simenv.h/cpp*) is the base class for components observing on-line the state of the simulation. Derived classes are `TMPatternRanking` and `TMPatternHistogram` (for more details see above module *F2_pattern.h/cpp*).

C.2 Simulation environment and components: Code examples

In the last section we have obtained an overview over the modules of Felix++. In the following we will have a closer look at the code defining some important Felix++ classes, the simulation environment, and the base class for components. The code examples shown below are shortened fragments of the declarations in the Felix++ headers.

C.2.1 Simulation environment: Class TSimulationEnvironment

The simulation environment is essentially a container for the simulation components used in a simulation program (cf. Fig. C.2), but also provides additional infrastructure such as look-up tables, random number generators, and some further global variables. The following code fragment taken from the header *F2_simenv.h* shows parts of the declaration of the class `TSimulationEnvironment`.

Code fragment C.1

```
class TSimulationEnvironment {
 public:
  // part 1: local types
  typedef enum { CC_NOISE,               // for TNoise component category
                 CC_CONNECTION,          // for TConnection
                 CC_INTEGRATOR,          // for TIntegrator
                 CC_NEURON,              // for TNeuron
                 CC_LEARNER,             // for TLearner
                 CC_SPACE,               // for TSpace
                 CC_SPACEREPRESENTATION, // for TSpaceRepresentation
                 CC_OBSERVER,            // for observers like TMPatternRanking
                 CC_ASYNCHRONY           // calls to step() are not controlled
  } TComponentCategory;
  static const int nCategories=9;        // number of different component categories

  // part 2: parameters
  TPar     parStepSize;       // simulation step size (in milliseconds)
  TsPar    parDataDirectory;  // default directory where simulation data is recorded
  TsPar    parDataPostFix;    // post fix for data file names

  // part 3: object fields
  const char* parameterFile;                 // name of root parameter file
  TParser parser;                            // parser for parameters
  vector<TComponent*> allComponents;         // all components in order of constr.
  vector<TComponent*> components[nCategories]; // components ordered after categories

  // part 4: time reference
  TInt    steps;              // current simulation step number
  TFloat  simTime;            // current simulation time, simTime:=steps*stepSize;

  // part 5: constructors/destructors
  TSimulationEnvironment(const char* parFile); // default constructor
  ~TSimulationEnvironment();                   // destructor

  // part 6: methods
  void parse();                                            // parsing of parameters
  void reparse();                                          // reparse parameters
  void addComponent(TComponent *c, TComponentCategory cc); // add component c

  void allocate(); // simulation phase 2 (after creation phase): memory allocation
  void init();     // simulation phase 3: Initialization of TComponents
  void step();     // simulation phase 4: compute one simulation step
};
```

In part 1 an enumeration type is declared for the different component categories such as neurons, connections, or observers. This corresponds approximately to the different component classes described in section C.1.3 (see Fig. C.3; see also the container declarations in part 3).

In part 2 parameters are declared such as the simulation step size, or the directory for recorded simulation data. These parameters are parsed from a parameter file (see code fragment C.5) during the first execution phase of the simulation program when the simulation environment is created by calling the constructor (see part 5).

In part 3 some object fields are declared such as the name of the parameter file (which is passed as an argument to the constructor; see part 5), or the parser used for parsing the parameters (see module *F2_parser.h/cpp* in section C.1.1). Here there are also the declarations of the containers for the simulation components (vectors of the STL library; cf. [178, 180]). The first container field `allComponents` contains references to the components in order of the construction of the components (which is important for reparsing the parameter file), while the second container field `components` contains the components ordered for the different categories (see part 1). The latter ordering is important to assert a defined synchronization of object of the same component category. For example, the calls to the `step()` methods of neuron objects should be *before* the `step()` calls of connection objects (cf. part 6). This synchronization will become even more important when parallelizing Felix++ as planned for future work (cf. [177]).

In part 4 some fields for time reference are defined. Felix++ is a step-based simulation tool (in contrast to event-based tools). This means that the state of the simulated system is updated step by step where one simulation step corresponds to a fixed time interval. The parameter `parStepSize` (see part 2) defines this time interval. The field `steps` is initialized by 0, and incremented for each call to the `step()` method (see part 6).

Part 5 contains the declarations of the constructors and destructors. In section C.3 an example is given how and when to apply the constructor in a simulation program.

In part 6 the methods are declared. Method `parse()` is normally called by the constructor in order to parse the parameter file. Method `reparse()` is called for reparsing a modified parameter file, for example, when pressing the *reparse-button* in an online-simulation. With `addComponent()` new simulation components can be added to the component containers (see part 3) which is usually done by the constructor of `TComponent` (see section C.2.2). Then there are further three important methods (cf. Fig. C.2): In order to allocate memory shared by different simulation components (for example state variables of neurons integrated by components of type `TIntegrator`; see module *F2_integrator.h/cpp*) a call from the simulation program to `allocate()` must occur *after* the construction of the simulation components (see procedure `main_init()` in code fragment C.4). A call to method `init()` initializes the simulated system essentially by calling the `init()` method of each simulation component. Similarly, a call to method `step()` computes one simulation step by calling the `step()` method of each simulation component. The calls to the simulation environment's `init()` and `step()` methods occur normally from the simulation program's `init()` and `step()` procedures (see code fragment C.4).

C.2.2 Components: Class TComponent

The class `TComponent` is the base class for all simulation components such as neurons or connections (see Figs. C.3 and C.1) and implements essentially a common interface of simulation components to the simulation environment (section C.2.1). The following code fragment taken from the header *F2_simenv.h* shows parts of the declaration.

Code fragment C.2

```
class TComponent : public TParamOwner {
 public:
  // part 1: object fields
  string name;                     // name of the component
  TSimulationEnvironment& simEnv; // reference to the simulation environment
  TComponentCategory category;     // component category
  vector<TfPort*> gradualPorts;    // gradual i/o ports of the component
  vector<TbPort*> binaryPorts;     // binary ports

  // part 2: constructors/destructors
  TComponent(TSimulationEnvironment& simEnv_arg, const char* name_arg,
             TComponentCategory cc_arg);
  TComponent(TComponent& pattern, const char* name_arg, TComponentCategory cc_arg);
  ~TComponent();

  // part 3: methods
  virtual void reparse();          // reparse parameters
  virtual void allocate();         // allocate memory for TIntegrator
  virtual void init();             // initialize component
  virtual void step();            // compute one simulation step of component

  virtual void derivs() {};       // compute derivatives for TIntegrator
};
```

Class `TComponent` is derived from class `TParamOwner` in order to provide the functionality of the parameter classes (see module *F2_parameter.h/cpp* in section C.1.1).

In part 1 object fields are declared. Each simulation component can be assigned a `name` which considerably relieves search for errors. Field `simEnv` refers to the simulation environment containing the object, and field `category` contains information about the component category of the object (cf. section C.2.1). The fields `gradualPorts` and `binaryPorts` are containers for gradual and binary ports (see module *F2_port.h/cpp* in section C.1.1) and constitute thereby the interface for communication between different simulation components. For example, the spikes of a neuron population will be represented by a binary port of type `TbPort` and similarly the dendritic inputs will be represented by a gradual port of type `TfPort`. Thus a connection component can connect two neuron populations, for example, by propagating spikes from the binary port of the first population through the synaptic network to the gradual input port of the second population.

In part 2 the constructors and destructors are declared. There are generally two constructor types for a simulation component. The *default* (or *complete*) constructor constructs a simulation component by parsing the parameters from the parameter file (see code fragment C.5). In contrast, the *pattern* constructor requires as argument a simulation object of the same type where the parameters of this pattern are used for construction of the new object (see also section C.2.3).

In part 3 a number of virtual methods is defined which are normally overridden by derived component classes and called by the simulation environment. The `reparse()` method reparses the parameters from the parameter file using `simEnv` → `parser`. If necessary the `allocate()` method requests memory shared with other components from a further object managing the shared memory (for example from an `TIntegrator` object; see module *F2_integrator.h/cpp* in section C.1.3). A call to `init()` will initialize the simulation component, and a call to `step()` will compute one simulation step for the component. The method `derivs()` can be used in derived classes to compute the (numerical) derivatives of some of the state variables of the component. This method is normally called by a `TIntegrator` object in order to integrate the differential equation associated with a component state.

C.2.3 Class TSSNeuron: a simple spiking neuron model

To illustrate how a concrete simulation component can be derived from the base class `TComponent` we will have a closer look at the class `TSSNeuron` implementing a simple spiking neuron model. Actually, this class (with a parameter file as shown in code fragment C.5) has been used to implement the model described in section 2.4 for the biological simulations in section 4.6 and chapter 5. The following code fragment taken from the header *SSNeuron.h* shows parts of the declaration.

Code fragment C.3

```
class TSSNeuron : public TNeuron, public TIntegratorClient {
 public:
  // part 1: parameters
  string  scopeID;                      // scope id for parsing parameters
  TVecPar tau_x;                        // membrane time constant
  TVecPar theta;                        // asymptotic threshold
  TVecPar refAbs;                       // absolute refractory time
  TVecPar refRel;                       // relative refractory parameter
  TVecPar tau_h;                        // decay time constant of habituation
  TVecPar thetaInc_h;                   // threshold increment after each spike
  TCompartmentReceptors* receptorPorts; // receptorPorts

  // part 2: integrator for membrane potential
  TDerivScope derivScope;
  TIntegrator* integrator;              // integrator

  // part 3: ports
  TbPort *out;                          // output queue for spikes
  TfPort *lastOut;                      // output queue for last spikes
  TfPort* excIn;                        // default excitatory in-port
  TfPort* inhIn;                        // default inhibitory in-port

  // part 4: state variables
  TFloat *current;                      // synaptic currents
  TFloat *x;                            // membrane potential
  TByte  *y;                            // output variable (refers to out)
  TFloat *last;                         // last spike time (refers to lastOut)
  TFloat *habituation;                  // habituation (fatigue) - increased threshold

  // part 5: constructors/destructors
  TSSNeuron(TSimulationEnvironment& simEnv_arg, const char* name_arg, // complete cnstr.
            TLayout* layout_arg, vector<TNoise*>* noiseSources_arg);
  TSSNeuron(TSimulationEnvironment& simEnv_arg, const char* name_arg, // default    "
            TLayout* layout_arg, vector<TNoise*>* noiseSources_arg,
            TIntegrator* integrator_arg, int parse);
  TSSNeuron(TSSNeuron& pattern, const char* name_arg,                 // patterned  "
            TLayout* layout_arg, vector<TNoise*>* noiseSources_arg);
  ~TSSNeuron();                                                       // destructor

  // part 6: methods
  void allocate();                                      // get memory from integrator
  void derivs() {};                                     // compute ...
  void derivs(int id, TFloat t,TFloat* x,TFloat* dxdt); //  derivatives for integrator

  void init();                                  // initialize states to zero values
  void step();                                  // one simulation step
  void reparse();                               // reparse parameters
  void handleUpdatedParameters();               // handle updated parameters
  void setParameterValues();                    // update parameter values
  friend ostream & operator<<(ostream& os    , const TSSNeuron & neuron); // output op.
  friend istream & operator>>(istream& parser,       TSSNeuron & neuron); // input op.
};
```

Class `TSSNeuron` is derived from `TNeuron` (which in turn is derived from `TComponent`; see Fig. C.3) and from `TIntegratorClient`. The derivation from the latter class is necessary for any class requiring integration of differential equations by a `TIntegrator` object (see module *F2_integrator.h/cpp* in section C.1.3).

In part 1 of code fragment C.3 parameters of the neuron model are declared. When comparing with the notation used in section 2.4 (cf. table 2.1 and code fragment C.5) parameter field `tau_x` corresponds to parameter τ_x, `theta` to Θ_∞, `refAbs` to R_a, `refRel` to R_r, `tau_h` to τ_h, and `thetaInc_h` to H. Field `scopeID` contains information about which parameter scope (see below) in the parameter file has been used to parse the parameters for this object. Field `receptorPorts` points to a container object for receptor ports (see module *F2_receptor.h/cpp* in section C.1.2) which becomes allocated during object construction. This field is used for implementing different excitatory or inhibitory synaptic currents (see code fragment C.5).

In part 2 field `integrator` declares the integrator object for integrating the differential equation for the membrane potential (cf. eq. 2.4). Currently, either a Euler or a Runge-Kutta method can be used (see module *F2_integrator.h/cpp* in section C.1.3). The field `derivScope` just serves to identify the memory for the state variable `x` (see part 4) which is administrated by the integrator object.

Part 3 declares the ports of the object (see module *F2_port.h/cpp* in section C.1.1). Port `out` is the output queue for the spikes of the neuron population, port `lastOut` contains information about the time of the last spike for each neuron. Input ports `excIn` and `inhIn` are essentially queues for synaptic input from other neuron populations mediated by connection objects (see type `TConnection` in section C.1.3).

In part 4 the state variables of the neurons are declared. Array `current` is essentially the sum of the synaptic input currents for each neuron as computed by `receptorPorts` (see part 1). Array `x` corresponds to the membrane potential x of the neuron model in section 2.4 (cf. eq. 2.4). Similarly `y` corresponds to the spike output variable y (eq. 2.6), `last` corresponds to the time point s of the last spike for each neuron (cf. eq. 2.5), and `habituation` corresponds to the neuronal habituation or fatigue h (cf. eq. 2.3).

In part 5 the constructors and destructors are declared (cf. part 2 in section C.2.2). The first constructor is the so-called *complete constructor* which is normally used in a simulation program for creating an object of type `TSSNeuron` for the *first* time. The third constructor is the *pattern constructor* which is normally used for further creations of `TSSNeuron` objects. While for the complete constructor the parameters (see part 1) are parsed from the parameter file (see code fragment C.5), the pattern constructor copies the parameters from a pattern object passed as the first argument. The second constructor in the code fragment is the so-called *default constructor* which is normally used by the constructor of another class derived from `TSSNeuron`. In contrast to the complete constructor, automatic parsing (which should be done only by the constructor of the derived class) can be suppressed by passing an additional flag argument `parse`.

Finally part 6 contains the declarations of the methods `allocate()`, `derivs()`, `init()`, `step()`, and `reparse()` which override the declarations explained above for the base class `TComponent` (see part 3 in section C.2.2). There are a few remaining methods: `handleUpdatedParameters()` and `setParameterValues()` manage updating of the object state if one of the parameters (see part 1) is changed (for example when reparsing the parameter file), while the input/output operators `operator>>()` and `operator<<()` are used for parsing the object parameters from the parameter file, or for printing the parameter data, for example, when debugging.

C.3 Structure of a Felix++ simulation

C.3.1 A skeleton simulation program

A Felix++ simulation is basically a C++ program that includes header files of Felix and/or Felix++. The structure of a Felix++ simulation typically looks similar to the following code fragment:

Code fragment C.4
```
// Part 1: Felix2 declarations
// ------------------------------------
#include "F2_simenv.h"
...                           // further includes (e.g., of Felix++ headers)

#define STANDALONE 1
...                           // further macro definitions
TSimulationEnvironment* senv;
TSSNeuron* popPe;
...                           // declaration of further simulation components

// Part 2: Felix1 (GUI) declarations
// ------------------------------------
extern "C" {
  #include "nn.h"
  ...                 // include of further Felix headers
}
#if STANDALONE
  NO_DISPLAY
#else
  BEGIN_DISPLAY
  ...                 // declaration of Felix GUI (switches, sliders, graphs, etc.)
  END_DISPLAY
  NO_OUTPUT           // Felix1 output mechanisms usually not used
```

```
#endif

// Part 3: main_init()
// ------------------------------------
int main_init() {
  senv = new TSimulationEnvironment(parameterFile);
  ...                       // further utility declarations/creations
  popPe = new TSSNeuron(*senv,"popPe",popLT_Pe,0,integrator1,1);
  ...                       // further creation of objects
  senv->allocate();
  ...                       // assigning of GUI variables
  return 0;
}

// Part 4: init()
// ------------------------------------
int init() {
  senv->init();
  ...                       // further initialization
  return 0;
}

// Part 5: step()
// ------------------------------------
int step() {
  senv->step();
  ...                       // further step()-stuff
  return 0;
}

// Part 6: main()
// ------------------------------------
#if STANDALONE
  int main(int nArgs, char** args) {
    main_init();
    init();
    for(int i=0;i<1000;i++) step();
      ...                   // further stuff, e.g. saving simulation data
  }
#endif
```

In part 1 Felix++ header files (and also other headers) are included, macros are defined such as `STANDALONE` (which switches between online and batch mode; see below parts 2 and 6), and the simulation environment and the simulation components are declared. For the sake of flexibility it is recommended to declare the simulation environment and the components as pointer variables which are allocated in the `main_init()` method (see part 3). For example, if the simulation environment and components would be already *constructed* here, it would not be possible to pass the name of the parameter file as an argument to the simulation program. In this example only the simulation environment `senv` and a neuron population `popPe` of type `TSSNeuron` (see section C.2.3) are declared. Usually the construction of the simulation environment and the simulation components is paralleled with the parsing of a parameter file (see below the code fragment C.5)

In part 2 the graphical user interface (GUI) of the simulation is declared (only necessary for online simulations, i.e., if the flag macro `STANDALONE` is inactive). For this purpose, first the Felix headers (see Fig. C.1) must be included (in `extern ''C''` brackets since Felix has been implemented in C). Then the GUI components of Felix can be specified in the `#else` branch of the `#if` directive (see [197] for details).

Part 3 is the `main_init()` procedure. Here the simulation environment and subsequently the simulation components are created by calling the corresponding constructors. After constructing all simulation components a call to the `allocate()` method of the simulation environment might be necessary (e.g., for simulation components such as `TSSNeuron` employing integrators;

see module *F2_integrator.h/cpp* in section C.1.3; cf. section C.2.3). The `main_init()` procedure is normally called only once at the beginning of the simulation to construct the simulation objects. This is done either by the `main()` procedure (see part 6) for batch simulations (for activated flag macro `STANDALONE=1`) or by the Felix GUI for online simulations (for `STANDALONE=0`).

In part 4 the `init()` procedure is defined. It contains normally at least the call to the `init()` method of the simulation environment, but possibly also further initialization code for the simulation. The `init()` procedure should be called after `main_init()` to initialize the states of the simulation objects before the actual simulation computations start (see part 5). In contrast to `main_init()` the `init()` procedure can be called more than once either from the `main()` procedure (see part 6) for batch simulations or for online simulations by pressing the *init* button (or the *run* button) in the main simulation window [197].

Part 5 is the `step()` procedure which computes one simulation step. It contains normally at least the call to the `step()` method of the simulation environment, but possibly also further code for the simulation. The `step()` procedure is called either from the `main()` procedure (see part 6) for batch simulations or for online simulations by pressing the *step* button (or the *run* button) in the main simulation window.

Part 6 defines the `main()` procedure for batch simulations with activated macro flag `STANDALONE=1` (see part 1). This procedure must contain calls to `main_init()` and `init()` before the calls to the `step()` procedure.

C.3.2 The parameter file

The construction of the simulation environment and the simulation components in `main_init()` (see part 3 in section C.3.1) is usually paralleled by the parsing of the *parameter file* in order to read in the parameters to be used for the respective simulation objects. The following code fragment shows parts of the parameter file for our skeleton simulation program above (code fragment C.4).

Code fragment C.5

```
#{ TSimulationEnvironment Simulation1    % parameter scope for simulation environment
  stepSize      : 0.1
  dataDirectory : /private/aknoblau/simdata/BC2
}

#{ SSNeuron popPe                        % parameter scope for neuron population
   tau_x(exp,sig,min,max)      :  10 0 1 0
   theta(exp,sig,min,max)      :  10 0 1 0
   refAbs(exp,sig,min,max)     :   2 0 1 0
   refRel(exp,sig,min,max)     :  3  0.5 1.75 4.25
   tau_h(exp,sig,min,max)      : 150 0 1 0
   thetaInc_h(exp,sig,min,max) :  0.6 1.0 0.2 1.0

   #{ TMCompartmentReceptors receptors
      nReceptorPorts : 2
      #{ TMOffDynamicsRP AMPA
         tau_OFF(exp,sig,min,max)             :   5 0 1 0
         E(exp,sig,min,max)                   :  80 0 1 0
         g0(exp,sig,min,max)                  :   0 0 1 0
         powerInpNoise(exp,order,sig,min,max) :   0.025 1 0 1 0
         qLen                                 : 600
      }
      #{ TMOffDynamicsRP GABAA
         tau_OFF(exp,sig,min,max)             :   7 0 1 0
         E(exp,sig,min,max)                   : -10 0 1 0
         g0(exp,sig,min,max)                  :   0 0 1 0
         powerInpNoise(exp,order,sig,min,max) :   0.02 1 0 1 0
         qLen                                 : 100
      }
   }
}
```

A parameter file is divided into various *parameter scopes*. A parameter scope is a group of parameters which has been put into scope brackets according to the syntax #{ `<scope type> <scope ID> <parameter1> <parameter2> ...` }. The scope type is given by the class of the object to be parsed, while the scope ID can be chosen arbitrarily.

This parameter file contains two *global* parameter scopes, one for the simulation environment `senv` and another for the neuron population `popPe` (cf. part 3 in code fragment C.4). Parameter scopes can be organized hierarchically: For example scope `SSNeuron popPe` contains a *sub-scope* for the `receptorPorts` object (see part 1 in code fragment C.4) which in turn can contain an arbitrary number of further sub-scopes for different receptor dynamics.

In the example there are two scopes for the receptor dynamics (cf. module *F2_receptor.h/cpp* in section C.1.2) implementing the dynamics of the synaptic conductances of the neuron model described in section 2.4. The parameters in scope `TMOffDynamicsRP AMPA` specify the dynamics of the excitatory conductance g_{ex} (eq. 2.1) which is implemented by the corresponding object of type `TMOffDynamicsRP`. Parameter `tau_OFF` corresponds to τ_{ex} (see eq. 2.1) and parameter `E` corresponds to E_{ex} (see eq. 2.4). The additional parameters determine conductance baseline (`g0`), noise power (`powerInpNoise`), and the queue length (`qLen`; measured in simulation steps) for incoming spikes propagated by connection objects (see class `TConnection` in section C.1.3). The latter parameter determines the maximal possible axonal delay for the connection projecting onto this receptor port.

The parameters in scope `TMOffDynamicsRP GABAA` have the analogous relation to the dynamics of the `inhibitory` conductance g_{in} (see eqs. 2.2 and 2.4).

Many parameters are specified not by a single value but by a vector of four values to define *individual* parameters for each member of a population. Parameter `refRel` (in scope `popPe`), for example, specifies that the parameter R_r of our neuron model (see eq. 2.5 in section 2.4) is distributed according to a Gaussian with mean 3, standard deviation 0.5, but limited to the interval [1.75; 4.25]. In contrast, if the standard deviation is 0 and/or the left interval border larger than the right one then the parameter is the same for all members of the population (see parameter `tau_x`, for example).

C.3.3 Compiling and running simulations

In sections C.3.1 and C.3.2 we have discussed how a Felix++ simulation program and the corresponding parameter file should be structured. Here we explain how one obtains an executable program from the source file. This process is illustrated in Figure C.4.

A simulation source file named *SIMULATION.cpp* can be compiled by the command "`Felix2 SIMULATION`". `Felix2` is a script that compiles the source file and sets the correct include- and link-paths by calling the `Makefile`. Compiling using `Felix2` yields as output the executable *SIMULATION*. When running this requires the dynamic (or shared) libraries as shown in Figure C.4.

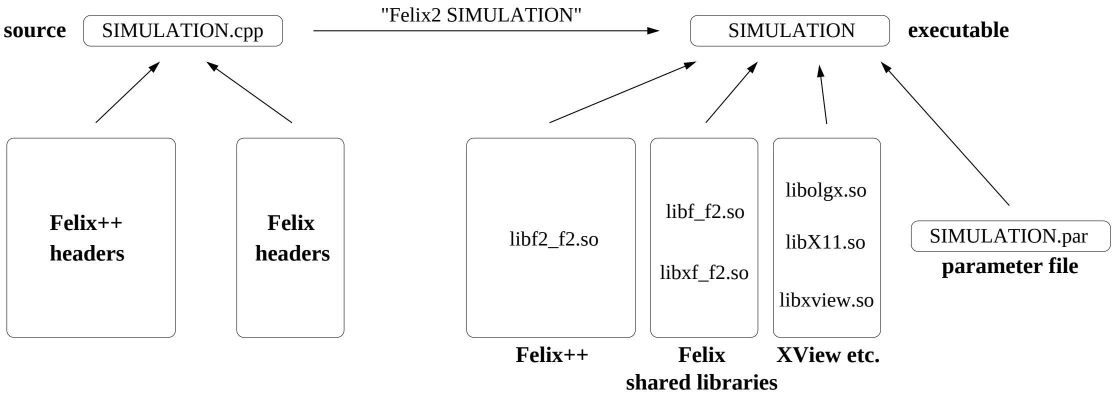


Figure C.4: Compilation and Linkage of a Felix++ simulation program. The simulation source file includes Felix++ and/or Felix headers (cf. Fig. C.1). The source file *SIMULATION.cpp* then is compiled by the command "`Felix2 SIMULATION`" which generates an executable *SIMULATION*. Running the executable *SIMULATION* requires the parameter file and dynamic ("shared") libraries of Felix++, Felix, and the X system.

Appendix D

Symbol reference

Typographical conventions

To save brackets in formulas we use the following conventions

$$a/bc := a/(b \cdot c) \tag{D.1}$$
$$\ln x := \log_e(x) \tag{D.2}$$
$$\mathrm{ld}\, x := \log_2(x) \tag{D.3}$$
$$\ln a/b := (\ln a)/b \tag{D.4}$$

For approximations and asymptotic behavior we use

$$a \approx b \Leftrightarrow a/b = 1 + \epsilon \tag{D.5}$$
$$f(n) \to c \Leftrightarrow \lim_{n \to \infty} f(n) = c \tag{D.6}$$
$$f(n) \asymp g(n) \Leftrightarrow \lim_{n \to \infty} (f(n) - g(n)) = 0 \tag{D.7}$$
$$f(n) \sim g(n) \Leftrightarrow f(n)/g(n) \to c \text{ for } n \to \infty \tag{D.8}$$
$$f(n) = O(g(n)) \Leftrightarrow f(n) < c \cdot g(n) \text{ for all } n > n_0 \tag{D.9}$$

where c, n_0, and $0 < \epsilon \ll 1$ are constants.

Index symbols

μ	pattern or assembly index
i, j	neuron indices

Associative memory

A	binary memory matrix
u, v	pattern vectors (originally used for storing)
$\tilde{u}$	address pattern used for the retrieval
$\hat{u}, \hat{v}$	retrieval result patterns
Θ	threshold
x	neuron potentials
n	length of the pattern vectors, or number of neurons per population
m	length of the pattern vectors for the associated patterns (usually $m = n$)
k	pattern activity: number of active neurons (one-entries) in a pattern
p_1	matrix load: fraction of one-entries in the binary memory matrix
$p_{1,\mathrm{max}}$	maximal matrix load allowed for high-fidelity (often we write simply p_1)
M	number of patterns stored in the memory matrix
M_{max}	maximal M allowed for high-fidelity (often we write simply M)
ϵ	high-fidelity parameter
λ	an address pattern contains λk correct one-entries
κ	an address pattern contains κk false one-entries
z	total number of one-entries in an address pattern, i.e. $z = (\lambda + \kappa)k$
k_1	number of (correct) ones in an address pattern ($k_1 := \lambda k$)
$k_{1,\mathrm{hifi}}$	minimal number of (correct) ones in an address pattern to obtain high-fidelity
p_{01}	error probability of a false one-entry in a given component of the retrieval result
p_{10}	error probability of a missing one in a given component of a retrieved pattern
C_A	absolute storage capacity (i.e., not normalized)
C	storage capacity normalized to the required physical memory or synapse number
C^{sprs}	normalized storage capacity for index representations of sparse memory matrices
C^{cmpr}	normalized storage capacity for optimally compressed memory matrices
t_{ret}	time (or number of steps) required for a retrieval
f_I	information current or information flow: retrieved information per time step
E	information current f_I normalized to a reference information current ${f_I}^{\mathrm{ref}}$
N	number of simultaneously presented address patterns
c^A, c^H, c^Σ	spike counters

Model of cortical areas

R, P, C	cortical areas
R	neuron population of area R
P, P^S, P^T	neuron populations of area P
C, C^S, C^T	neuron populations of area C

Bibliography

[1] L.F. Abbot and S.B. Nelson. Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000.

[2] M. Abeles. *Local cortical circuits.* Springer, Berlin Heidelberg New York, 1982.

[3] M. Abeles. *Corticonics: Neural circuits of the cerebral cortex.* Cambridge University Press, Cambridge UK, 1991.

[4] M. Abeles, H. Bergman, I. Gat, I. Meilijson, E. Seidemann, N. Tishby, and E. Vaadia. Cortical activity flips among quasi stationary states. *Proceedings of the National Academy of Science, USA*, 92:8616–8620, 1995.

[5] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia. Spatio-temporal firing patterns in frontal cortex of behaving monkeys. *Journal of Neurophysiology*, 70:1629–1643, 1993.

[6] A.M.H.J. Aertsen, M. Erb, and G. Palm. Dynamics of functional coupling in the cerebral cortex: an attempt at a model-based interpretation. *Physica D*, 75:103–128, 1994.

[7] A.M.H.J. Aertsen, M. Erb, G. Palm, and A. Schüz. Coherent assembly dynamics in the cortex: multi-neuron recordings, network simulations and anatomical considerations. In *Proceedings Symposium Oscillatory Event Related Brain Dynamics*. Plenum, 1994.

[8] M. Agu, K. Yamanaka, and H. Takahashi. A local property of the phasor model of neural networks. *IEICE Transactions on Information and Systems*, vol.E79–D, no.8:1209–1211, 1996.

[9] C. Allen and C.F. Stevens. An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Science USA*, 91:10380–10383, 1994.

[10] J. Anderson, I. Lampl, I. Reichova, M. Carandini, and D. Ferster. Stimulus dependence of two-state fluctuations of membrane potential in cat visual cortex. *Nature Neuroscience*, 3(6):617–621, 2000.

[11] A. Arieli, A. Sterkin, A. Grinvald, and A. Aertsen. Dynamics of ongoing activity: Explanation of the large variability in evoked cortical responses. *Science*, 273:1868–1871, 1996.

[12] H.B. Barlow. Single units and sensation: a neuron doctrine for perceptual psychology. *Perception*, 1:371–394, 1972.

[13] A.P. Bartsch and J.L. van Hemmen. Combined Hebbian development of geniculocortical and lateral connectivity in a model of primary visual cortex. *Biological Cybernetics*, 84:41–55, 2001.

[14] H.J. Bentz, M. Hagstroem, and G. Palm. Information storage and effective data retrieval in sparse matrices. *Neural Networks*, 2:289–293, 1989.

[15] G. Bi and M. Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, 18(24):10464–10472, 1998.

[16] A. Bibbig. *Oszillationen, Synchronisation, Mustertrennung und Hebb'sches Lernen in Netzwerken aus erregenden und hemmenden Neuronen (in German).* PhD thesis, Department of Neural Information Processing, University of Ulm, Germany, 2000.

[17] A. Bibbig, H.J. Faulkner, M.A. Whittington, and R.D. Traub. Self-organized synaptic plasticity contributes to the shaping of γ and β oscillations in vitro. *The Journal of Neuroscience*, 21(22):9053–9067, 2001.

[18] R. Blake and N. Logothetis. Visual competition. *Nature Reviews Neuroscience*, 3:1–12, 2002.

[19] T.V.P. Bliss and G.L. Collingridge. A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361:31–39, 1993.

[20] T. Bonhoeffer and A. Grinvald. Iso-orientation domains in cat visual cortex are arranged in pinwheel-like patterns. *Nature*, 353:429–431, 1991.

[21] T. Bonhoeffer and A. Grinvald. The layout of iso-orientation domains in area 18 of cat visual cortex: optical imaging reveals a pinwheel-like organization. *Journal of Neuroscience*, 13:4157–4180, 1993.

[22] E.G. Boring. A new ambiguous figure. *American Journal of Psychology*, 42:444, 1930.

[23] G.N. Borisyuk, R.M. Borisyuk, and Y.B. Kazanovich. Temporal structure of neural activity and models of information processing in the brain. In Miller [118], chapter 13, pages 331–349.

[24] H. Bosch and F. Kurfess. Information storage capacity of incompletely connected associative memories. *Neural Networks*, 11(5):869–876, 1998.

[25] J.M. Bower and C. Beeman, editors. *The book of GENESIS: Exploring realistic neural models with the GEneral NEural SImulation System.* Springer, New York, 1995.

[26] V. Braitenberg. Cell assemblies in the cerebral cortex. In R. Heim and G. Palm, editors, *Lecture notes in biomathematics (21). Theoretical approaches to complex systems.*, pages 171–188. Springer-Verlag, Berlin Heidelberg New York, 1978.

[27] V. Braitenberg and A. Schüz. *Anatomy of the cortex. Statistics and geometry.* Springer-Verlag, Berlin, 1991.

[28] I.N. Bronstein and K.A. Semendjajew. *Taschenbuch der Mathematik (in German).* Nauka, Teubner, Deutsch, Moscow, Stuttgart, Thun, 1991.

[29] J. Buckingham and D. Willshaw. Performance characteristics of associative nets. *Network: Computation in Neural Systems*, 3:407–414, 1992.

[30] J. Buckingham and D. Willshaw. On setting unit thresholds in an incompletely connected associative net. *Network: Computation in Neural Systems*, 4:441–459, 1993.

[31] J.T. Buckingham. Delicate nets, faint recollections: a study of partially connected associative network memories. *PhD thesis, University of Edingburgh*, 1991.

[32] S. Cash and R. Yuste. Linear summation of excitatory inputs by CA1 pyramidal neurons. *Neuron*, 22:383–394, 1999.

[33] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms.* MIT Press, Cambridge, MA, 1990.

[34] T.M. Cover and J.A. Thomas. *Elements of information theory.* Wiley, New York, 1991.

[35] F. Crick and C. Koch. Towards a neurobiological theory of consciousness. *Sem. Neurosci.*, 2:263–275, 1990.

[36] H.H. Dale. Pharmacology and the nerve endings. *Proceedings of the Royal Society of Medicine*, 28:319–332, 1935.

[37] R. Desimone. Face-selective cells in the temporal cortex of monkeys. *Journal of Cognitive Neuroscience*, 3(1):1–8, 1991.

[38] A. Destexhe, Z.F. Mainen, and T.J. Sejnowski. Kinetic models of synaptic transmission. In Koch and Segev [102], chapter 1, pages 1–25.

[39] J.C. Eccles. *Physiology of synapses.* Springer, Berlin, 1964.

[40] R. Eckhorn. Oscillatory and non-oscillatory synchronizations in the visual cortex and their possible roles in associations of visual features. In J.van Pelt, M.A. Corner, H.B.M. Uylings, and F.H. Lopes da Silva, editors, *Progress in Brain Research*, volume 102, chapter 28, pages 405–426. Elsevier Science BV, 1994.

[41] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, and H.J. Reitboeck. Coherent Oscillations: A mechanism of feature linking in the visual cortex? *Biol. Cybern.*, 60:121–130, 1988.

[42] R. Eckhorn, A. Bruns, M. Saam, A. Gail, A. Gabriel, and H.J. Brinksmeyer. Flexible cortical gamma-band correlations suggest neural principles of visual processing. *Visual Cognition*, 8(3/4/5):519–530, 2001.

[43] A.K. Engel, P. König, A.K. Kreiter, and W. Singer. Interhemispheric synchronization of oscillatory neuronal responses in cat visual cortex. *Science*, 252:1177–1179, 1991.

[44] A.K. Engel, P. König, and W. Singer. Direct physiological evidence for scene segmentation by temporal coding. *Proc. Natl. Acad. Sci. USA*, 88:9136–9140, 1991.

[45] A.K. Engel, A.K. Kreiter, P. König, and W. Singer. Synchronization of oscillatory neuronal responses between striate and extrastriate visual cortical areas of the cat. *Proc. Natl. Acad. Sci. USA*, 88:6048–6052, 1991.

[46] M. Fahle and G. Palm. Perceptual rivalry between illusory and real contours. *Biological Cybernetics*, 66:1–8, 1991.

[47] M.J. Farah and T.E. Feinberg, editors. *Patient-based approaches to cognitive neuroscience.* MIT-Press, Cambridge, MA, 2000.

[48] M.J. Farah and T.E. Feinberg. Visual object agnosia. In *Patient-based approaches to cognitive neuroscience.* [47], chapter 6, pages 79–84.

[49] T.E. Feinberg and M.J. Farah. A historical perspective on cognitive neuroscience. In Farah and Feinberg [47], chapter 1, pages 3–20.

[50] D.J. Felleman and D.C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, 1:1–47, 1991.

[51] J.A. Fodor and Z.W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:3–71, 1988.

[52] A. Frien, R. Eckhorn, R. Bauer, T. Woelbern, and H. Kehr. Stimulus-specific fast oscillations at zero phase between visual areas V1 and V2 of awake monkey. *NeuroReport*, 5(17):2273–2277, 1994.

[53] R.C. Froemke and Y. Dan. Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature*, 416:433–438, 2002.

[54] A.P. Georgopoulos, J.F. Kalaska, R. Caminiti, and J.T. Massey. On the relations between the directions of two-dimensional arm movements and cell discharge in primate motor cortex. *Journal of Neuroscience*, 2:1527–1537, 1982.

[55] W. Gerstner, R. Ritz, and J.L. van Hemmen. A biologically motivated and analytically soluble model of collective oscillations in the cortex, I. Theory of weak locking. *Biological Cybernetics*, 68:363–374, 1993.

[56] G.M. Ghose and J. Maunsell. Specialized representations in visual cortex: A role for binding? *Neuron*, 24:79–85, 1999.

[57] S.W. Golomb. Run-length encodings. *IEEE Transactions on Information Theory*, 12:399–401, 1966.

[58] P. Gouras. Color vision. In Kandel et al. [82], chapter 31, pages 467–480.

[59] C.M. Gray. The temporal correlation hypothesis of visual feature integration: Still alive and well. *Neuron*, 24:31–47, 1999.

[60] C.M. Gray, P. König, A.K. Engel, and W. Singer. Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature*, 338:334–337, 1989.

[61] C.M. Gray and W. Singer. Stimulus specific neural oscillations in orientation columns of cat visual cortex. *Proceedings of the National Academy of Sciences, USA*, 86:1698–1702, 1989.

[62] D. Hammerstrom. A VLSI architecture for high-performance, low-cost, on-chip learning. In *Proceedings of the IEEE International Joint Conference on Neural Networks 1990*, pages II:537–543. IEEE Press, 1990.

[63] N.R. Hardingham and A.U. Larkman. The reliability of excitatory synaptic transmission in slices of rat visual cortex *in vitro* is temperature dependent. *Journal of Physiology (London)*, 507:249–256, 1998.

[64] D.O. Hebb. *The organization of behavior. A neuropsychological theory.* Wiley, New York, 1949.

[65] B. Hellwig. A quantitative analysis of the local connectivity between pyramidal neurons in layers 2/3 of the rat visual cortex. *Biological Cybernetics*, 82:111–121, 2000.

[66] M.L. Hines and N.T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[67] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology (London)*, 117:500–544, 1952.

[68] K.P. Hoffmann and C. Wehrhahn. Zentrale Sehsysteme (in German). In J. Dudel, R. Menzel, and R.F. Schmidt, editors, *Neurowissenschaft (in German)*, chapter 18, pages 405–426. Springer-Verlag, Berlin/New York, 1996.

[69] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science, USA*, 79:2554–2558, 1982.

[70] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science, USA*, 81(10):3088–3092, 1984.

[71] J.J. Hopfield and D.W. Tank. Computing with neural circuits. *Science*, 233:625–633, 1986.

[72] D. Horn, D. Sagi, and M. Usher. Segmentation, binding, and illusory conjunctions. *Neural Computation*, 3:510–525, 1991.

[73] D.H. Hubel and T.N. Wiesel. Functional architecture of macaque monkey visual cortex. *Proceedings of the Royal Society of London Series B*, 198:1–59, 1977.

[74] D.A. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40:1098–1101, 1952.

[75] R.C. James. Photo of a dalmatian., 1966. in: R.L. Gregory. The intelligent eye. McGraw-Hill, New York, 1973.

[76] W. James. *Psychology.* Holt, New York, 1890.

[77] O. Jensen and J.E. Lisman. Novel lists of 7+/-2 known items can be reliably stored in an oscillatory short-term memory network: Interaction with long-term memory. *Learning and Memory*, 3:257–263, 1996.

[78] W. Kahle. *Nervensystem und Sinnesorgane.* Thieme Verlag, Stuttgart, 1991.

[79] E.R. Kandel. Brain and behavior. In Kandel et al. [82], chapter 1, pages 5–17.

[80] E.R. Kandel. Nerve cells and behavior. In Kandel et al. [82], chapter 2, pages 18–32.

[81] E.R. Kandel. Perception of motion, depth, and form. In Kandel et al. [82], chapter 30, pages 440–466.

[82] E.R. Kandel, J.H. Schwartz, and T.M. Jessell, editors. *Principles of neural science.* Elsevier, New York, 1991.

[83] C.U.A. Kappers, G.C. Huber, and E.C. Crosby. *The comparative anatomy of the nervous system of vertebrates, including man.* Macmillan & Co., Ltd., 1936.

[84] S. Kastner and L.G. Ungerleider. Mechanisms of visual attention in the human cortex. *Annu.Rev.Neurosci.*, 23:315–341, 2000.

[85] T. Kenet, D. Bibitchkov, M. Tsodyks, A. Grinvald, and A. Arieli. Spontaneously emerging cortical representations of visual attributes. *Nature*, 425:954–956, 2003.

[86] A. Knoblauch. Assoziativspeicher aus spikenden Neuronen und Synchronisation im visuellen Kortex (in German). *Diploma thesis, Department of Neural Information Processing, University of Ulm, Germany*, 1999.

[87] A. Knoblauch. Optimal matrix compression yields storage capacity 1 for binary Willshaw associative memory. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *Artificial Neural Networks and Neural Information Processing - ICANN/ICONIP 2003.*, LNCS 2714, pages 325–332. Springer Verlag, Berlin, 2003.

[88] A. Knoblauch. *Felix++ Manual.* in preparation, 2004.

[89] A. Knoblauch and G. Palm. Pattern separation and synchronization in spiking associative memories and visual areas. *Neural Networks*, 14:763–780, 2001.

[90] A. Knoblauch and G. Palm. Spiking associative memory and scene segmentation by synchronization of cortical activity. In S. Wermter, J. Austin, and D. Willshaw, editors, *Emerging Neural Computational Architectures Based on Neuroscience*, pages 407–427. Springer-Verlag, 2001.

[91] A. Knoblauch and G. Palm. Scene segmentation by spike synchronization in reciprocally connected visual areas. I. Local effects of cortical feedback. *Biological Cybernetics*, 87(3):151–167, 2002.

[92] A. Knoblauch and G. Palm. Scene segmentation by spike synchronization in reciprocally connected visual areas. II. Global assemblies and synchronization on larger space and time scales. *Biological Cybernetics*, 87(3):168–184, 2002.

[93] A. Knoblauch and G. Palm. Attentional effects in a model of two reciprocally connected visual areas. In F. Detje, D. Dörner, and H. Schaub, editors, *The logic of cognitive systems. Proceedings of the fifth international conference on cognitive modeling.*, pages 271–272. Universitätsverlag Bamberg, 2003.

[94] A. Knoblauch and G. Palm. Binding and synchronization in reciprocally connected cortical areas. In N. Elsner and H. Zimmermann, editors, *The neurosciences from basic research to therapy. Proceedings of the 29th Göttingen Neurobiology Conference and the 5th meeting of the German Neuroscience Society 2003*, pages 1045–1046. Thieme, Stuttgart, 2003.

[95] A. Knoblauch and G. Palm. Cortical assemblies of language areas: Development of cell assembly model for Broca/Wernicke areas. Technical report, Department of Neural Information Processing, University of Ulm, 2003. Report 5 of the MirrorBot project of the European Union.

[96] A. Knoblauch and G. Palm. Synchronization of neuronal assemblies in reciprocally connected cortical areas. *Theory in Biosciences*, 122:37–54, 2003.

[97] A. Knoblauch and F.T. Sommer. Synaptic plasticity, conduction delays, and inter-areal phase relations of spike activity in a model of reciprocally connected areas. *Neurocomputing*, 52–54:301–306, 2003.

[98] A. Knoblauch and F.T. Sommer. Spike-timing-dependent synaptic plasticity can form "zero lag" links for cortical oscillations. *to appear in Neurocomputing (also in Proceedings of CNS 2003)*, 2004.

[99] A. Knoblauch, F.T. Sommer, and G. Palm. Phase relations of cortical oscillators coupled reciprocally by delayed connections. *in preparation*, 2003.

[100] A. Knoblauch, F.T. Sommer, and G. Palm. Spike-timing-dependent synaptic plasticity as a mechanism for long-range synchronization. *in preparation*, 2003.

[101] A. Knoblauch, T. Wennekers, and F.T. Sommer. Is voltage-dependent synaptic transmission in NMDA receptors a robust mechanism for working memory? *Neurocomputing*, 44-46:19–24, 2002.

[102] C. Koch and I. Segev, editors. *Methods in neuronal modeling.* MIT Press, Cambridge, Massachusetts, 1998.

[103] T. Kohonen. Correlation Matrix memories. *IEEE Transactions on Computers*, C-21:353–359, 1972.

[104] T. Kohonen. *Associative memory: a system theoretic approach.* Springer, Berlin, 1977.

[105] T. Kohonen. *Self-organization and associative memory.* Springer, Berlin, 1988.

[106] P. König, A.K. Engel, P.R. Roelfsema, and W. Singer. How precise is neuronal synchronization? *Neural Computation*, 7:469–485, 1995.

[107] B. Kosko. Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:49–60, 1988.

[108] A.K. Kreiter and W. Singer. Stimulus-dependent synchronization of neuronal responses in the visual cortex of the awake macaque monkey. *J. of Neurophys.*, 16(7):2381–2396, 1996.

[109] N.K. Logothetis, D.A. Leopold, and D.L. Sheinberg. What is rivalling during binocular rivalry? *Nature*, 380:621–624, 1996.

[110] M. London and I. Segev. Synaptic scaling in vitro and in vivo. *Nature Neuroscience*, 4(9):853–854, 2001.

[111] R.J. MacGregor. *Neural and Brain Modeling.* Academic Press, San Diego, 1987.

[112] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275:213–215, 1997.

[113] D. Marr. *Vision.* W.H.Freeman, New York, 1982.

[114] J.L. Marroquin. Human visual perception of structure. *Master thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, 1976.

[115] A. Mason, A. Nicoll, and K. Stratford. Synaptic transmission between individual pyramidal neurons of the rat visual vortex in *vitro*. *The Journal of Neuroscience*, 11(1):72–84, 1991.

[116] C. Mason and E.R. Kandel. Central visual pathways. In Kandel et al. [82], chapter 29, pages 420–439.

[117] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[118] R. Miller, editor. *Time and the Brain*. Conceptual Advances in Brain research. Harwood Academic Publishers, Amsterdam, 2000.

[119] M.L. Minsky and S. Papert. *Perceptrons: An introduction to computational geometry.* MIT Press, Cambridge, MA, 1969.

[120] U. Mitzdorf. Current source-density method and application in cat cerebral cortex: investigation of evoked potentials and EEG phenomena. *Physiological Reviews*, 65(1):37–100, 1985.

[121] J. Moran and R. Desimone. Selective attention gates visual processing in the extrastriate cortex. *Science*, 229:782–784, 1985.

[122] M.H.J. Munk, L.G. Nowak, J.I. Nelson, and J. Bullier. Structural basis of cortical synchronization. II. Effects of cortical lesions. *Journal of Neurophysiology*, 74(6):2401–2414, 1995.

[123] J.I. Nelson, P.A. Salin, M.H.J. Munk, M. Arzi, and J. Bullier. Spatial and temporal coherence in corticocortical connections: a cross-correlation study in areas 17 and 18 in the cat. *Visual Neuroscience*, 9:21–38, 1992.

[124] H. Neumann and W. Sepp. Recurrent V1-V2 interaction in early visual boundary processing. *Biological Cybernetics*, 81:425–444, 1999.

[125] L.G. Nowak and J. Bullier. Cross correlograms for neuronal spike trains. Different types of temporal correlation in neocortex, their origin and significance. In Miller [118], chapter 2, pages 53–96.

[126] L.G. Nowak, M.H.J. Munk, A.C. James, P. Girard, and J. Bullier. Cross-correlation study of the temporal interactions between areas V1 and V2 of macaque monkey. *J. Neurophysiol.*, 81:1057–1074, 1999.

[127] L.G. Nowak, M.H.J. Munk, J.I. Nelson, A.C. James, and J. Bullier. Structural basis of cortical synchronization. I.Three types of interhemispheric coupling. *Journal of Neurophysiology*, 74(6):2379–2400, 1995.

[128] M.W. Oram and D.I. Perrett. Modeling visual recognition from neurobiological constraints. *Neural Networks*, 7:945–972, 1994.

[129] G. Palm. On associative memories. *Biological Cybernetics*, 36:19–31, 1980.

[130] G. Palm. *Neural Assemblies. An Alternative Approach to Artificial Intelligence.* Springer, Berlin, 1982.

[131] G. Palm. Associative memory and threshold control in neural networks. In J.L. Casti and A. Karlqvist, editors, *Real Brains - Artificial Minds.* North-Holland, New York, Amsterdam, London, 1987.

[132] G. Palm. Computing with neural networks. *Science*, 235:1227–1228, 1987.

[133] G. Palm. Cell assemblies as a guideline for brain research. *Concepts in Neuroscience*, 1:133–148, 1990.

[134] G. Palm. Memory capacities of local rules for synaptic modification. A comparative review. *Concepts in Neuroscience*, 2:97–128, 1991.

[135] G. Palm. Cell assemblies, coherence, and corticohippocampal interplay. *Hippocampus*, 3:219–226, 1993.

[136] G. Palm. On the internal structure of cell assemblies. In A. Aertsen, editor, *Brain Theory*. Elsevier, Amsterdam, 1993.

[137] C.C.H. Petersen, R.C. Malenka, R.A. Nicoll, and J.J. Hopfield. All-or-none potentiation at CA3-CA1 synapses. *Proceedings of the National Academy of Science, USA*, 95:4732–4737, 1998.

[138] P.F. Pinsky and J. Rinzel. Intrinsic and network rhythmogenesis in a reduced Traub model for CA3 neurons. *Journal of Computational Neuroscience*, 1:39–60, 1994.

[139] E. Pöppel. Temporal mechanisms in perception. *International Review of Neurobiology*, 37:185–202, 1994.

[140] E. Pöppel. *Grenzen des Bewußtseins. Wie kommen wir zur Zeit, und wie entsteht Wirklichkeit?* Insel Verlag, Frankfurt, 1997.

[141] E. Pöppel and K. Schill. Time perception: problems of representation and processing. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 987–990. MIT Press, Cambridge, MA., 1995.

[142] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C.* Cambridge University Press, Cambridge, UK, 1992.

[143] F. Pulvermüller. Words in the brain's language. *Behavioral and Brain Sciences*, 22:253–336, 1999.

[144] D.A. Rachkovskij and E.M. Kussul. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13:411–452, 2001.

[145] W. Rall and H. Agmon-Snir. Cable theory for dendritic neurons. In Koch and Segev [102], chapter 2, pages 27–92.

[146] J.H. Reynolds, L. Chelazzi, and R. Desimone. Competitive mechanisms subserve attention in macaque areas V2 and V4. *The Journal of Neuroscience*, 19(5):1736–1753, 1999.

[147] J.H. Reynolds and R. Desimone. The role of neural mechanisms of attention in solving the binding problem. *Neuron*, 24:19–29, 1999.

[148] M. Riesenhuber and T. Poggio. Are cortical models really bound by the "binding problem"? *Neuron*, 24:87–93, 1999.

[149] D.L. Ringach. States of mind. *Nature*, 425:912–913, 2003.

[150] R. Ritz, W. Gerstner, U. Fuentes, and J.L. van Hemmen. A biologically motivated and analytically soluble model of collective oscillations in the cortex. II. Applications to binding and pattern segmentation. *Biol. Cybern.*, 71:349–358, 1994.

[151] A. Robert. From contour integration to image schemas: a modern perspective on Gestalt psychology. Technical Report 97.02, Department of Cognitive Science, University of California at San Diego, 1997.

[152] A.J. Rockel, R.W. Hiorns, and T.P.S. Powell. The basic uniformity of the neocortex. *Brain*, 103:221–244, 1980.

[153] R. Rojas. *Theorie der neuronalen Netze.* Springer Verlag, Berlin, 1993.

[154] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms.* Spartan Books, Washington, D.C., 1961.

[155] D. Rumelhart and J. McClelland. *Parallel distributed processing..* MIT Press, Cambridge, MA, 1986.

[156] S. Russel and P. Norvig. *Artificial intelligence.* Prentice-Hall, London, 1995.

[157] B. Sagi, S.C. Nemat-Nasser, R. Kerr, R. Hayek, C. Downing, and R. Hecht-Nielsen. A biologically motivated solution to the cocktail party problem. *Neural Computation*, 13(7):1575–1602, 2001.

[158] C.D. Salzman and Newsome W.T. Neural mechanisms for forming a perceptual decision. *Science*, 264:231–237, 1994.

[159] U. Schöning. *Theoretische Informatik kurz gefasst (in German).* BI Wissenschaftsverlag, Mannheim, 1992.

[160] F. Schwenker, F.T. Sommer, and G. Palm. Iterative retrieval of sparsely coded associative memory patterns. *Neural Networks*, 9:445–455, 1996.

[161] I. Segev, R.E. Burke, and M. Hines. Compartmental models of complex neurons. In Koch and Segev [102], chapter 3, pages 93–136.

[162] M.I. Sereno, A.M. Dale, J.B. Reppas, K.K. Kwong, J.W. Belliveau, T.J. Brady, B.R. Rosen, and R.B.H. Tootell. Borders of multiple visual areas in humans revealed by functional MRI. *Science*, 268:889–893, 1995.

[163] M.N. Shadlen and J.A. Movshon. Sychrony unbound: A critical evaluation of the temporal binding hypothesis. *Neuron*, 24:67–77, 1999.

[164] C.E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois Press, Urbana/Chicago, 1949.

[165] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning: a connectionistic representation of rules, variables and dynamic bindings. *Behavioral and Brain Sciences*, 16(3):417–494, 1993.

[166] D.L. Sheinberg and N.K. Logothetis. The role of temporal cortical areas in perceptual organization. *Proceedings of the National Academy of Sciences, USA*, 94:3408–3413, 1997.

[167] W. Singer. Neuronal synchrony: A versatile code for the definition of relations? *Neuron*, 24:49–65, 1999.

[168] W. Singer and C.M. Gray. Visual feature integration and the temporal correlation hypothesis. *Annu.Rev.Neurosci.*, 18:555–586, 1995.

[169] F.T. Sommer. On cell assemblies in a cortical column. *Neurocomputing*, 32-33:517–522, 2000.

[170] F.T. Sommer and G. Palm. Bidirectional retrieval from associative memory. In *Advances in Neural Information Processing Systems 10 (NIPS 1997)*, pages 675–681. MIT Press, Cambridge, MA., 1998.

[171] F.T. Sommer and G. Palm. Improved bidirectional retrieval of sparse patterns stored by hebbian learning. *Neural Networks*, 12:281–297, 1999.

[172] F.T. Sommer and T. Wennekers. Associative memory in networks of spiking neurons. *Neural Networks*, 14(6/7):825–834, 2001.

[173] S. Song, K.D. Miller, and L.F. Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.

[174] K. Steinbuch. Die Lernmatrix. *Kybernetik*, 1:36–45, 1961.

[175] M. Steriade and F. Amzica. Intracortical and corticothalamic coherency of fast spontaneous oscillations. *Proc. Natl. Acad. Sci. ,USA*, 93:2533–2538, 1996.

[176] M. Steriade, F. Amzica, and D. Contreras. Synchronization of fast (30-40 Hz) spontaneous cortical rhythms during brain activation. *J.Neurosci.*, 16(1):392–417, 1996.

[177] A. Strey. EpsiloNN - A specification language for the Efficient Parallel Implementation of Neural Networks. In *Biological and artificial computation: From neuroscience to technology.*, Lecture Notes in Computer Science 1240, pages 714–722. Springer, Berlin, 1997.

[178] B. Stroustrup. *C++ programming language. Third Edition.* Addison-Wesley, Reading, Massachusetts, 1997.

[179] H. Swadlow. Information flow along neocortical axons. In Miller [118], chapter 4, pages 131–155.

[180] T. Swan. *GNU C++ for Linux.* Que, Indianapolis, 1999.

[181] J. Szentagothai. The 'module concept' in cerebral cortex architecture. *Brain Research*, 95:475–496, 1975.

[182] J. Szentagothai. The neuron network of the cerebral cortex: a functional interpretation. *Proceedings of the Royal Society of London Series B*, 201:219–248, 1978.

[183] S.J. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520–522, 1996.

[184] M.J. Tovée. *An introduction to the visual system.* Cambridge University Press, Cambridge, 1996.

[185] R.D. Traub, R.K.S. Wong, R. Miles, and H. Michelson. A model of a CA3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. *Journal of Neurophysiology*, 66(2):635–650, 1991.

[186] A. Treisman. Feature binding, attention and object perception. *Phil. Trans. R. Soc. London B*, 353:1295–1306, 1998.

[187] A. Treisman. Solutions to the binding problem: progress through controversy and convergence. *Neuron*, 24:105–110, 1999.

[188] A. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[189] M.V. Tsodyks and M.V. Feigelman. The enhanced storage capacity in neural networks with low activity level. *Europhysics Letters*, 6:101–105, 1988.

[190] M.V. Tsodyks, T. Kenet, A. Grinvald, and A. Arieli. Linking spontaneous activity of single cortical neurons and the underlying functional architecture. *Science*, 286:1943–1946, 1999.

[191] C. von der Malsburg. The correlation theory of brain function. Technical Report Interner Report 81-2, Max-Planck Institut für Biophysikalische Chemie, Göttingen, Germany, 1981.

[192] C. von der Malsburg. Am I thinking assemblies? In G. Palm and A. Aertsen, editors, *Brain Theory*, pages 161–176. Springer-Verlag, Berlin/Heidelberg, 1986.

[193] C. von der Malsburg. Binding in models of perception and brain function. *Current Opinion in Neurobiology*, 5:520–526, 1995.

[194] C. von der Malsburg. The what and why of binding: The modeler's perspective. *Neuron*, 24:95–104, 1999.

[195] C. von der Malsburg and J. Buhmann. Sensory segmentation with coupled neural oscillators. *Biological Cybernetics*, 67:233–242, 1992.

[196] C. von der Malsburg and W. Schneider. A neural cocktail-party processor. *Biological Cybernetics*, 54:29–40, 1986.

[197] T. Wennekers. *Synchronisation und Assoziation in Neuronalen Netzen (in German)*. Shaker Verlag, Aachen, 1999.

[198] T. Wennekers and G. Palm. Controlling the speed of synfire chains. In C.v.d. Malsburg, W.v. Seelen, J.C. Vorbrüggen, and B. Sendhoff, editors, *Proceedings of the ICANN 1996*, pages 451–456, Berlin, Heidelberg, New York, 1996. Springer Verlag.

[199] T. Wennekers and G. Palm. On the relation between neural modelling and experimental neuroscience. *Theory in Bioscience*, 116:273–289, 1997.

[200] T. Wennekers and G. Palm. How imprecise is neuronal synchronization. *Neurocomputing*, 26-27:579–585, 1999.

[201] T. Wennekers and G. Palm. Cell assemblies, associative memory and temporal structure in brain signals. In Miller [118], chapter 10, pages 251–273.

[202] T. Wennekers, F.T. Sommer, and G. Palm. Iterative retrieval in associative memories by threshold control of different neural models. In H.J. Herrmann, D.W. Wolf, and E. Pöppel, editors, *Supercomputing in Brain Research: From Tomography To Neural Networks*, pages 301–319. World Scientific, Singapore, 1995.

[203] J.R. Wickens and R. Miller. A formalisation of the neural assembly concept. 1.Constraints on neural assembly size. *Biological Cybernetics*, 77:351–358, 1997.

[204] D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, 1969.

[205] J.M. Wolfe and K.R. Cave. The psychophysical evidence for a binding problem in human vision. *Neuron*, 24:11–17, 1999.

[206] C. Zetzsche. Sparse coding: the link between low level vision and associative memory. In R. Eckmiller, G. Hartmann, and G. Hauske, editors, *Parallel processing in neural systems and computers.*, pages 273–276. Elsevier, North-Holland, 1990.

Lebenslauf und wissenschaftlicher Werdegang

Persönliche Angaben

Name	Andreas Knoblauch
Geburtsdatum	25. August 1972
Geburtsort	Geislingen an der Steige
Staatsangehörigkeit	deutsch
Familienstand	verheiratet mit Kattia Chaves Monge, BBA
Eltern	Hermann Knoblauch, Konditormeister Gertrud Knoblauch (geb. Winkler), Einzelhandelskauffrau
Geschwister	Stefanie Knoblauch, Erzieherin

Ausbildung

1979 - 1983	Grundschule in Böhmenkirch
1983 - 1992	Helfensteingymnasium in Geislingen an der Steige
1992 - 1993	Zivildienst an der Sozialstation Donzdorf
1993 - 2000	Studium der Informatik mit Nebenfach Elektrotechnik an der Universität Ulm
seit 2000	Promotionsstudium an der Abteilung Neuroinformatik der Universität Ulm

Prüfungen

19.05.1992	Abitur
27.09.1995	Vordiplom in Informatik mit Nebenfach Elektrotechnik
14.01.2000	Diplom in Informatik mit Nebenfach Elektrotechnik

Berufstätigkeit

1988 - 2000	Computeradministration für Lebensmittelmarkt Knoblauch in Böhmenkirch
1995 - 2000	Wissenschaftliche Hilfskraft an der Universität Ulm
seit 2000	Wissenschaftlicher Angestellter an der Abteilung Neuroinformatik der Universität Ulm