

Biological and Medical Physics, Biomedical Engineering

Thomas Lindblad
Jason M. Kinser

Image Processing Using Pulse-Coupled Neural Networks

Applications in Python

Third Edition

 Springer

Biological and Medical Physics, Biomedical Engineering

For further volumes:
<http://www.springer.com/series/3740>

The fields of biological and medical physics and biomedical engineering are broad, multidisciplinary and dynamic. They lie at the crossroads of frontier research in physics, biology, chemistry, and medicine. The Biological and Medical Physics, Biomedical Engineering Series is intended to be comprehensive, covering a broad range of topics important to the study of the physical, chemical and biological sciences. Its goal is to provide scientists and engineers with textbooks, monographs, and reference works to address the growing need for information.

Books in the series emphasize established and emergent areas of science including molecular, membrane, and mathematical biophysics; photosynthetic energy harvesting and conversion; information processing; physical principles of genetics; sensory communications; automata networks, neural networks, and cellular automata. Equally important will be coverage of applied aspects of biological and medical physics and biomedical engineering such as molecular electronic components and devices, biosensors, medicine, imaging, physical principles of renewable energy production, advanced prostheses, and environmental control and engineering.

Editor-in-Chief:

Elias Greenbaum, Oak Ridge National Laboratory, Oak Ridge, TN, USA

Editorial Board

Masuo Aizawa, Department of Bioengineering, Tokyo Institute of Technology, Yokohama, Japan

Olaf S. Andersen, Department of Physiology, Biophysics and Molecular Medicine Cornell University New York, NY, USA

Robert H. Austin, Department of Physics, Princeton University, Princeton, NJ, USA

James Barber, Department of Biochemistry, Imperial College of Science, Technology and Medicine London, UK

Howard C. Berg, Department of Molecular and Cellular Biology, Harvard University Cambridge, MA, USA

Victor Bloomfield, Department of Biochemistry University of Minnesota, St. Paul, MN, USA

Robert Callender, Department of Biochemistry Albert Einstein College of Medicine Bronx, NY, USA

Britton Chance

Steven Chu, Lawrence Berkeley National Laboratory Berkeley, CA, USA

Louis J. DeFelice, Department of Pharmacology Vanderbilt University, Nashville, TN, USA

Johann Deisenhofer, Howard Hughes Medical Institute The University of Texas Dallas, TX, USA

George Feher, Department of Physics, University of California, San Diego, La Jolla, CA, USA

Hans Frauenfelder, Los Alamos National Laboratory Los Alamos, NM, USA

Ivar Giaever, Rensselaer Polytechnic Institute Troy, NY, USA

Sol M. Gruner, Cornell University Ithaca, NY, USA

Judith Herzfeld, Department of Chemistry Brandeis University, Waltham, MA, USA

Mark S. Humayun, Doheny Eye Institute Los Angeles, CA, USA

Pierre Joliot, Institute de Biologie Physico-Chimique Fondation Edmond de Rothschild, Paris, France

Lajos Keszthelyi, Institute of Biophysics, Hungarian Academy of Sciences, Szeged, Hungary

Robert S. Knox, Department of Physics and Astronomy, University of Rochester Rochester, NY, USA

Aaron Lewis, Department of Applied Physics Hebrew University, Jerusalem, Israel

Stuart M. Lindsay, Department of Physics and Astronomy, Arizona State University, Tempe, AZ, USA

David Mauzerall, Rockefeller University New York, NY, USA

Eugenie V. Mielczarek, Department of Physics and Astronomy, George Mason University Fairfax, VA, USA

Markolf Niemz, Medical Faculty Mannheim, University of Heidelberg, Mannheim, Germany

V. Adrian Parsegian, Physical Science Laboratory National Institutes of Health, Bethesda, MD, USA

Linda S. Powers, University of Arizona Tucson, AZ, USA

Earl W. Prohofsky, Department of Physics Purdue University, West Lafayette, IN, USA

Andrew Rubin, Department of Biophysics Moscow State University, Moscow, Russia

Michael Seibert, National Renewable Energy Laboratory, Golden, CO, USA

David Thomas, Department of Biochemistry, University of Minnesota Medical School, Minneapolis, MN, USA

Thomas Lindblad · Jason M. Kinser

Image Processing Using Pulse-Coupled Neural Networks

Applications in Python

Third Edition

 Springer

Thomas Lindblad
Department of Physics
Royal Institute of Technology (KTH)
Stockholm
Sweden

Jason M. Kinser
School of Physics and Computational
Sciences
George Mason University
Fairfax, VI
USA

ISSN 1618-7210

ISBN 978-3-642-36876-9

ISBN 978-3-642-36877-6 (eBook)

DOI 10.1007/978-3-642-36877-6

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013935478

© Springer-Verlag Berlin Heidelberg 1998, 2005, 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

*Dedicated to
John L. Johnson
and
H. John Caulfield (1936–2012)*

Preface to the Third Edition

This third edition has included two major components over the second edition. The first is that a selection of new applications has been addressed. There has been a recent surge in publications using the PCNN or ICM and a few of these have been included.

The second major change has been the inclusion of Python scripts. Over the past decade Python has emerged as a very powerful tool and its use is seen in many applications in the sciences. With the inclusion of a numeric library, Python has the ability to easily handle linear algebra operations with relatively few lines of code. With such efficiency it becomes possible to embed Python scripts in the text along with the theory and applications.

Every attempt has been made to ensure that the Python scripts are complete for applications that are demonstrated here. The scripts were written with Python 2.7 since it is still the standard in LINUX distributions. Users of Python 3.x will find minor differences which are customary when translating from 2.7.

Some readers who are experienced Python programmers will notice that the codes included here could be compressed to even fewer lines. However, the intent of including code is more educational in nature and so the scripts are designed to be readable before being highly compressed.

A website hosted at <http://www.binf.gmu.edu/kinser/> will maintains a ZIP file with all of the Python scripts written by the authors. The Python system, the numeric Python (NumPy), scientific Python packages (SciPy), and the Python Image Library (PIL) can be obtained from their home sites as explained in [Chap. 3](#). All of the scripts provided by the author are copyrighted and can be used only for academic purposes. Commercial applications without expressed written permission of the script's author is prohibited.

Stockholm and Manassas, 2012

Thomas Lindblad
Jason M. Kinser

Preface to the Second Edition

It was stated in the preface of the first edition of this book that image processing by electronic means has been a very active field for decades. This is certainly still true and the goal has been, and still is, to have a machine perform the same functions which humans do quite easily. In reaching this goal we have learned much about human mechanisms and how to apply this knowledge to image processing problems. Although there is still a long way to go, we have learned a lot during the last five or six years. This information and some ideas based upon it has been added to the second edition of this book.

The present edition includes the theory and application of two cortical models: the PCNN (the pulse coupled neural network) and the ICM (intersecting cortical model). These models are based upon biological models of the visual cortex and it is prudent to review the algorithms that strongly influenced the development of the PCNN and the ICM. The outline of the book is otherwise very much the same as in the first edition, although several new applications have been added.

In [Chap. 7](#) a few of these applications will be reviewed including original ideas by co-workers and colleagues. Special thanks are due to Soonil D. D. V. Rughooputh, the dean of the Faculty of Science at the University of Mauritius and Harry C. S. Rughooputh, the dean of the Faculty of Engineering at the University of Mauritius.

We should also like to acknowledge that Guisong Wang, a doctoral candidate in the School of Computational Sciences at GMU, made a significant contribution to [Chap. 5](#).

We would also like to acknowledge the work of several diploma and Ph.D. students at KTH, in particular Jenny Atmer, Nils Zetterlund, and Ulf Ekblad.

Stockholm and Manassas, April 2005

Thomas Lindblad
Jason M. Kinser

Preface to the First Edition

Image processing by electronic means has been a very active field for decades. The goal has been, and still is, to have a machine perform the same image functions which humans do quite easily. This goal is still far from being reached. So we must learn more about the human mechanisms and how to apply this knowledge to image processing problems. Traditionally, the activities in the brain are assumed to take place through the aggregate action of billions of simple processing elements referred to as neurons and connected by complex systems of synapses. Within the concepts of artificial neural networks, the neurons are generally simple devices performing summing, thresholding, etc. However, we show now that the biological neurons are fairly complex and perform much more sophisticated calculations than their artificial counterparts. The neurons are also very specialized and it is thought that there are several hundred types in the brain and messages travel from one neuron to another as pulses.

Recently, scientists have begun to understand the visual cortex of small mammals. This understanding has led to the creation of new algorithms that are achieving new levels of sophistication in electronic image processing. With the advent of such biologically inspired approaches, in particular with respect to neural networks, we have taken another step towards the aforementioned goal.

In our presentation of the visual cortical models we will use the term Pulse-Coupled Neural Network (PCNN). The PCNN is a neural network algorithm that produces a series of binary pulse images when stimulated with a gray scale or color image. This network is different from what we generally mean by artificial neural networks in the sense that it does not train. The goal for image processing is to eventually reach a decision on the content of that image. These decisions are generally far easier to accomplish by examining the pulse outputs of the PCNN rather than the original image. Thus, the PCNN becomes a very useful pre-processing tool. There exists, however, an argument that the PCNN is more than a pre-processor. It is possible that the PCNN also has self-organizing abilities which make it possible to use the PCNN as an associative memory. This is unusual for an algorithm that does not train.

Finally, it should be noted that the PCNN is quite feasible to implement in specialized hardware. Traditional neural networks have had a large fan-in and fan-out. In other words, each neuron was connected to several other neurons. In electronics a

different “wire” is needed to make each connection and large networks are quite difficult to build. The PCNN, on the other hand, has only local connections and in most cases these are always positive. This is quite plausible for electronic implementation.

The PCNN is quite powerful and we are just beginning to explore the possibilities. This text will review the theory and then explore its known image processing applications: segmentation, edge extraction, texture extraction, object identification, object isolation, motion processing, foveation, noise suppression, and image fusion. This text will also introduce arguments as to its ability to process logical arguments and its use as a synergetic computer. Hardware realization of the PCNN will also be presented.

This text is intended for the individual who is familiar with image processing terms and has a basic understanding of previous image processing techniques. It does not require the reader to have an extensive background in these areas. Furthermore, the PCNN is not extremely complicated mathematically so it does not require extensive mathematical skills. However, this text will use Fourier image processing techniques and a working understanding of this field will be helpful in some areas.

The PCNN is fundamentally unique from many of the standard techniques being used today. Many of these fields have the same basic mathematical foundation and the PCNN deviates from this path. It is an exciting field that shows tremendous promise.

Stockholm and Manassas, 1997

Thomas Lindblad
Jason M. Kinser

Acknowledgments

The work reported in this book includes research carried out by the authors together with co-workers at various universities and research establishments. Several research councils, foundations, and agencies have supported the work and made the collaboration possible. Their support is gratefully acknowledged. In particular, we would like to acknowledge the fruitful collaboration and discussions with the following scientists: Kenneth Agehed, Randy Broussard, Åge J. Eide, John Caulfield, Bruce Denby, W. Friday, John L. Johnson, Clark S. Lindsey, Steven Rogers, Thaddeus Roppel, Manuel Samuelides, Åke Steen, Géza Székely, Mary Lou Padgett, and Ilya Rybak. The authors would also like to extend their gratitude to Stefan Rydström for his invaluable editing.

Contents

| | | |
|----------|--|----|
| 1 | Biological Models | 1 |
| 1.1 | Introduction | 3 |
| 1.2 | Biological Foundation | 5 |
| 1.3 | Hodgkin-Huxley | 6 |
| 1.4 | Fitzhugh-Nagumo | 7 |
| 1.5 | Eckhorn Model | 9 |
| 1.6 | Rybak Model | 10 |
| 1.7 | Parodi Model | 10 |
| 1.8 | Summary | 11 |
| | | |
| 2 | Programming in Python | 13 |
| 2.1 | Environment | 13 |
| 2.1.1 | Command Interface | 14 |
| 2.1.2 | IDLE | 14 |
| 2.1.3 | Establishing a Working Environment | 14 |
| 2.2 | Data Types and Simple Math | 15 |
| 2.3 | Tuples, Lists, and Dictionaries | 16 |
| 2.3.1 | Tuples | 16 |
| 2.3.2 | Lists | 17 |
| 2.3.3 | Dictionaries | 18 |
| 2.4 | Slicing | 19 |
| 2.5 | Strings | 20 |
| 2.5.1 | String Functions | 21 |
| 2.5.2 | Type Casting | 23 |
| 2.6 | Control | 23 |
| 2.7 | Input and Output | 25 |
| 2.7.1 | Basic Files | 25 |
| 2.7.2 | Pickle | 26 |
| 2.8 | Functions | 27 |
| 2.9 | Modules | 28 |
| 2.10 | Object Oriented Programming | 30 |
| 2.10.1 | Content of a Class | 30 |
| 2.10.2 | Operator Definitions | 30 |

- 2.10.3 Inheritance 31
- 2.11 Error Checking 32
- 2.12 Summary 33
- 3 NumPy, SciPy and Python Image Library 35**
 - 3.1 NumPy 35
 - 3.1.1 Creating Arrays 35
 - 3.1.2 Converting Arrays 38
 - 3.1.3 Matrix: Vector Multiplications 38
 - 3.1.4 Justification for Arrays 39
 - 3.1.5 Data Types 41
 - 3.1.6 Sorting 43
 - 3.1.7 Conversions to Strings and Lists 45
 - 3.1.8 Changing the Matrix 47
 - 3.1.9 Advanced Slicing 47
 - 3.2 SciPy 49
 - 3.3 Designing in Numpy 52
 - 3.4 Python Image Library 54
 - 3.4.1 Reading an Image 54
 - 3.4.2 Writing an Image 55
 - 3.4.3 Transforming an Image 56
 - 3.5 Summary 56
- 4 The PCNN and ICM 57**
 - 4.1 The PCNN 57
 - 4.1.1 Original Model 57
 - 4.1.2 Implementing in Python 59
 - 4.1.3 Spiking Behaviour 61
 - 4.1.4 Collective Behaviour 64
 - 4.1.5 Time Signatures 66
 - 4.1.6 Neural Connections 67
 - 4.1.7 Fast Linking 70
 - 4.1.8 Models in Analogue Time 73
 - 4.2 The ICM 74
 - 4.2.1 Minimum Requirements 75
 - 4.2.2 ICM Theory 76
 - 4.2.3 Connections in the ICM 77
 - 4.2.4 Python Implementation 83
 - 4.3 Summary 84
- 5 Image Analysis 87**
 - 5.1 Pertinent Image Information 87
 - 5.2 Image Segmentation 92

| | | |
|----------|---|------------|
| 5.2.1 | Blood Cells | 92 |
| 5.2.2 | Mammography | 92 |
| 5.3 | Adaptive Segmentation | 95 |
| 5.4 | Focus and Foveation | 96 |
| 5.4.1 | The Foveation Algorithm. | 97 |
| 5.4.2 | Target Recognition by a PCNN-Based Foveation Model | 99 |
| 5.5 | Image Factorisation. | 104 |
| 5.6 | Summary | 105 |
| 6 | Feedback and Isolation | 107 |
| 6.1 | A Feedback PCNN | 107 |
| 6.2 | Object Isolation | 109 |
| 6.2.1 | Input Normalisation | 111 |
| 6.2.2 | Creating the Filter | 111 |
| 6.2.3 | Edge Enhancement of Pulse Images | 113 |
| 6.2.4 | Correlation and Modifications | 114 |
| 6.2.5 | Peak Detection | 116 |
| 6.2.6 | Modifications to the Input and PCNN. | 116 |
| 6.2.7 | Drivers | 118 |
| 6.3 | Dynamic Object Isolation | 119 |
| 6.4 | Shadowed Objects | 119 |
| 6.5 | Consideration of Noisy Images. | 122 |
| 6.6 | Summary | 125 |
| 7 | Recognition and Classification. | 127 |
| 7.1 | Aircraft | 127 |
| 7.2 | Aurora Borealis | 128 |
| 7.3 | Target Identification: Binary Correlations | 129 |
| 7.4 | Galaxies | 133 |
| 7.5 | Hand Gestures | 137 |
| 7.6 | Road Surface Inspection | 139 |
| 7.7 | Numerals. | 143 |
| 7.7.1 | Data Set | 143 |
| 7.7.2 | Isolating a Class for Training. | 144 |
| 7.8 | Generating Pulse Images | 145 |
| 7.8.1 | Analysis of the Signatures | 146 |
| 7.9 | Face Location and Identification. | 148 |
| 7.10 | Summary | 153 |
| 8 | Texture Recognition | 155 |
| 8.1 | Pulse Spectra | 155 |
| 8.2 | Statistical Separation of the Spectra | 159 |

- 8.3 Recognition Using Statistical Methods 160
- 8.4 Recognition of the Pulse Spectra via an Associative
Memory 161
- 8.5 Biological Application 162
- 8.6 Texture Study 167
- 8.7 Summary 170

- 9 Colour and Multiple Channels 171**
 - 9.1 The Model. 171
 - 9.1.1 Colour Example 172
 - 9.1.2 Python Implementation 176
 - 9.2 Multi-Spectral Example. 180
 - 9.3 Application of Colour Models 183
 - 9.4 Summary 185

- 10 Image Signatures 187**
 - 10.1 Image Signature Theory 187
 - 10.1.1 The PCNN and Image Signatures 188
 - 10.1.2 Colour Versus Shape. 189
 - 10.2 The Signature of Objects. 189
 - 10.3 The Signatures of Real Images. 191
 - 10.4 Image Signature Database 192
 - 10.5 Computing the Optimal Viewing Angle 193
 - 10.6 Motion Estimation 196
 - 10.7 Summary 198

- 11 Logic 201**
 - 11.1 Maze Running and TSP 201
 - 11.2 Barcodes and Navigation. 203
 - 11.3 Summary 208

- Appendix A: Image Converters 209**
- Appendix B: The Geometry Module. 215**
- Appendix C: The Fractional Power Filter 217**
- Appendix D: Correlation 219**
- Appendix E: The FAAM 223**
- Appendix F: Principal Component Analysis 227**

| | |
|-----------------------------|------------|
| Contents | xix |
| References | 229 |
| Index | 235 |

Python Codes

| | | |
|------|--|----|
| 2.1 | Python performing simple calculations. | 14 |
| 2.2 | Setting up the Python environment for IDLE users. Directory names will be unique for each user. | 15 |
| 2.3 | Division in Python. | 16 |
| 2.4 | Conversion of integers to floats. | 16 |
| 2.5 | A simple tuple demonstration | 17 |
| 2.6 | A simple list demonstration | 17 |
| 2.7 | The use of remove and pop | 18 |
| 2.8 | A simple dictionary example | 19 |
| 2.9 | The key commands | 19 |
| 2.10 | Simple indices | 19 |
| 2.11 | Simple slicing | 20 |
| 2.12 | Slicing in steps | 20 |
| 2.13 | Creating simple strings. | 21 |
| 2.14 | Accessing characters in a string | 21 |
| 2.15 | Finding characters in a string | 22 |
| 2.16 | Converting characters to upper case and replacing characters | 22 |
| 2.17 | Splitting and joining strings | 22 |
| 2.18 | Converting strings to other data types | 23 |
| 2.19 | A simple <code>if</code> statement. | 24 |
| 2.20 | A simple <code>if</code> statement with multiple commands. | 24 |
| 2.21 | A compound <code>if</code> statement | 24 |
| 2.22 | A <code>while</code> statement. | 25 |
| 2.23 | A <code>for</code> loop | 25 |
| 2.24 | A traditional <code>for</code> loop. | 25 |
| 2.25 | Writing and reading a text file | 26 |
| 2.26 | Pickling | 26 |
| 2.27 | A simple function | 27 |
| 2.28 | A function returning data | 27 |
| 2.29 | Default arguments | 28 |
| 2.30 | Creating a module | 29 |
| 2.31 | From: import | 29 |

- 2.32 Using `execfile`. 29
- 2.33 A simple object. 31
- 2.34 Operator definition 31
- 2.35 Inheritance 32
- 2.36 Trapping an error 33
- 3.1 Creation of vectors 36
- 3.2 Math operations for vectors 36
- 3.3 Math operations for two vectors 36
- 3.4 Creating matrices 37
- 3.5 Creating tensors 37
- 3.6 Accessing data in a matrix 37
- 3.7 Converting between vectors and matrices. 38
- 3.8 Vector-matrix and matrix-vector multiplications 39
- 3.9 Multiplying columns of a matrix with elements in a vector 39
- 3.10 Comparing the computational costs of interpreted commands. 40
- 3.11 Retrieving the type of data within an array. 41
- 3.12 Using the `max` function 42
- 3.13 Using the similar functions. 42
- 3.14 Using the similar functions. 43
- 3.15 Using the similar functions. 43
- 3.16 Using the `nonzero` function. 44
- 3.17 Mathematical functions for an array 44
- 3.18 Sorting arrays 45
- 3.19 Conversions to and from a string 46
- 3.20 Swapping bytes in an array 47
- 3.21 Examples of the `transpose` function 48
- 3.22 Examples of the `resize` function 49
- 3.23 Advanced slicing for arrays 49
- 3.24 Advanced slicing for arrays with multiple dimensions 50
- 3.25 Matrix inverse using the SciPy function. 50
- 3.26 Isolating two contiguous regions 51
- 3.27 Execution time for a double loop 52
- 3.28 Execution time for a single command 53
- 3.29 Inserting a safety print statement. 54
- 3.30 Loading an image 54
- 3.31 Writing an image 55
- 3.32 Converting an image 56
- 3.33 Other transformations 56
- 4.1 Part 1 of `pcnn.py`. 59
- 4.2 Part 2 of `pcnn.py`. 60
- 4.3 Creating an image of a ‘T’ 60
- 4.4 Driver for the PCNN 61
- 4.5 Collecting the internal neural activities 65
- 4.6 Fast linking iteration for `pcnn.py`. 72

| | | |
|------|--|-----|
| 4.7 | Executing a fast linking | 73 |
| 4.8 | Constructor for ICM | 83 |
| 4.9 | Iteration for ICM. | 84 |
| 4.10 | Iteration for ICM creating centripetal autowaves. | 84 |
| 4.11 | Driving the ICM | 84 |
| 4.12 | The LevelSet function | 85 |
| 5.1 | Iterations for the ICM | 94 |
| 5.2 | The Corners function | 99 |
| 5.3 | The peak detecting function | 100 |
| 5.4 | The Mark and Mix functions | 100 |
| 5.5 | Loading the image and finding the peaks | 101 |
| 6.1 | The LoadImage function | 111 |
| 6.2 | The LoadTarget function | 112 |
| 6.3 | The EdgeEncourage function. | 113 |
| 6.4 | The NormFilter function | 113 |
| 6.5 | The EdgeEnhance function | 114 |
| 6.6 | The PCECorrelate function. | 115 |
| 6.7 | The Peaks function. | 116 |
| 6.8 | The Enhance function | 117 |
| 6.9 | The SingleIteration function | 118 |
| 6.10 | The Driver function | 119 |
| 7.1 | The UnpackImages function | 144 |
| 7.2 | The UnpackLabels function. | 144 |
| 7.3 | The IsolateClass function | 145 |
| 7.4 | The PulseOnNumeral function. | 145 |
| 7.5 | The RunAll function | 146 |
| 7.6 | Isolating candidate skin pixels | 151 |
| 7.7 | Running the modified PCNN | 152 |
| 7.8 | The FastLYiterate function. | 152 |
| 7.9 | Horizontal sums across a candidate shape | 152 |
| 8.1 | The FileNames and LoadImage functions. | 168 |
| 8.2 | The Cutup function. | 168 |
| 8.3 | The ManySignatures function | 168 |
| 8.4 | The Driver function | 169 |
| 9.1 | The constructor for <i>ucm3D</i> | 176 |
| 9.2 | The Image2Stim function | 176 |
| 9.3 | The Iterate function | 177 |
| 9.4 | The Y2Image function. | 178 |
| 9.5 | Example implementation | 179 |
| 9.6 | Converting an image to the YUV format | 184 |
| 9.7 | Running 3 ICMs | 184 |
| 9.8 | Saving the pulse images as colour images | 184 |
| 11.1 | The Mazeliterate function | 202 |
| 11.2 | The RunMaze function | 203 |

- A.1 Functions for converting between images and arrays 209
- A.2 The **a2i** function 210
- A.3 The **i2a** function 210
- A.4 The **RGB2cube** function 211
- A.5 The **Cube2Image** function 211
- A.6 Functions for color conversions 212
- A.7 Functions from the *convert.py* module 212
- B.1 The **Circle** function 215
- B.2 The **Plop** function 216
- C.1 The **FPF** function 218
- C.2 Computing the FPF for multiple matrices 218
- D.1 The **Swap** function 220
- D.2 The **Correlate** function 221
- D.3 The **PCE** function 221
- E.1 Running the FAAM 224
- F.1 The **PCA** function 228

Chapter 1

Biological Models

Humans have an outstanding ability to recognise, classify and discriminate objects with extreme ease. For example, if a person was in a large classroom and was asked to find the light switch it would not take more than a second or two. Even if the light switch was located in a different place than the person expected or it was shaped differently than expected it would not be difficult to find the switch. Humans also do not need to see hundreds of exemplars in order to identify similar objects. A person needs to see only a few dogs and then he is able to recognise dogs even from species that he has not seen before. This recognition ability also holds true for animals, to a greater or lesser extent. A spider has no problem recognising a fly as even a baby spider can do that. At this level we are talking about a few hundred to a thousand processing elements or neurons. Nevertheless the biological systems seem to do their job very well.

Computers, on the other hand, have a very difficult time with these tasks. Von Neumann machines need a large amount of memory and significant speed to even come close to the processing time of a human. Furthermore, the software for such simple general tasks does not exist. There are special problems where the machine can perform specific functions well, but the machines do not perform general image processing and recognition tasks to the extent that animals and humans do.

Implementations of neural systems in silicon hardware have been tried by companies Intel and IBM. The Electrically Trainable Neural Network (ETANN) chip [43] from Intel had 128 neurons¹ and the first Zero Instruction Set Computer (ZISC36) chip[1] from IBM had 36 neurons. However, these are all “mathematical neurons” based on the back-propagation algorithm [14] and the radial basis function algorithms [73] and really not any implementation of biological systems. The ZISC36 chip could easily be put in parallel [66] to make use of several hundreds of neurons. It has also been further developed and is today available as CIMK with a thousand neurons [21]. This chip is between a fly and a worm with respect to the number of interconnections, although a bit “faster” than both. However, there is still a long way to go before reaching small mammals and humans as illustrated in

¹ There are several examples of how 128 neurons can be used with one example show in [65].

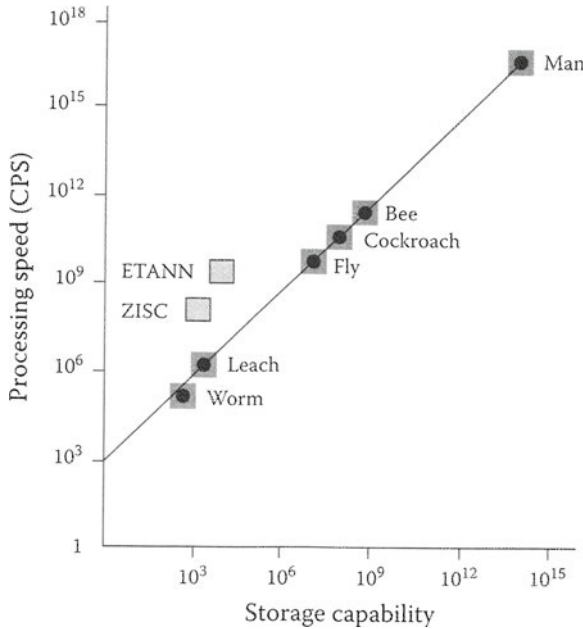


Fig. 1.1 Comparing “brains” of some animals in some neural networks systems as discussed in the text

Fig. 1.1. Recent work at Manchester University is to develop SpiNNaker (Spiking Neural Network Architecture) which is a parallel computer specifically designed to model large scale spiking neural networks. The design will allow each computer to contain more than one million cores. Completion of the first machine is expected by the end of 2013. The ETANN, ZISC and a few other neural network chips are shown in Fig. 1.2.

It is often claimed that neuromorphic machines outperform von Neumann machines at certain environmental complexity (e.g., input combinatorics). There is a “break even point” and after this point the machine complexity (e.g., size, power, memory, gates, synapses) increases steeply for von Neumann computers but not so much for the neural architectures. However, there are still many orders of magnitude of complexity before one reaches the “human” level of performance. Besides the above problem of the number of neurons, there are at least two other fundamental items to consider: the neurons designed for specific tasks and the intelligent mammal sensors. This is particularly true for the visual system. The neurons get auxiliary information from adjacent neurons, the information is sent in separate paths in the mid-brain, backward signals are used to prioritise important information. Using computer language one would say that the feature extraction system, its redundancy and the parallel triggering system in the mid-brain ensure that important information reaches the visual cortex. At the same time, there is a tremendous reduction in data volume from perhaps initially 10^8 neurons and a bit-rate of 50 Mbits/s to approximate video speed when we become aware of what we are seeing.

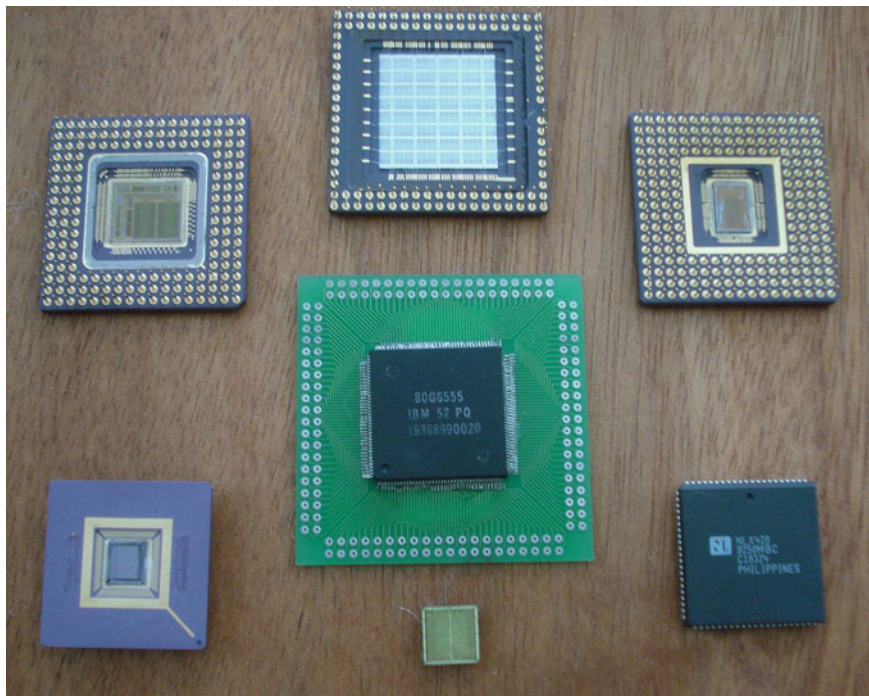
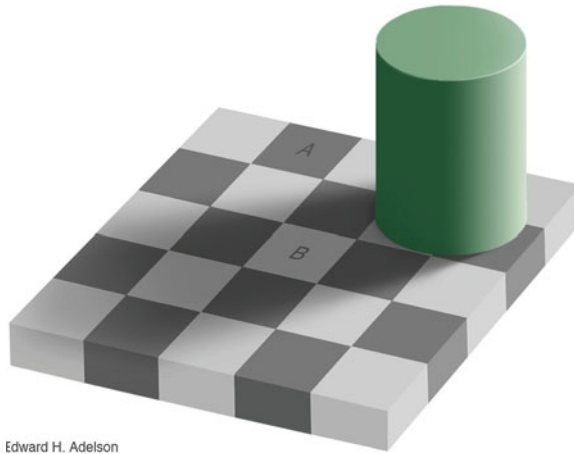


Fig. 1.2 A few neural network chips. The ETANN (*top right*) and ZISC036 (*middle and lower middle*) are mentioned in the text and plotted in Fig. 1.1

1.1 Introduction

One of the processes occurs in the visual cortex, which is the part of the brain that receives information from the eye. At this point in the system the eye has already processed and significantly changed the image. The visual cortex converts the resultant eye image into a stream of pulses. Synthetic models of this portion of the brain for small mammals have been developed and successfully applied to many image processing applications.

The mammalian visual system is considerably more elaborate than simply processing an input image with a set of inner products. Many operations are performed before decisions are reached as to the content of the image. Neuro-science does not yet understand all of processes. However, sometimes the visual system is fooled, in particular where we expect colour and shades to follow some rules and patterns. There are very many examples of this, (e.g. the shadow of a cylinder on a board of chess shown in Fig. 1.3). Most people would say that the grey scale of square “A” is darker than that of square “B”, but even the simplest Paint program of a von Neumann computer would say that they are exactly the same.



Edward H. Adelson

Fig. 1.3 The shadow of a cylinder on a checkerboard. Is square **a** really darker than square **b**?

Another example is the case of hiding a person from a searching adversary. Hiding in an open field may offer advantages over hiding in a ditch in that such a place is unexpected and away from visual edges which are natural attractors in human vision (see Sect. 5.4). “What you see is not always the truth.” An excellent example is the awareness test [4] in which the viewer is asked to count the number of passes of a ball between a set of players wearing a particular jersey. In this video a dancer dressed as a bear moves across the frame of view and most viewers completely miss his presence. 80% or more of our (university) students did not see the bear. Human processing of information is clearly not based solely on the visual input but also highly affected by other processes in the brain.

This chapter will mention a few of the important operations to provide a glimpse of the complexity of the processes. It soon becomes clear that the mammalian system is far more complicated than the usual computer algorithms used in image recognition. It is almost silly to assume that such simple operations can match the performance of the biological system. Of course, image input is performed through the eyes. Receptors within the retina at the back of the eye are not evenly distributed nor are they all sensitive to the same optical information. Some receptors are more sensitive to motion, colour, or intensity. Furthermore, the receptors are interconnected. When one receptor receives optical information it alters the behaviour of other surrounding receptors. A mathematical operation is thus performed on the image before it even leaves the eye. The eye also receives feedback information. We humans do not stare at images, we foveate. Our centre of attention moves about portions of the image as we gather clues as to the content. Furthermore, feedback information also alters the output of the receptors.

After the image information leaves the eye it is received by the visual cortex. Here the information is further analysed by the brain. The investigation of the visual cortex of the cat [26] and the guinea pig [93] have been the foundation of the digital models

used in this text. Although these models are a big step in emulating the mammalian visual system, they are still very simplified models of a very complicated system. Intensive research continues to understand fully the processing. However, much can still be implemented or applied already today.

1.2 Biological Foundation

While there are discussions as to the actual cortex mechanisms, the products of these discussions are quite useful and applicable to many fields. In other words, the algorithms being presented as cortical models are quite useful regardless of their accuracy in modelling the cortex. Following this brief introduction to the primate cortical system, the rest of this book will be concerned with applying cortical models and not with the actual mechanisms of the visual cortex.

In spite of its enormous complexity, two basic hierarchical pathways can model the visual cortex system: the pavoellular one and the mangnocellular one, processing (mainly) colour information and form/motion, respectively. Figure 1.4 shows a model of these two pathways. The retina has luminance and colour detectors which interpret images and pre-process them before conveying the information to the visual cortex. The lateral geniculate nucleus, LGN, separates the image into components that include luminance, contrast, frequency, etc. before information is sent to the visual cortex (labelled V in Fig. 1.4).

The cortical visual areas are labelled V1–V5 in Fig. 1.4. V1 represents the striate visual cortex and is believed to contain the most detailed and least processed image.

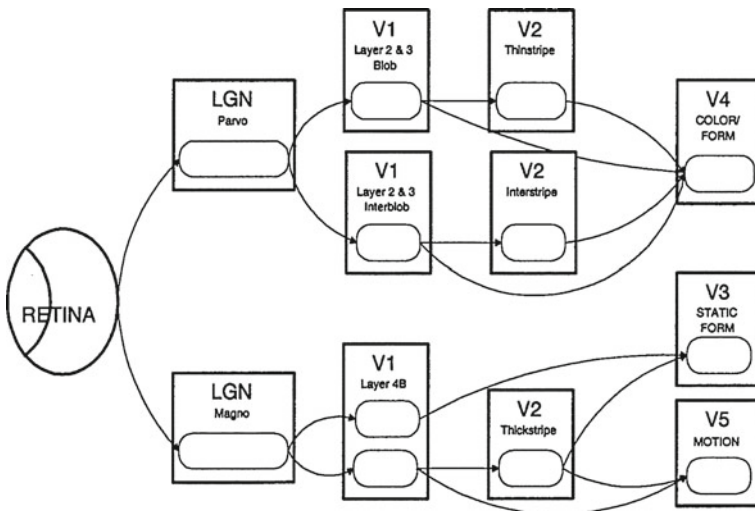


Fig. 1.4 A model of the visual system. The abbreviations are explained in the text. Only feedforward signals are shown