COMPUTATIONAL IMAGING AND VISION

Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, written in Mathematica

Bart M. ter Haar Romeny

Kluwer Academic Publishers

Front-End Vision and Multi-Scale Image Analysis

Computational Imaging and Vision

Volume 27

Front-End Vision and Multi-Scale Image Analysis

Multi-Scale Computer Vision Theory and Applications, written in Mathematica

by

Bart M. ter Haar Romeny

Eindhoven University of Technology,
Department of Biomedical Engineering,
Biomedical Imaging and Informatics,
Eindhoven, The Netherlands

Springer

Prof. Dr. Bart M. ter Haar Romeny
Eindhoven University of Technology
Fac. Biomedical Engineering
Dept. Image Analysis & Interpretation
5600 MB Eindhoven
Netherlands

This book is available in Mathematica, a mathematical programming language, at the following website:
http://bmia.bmt.tue.nl/Education/Courses/FEV/book/index.html

ISBN 978-1-4020-1503-8 (HB) e-ISBN 978-1-4020-8840-7 (e-book)
ISBN 978-1-4020-1507-6 (PB)

Library of Congress Control Number: 2008929603

Reprinted in 2008

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Table of contents

Front-End Vision and Multi-Scale Image Analysis xiii
The purpose of this book xiii
Scale-space theory is biologically motivated computer vision xiv
This book has been written in *Mathematica*. xvi
Acknowledgements xviii

1. Apertures and the notion of scale 1
1.1 Observations and the size of apertures 1
1.2 Mathematics, physics, and vision 2
1.3 We blur by looking 5
1.4 A critical view on observations 9
1.5 Summary of this chapter 12

2. Foundations of scale-space 13
2.1 Constraints for an uncommitted front-end 13
2.2 Axioms of a visual front-end 15
2.2.1 Dimensional analysis 15
2.2.2 The cooking of a turkey 16
2.2.3 Reynold's number 18
2.2.4 Rowing: more oarsmen, higher speed? 19
2.3 Axiomatic derivation of the Gaussian kernel 21
2.4 Scale-space from causality 23
2.5 Scale-space from entropy maximization 25
2.6 Derivatives of sampled, observed data 27
2.7 Scale-space stack 31
2.8 Sampling the scale-axis 32
2.9 Summary of this chapter 35

3. The Gaussian kernel 37
3.1 The Gaussian kernel 37
3.2 Normalization 38
3.3 Cascade property, selfsimilarity 39
3.4 The scale parameter 40
3.5 Relation to generalized functions 40
3.6 Separability 43
3.7 Relation to binomial coefficients 43
3.8 The Fourier transform of the Gaussian kernel 44
3.9 Central limit theorem 46
3.10 Anisotropy 48
3.11 The diffusion equation 49
3.12 Summary of this chapter 50

4. Gaussian derivatives ... 53
4.1 Introduction ... 53
4.2 Shape and algebraic structure ... 53
4.3 Gaussian derivatives in the Fourier domain ... 57
4.4 Zero crossings of Gaussian derivative functions ... 59
4.5 The correlation between Gaussian derivatives ... 60
4.6 Discrete Gaussian kernels ... 64
4.7 Other families of kernels ... 65
4.8 Higher dimensions and separability ... 67
4.9 Summary of this chapter ... 69

5. Multi-scale derivatives: implementations ... 71
5.1 Implementation in the spatial domain ... 71
5.2 Separable implementation ... 73
5.3 Some examples ... 74
5.4 N-dim Gaussian derivative operator implementation ... 78
5.5 Implementation in the Fourier domain ... 79
5.6 Boundaries ... 83
5.7 Advanced topic: speed concerns in *Mathematica* ... 85
5.8 Summary of this chapter ... 89

6. Differential structure of images ... 91
6.1 The differential structure of images ... 91
6.2 Isophotes and flowlines ... 92
6.3 Coordinate systems and transformations ... 96
6.4 Directional derivatives ... 102
6.5 First order gauge coordinates ... 103
6.6 Gauge coordinate invariants: examples ... 108
6.6.1 Ridge detection ... 108
6.6.2 Isophote and flowline curvature in gauge coord ... 110
6.6.3 Affine invariant corner detection ... 113
6.7 A curvature illusion ... 115
6.8 Second order structure ... 117
6.8.1 The Hessian matrix and principal curvatures ... 119
6.8.2 The shape index ... 120
6.8.3 Principal directions ... 122
6.8.4 Gaussian and mean curvature ... 123
6.8.5 Minimal and zero Gaussian curvature surfaces ... 126
6.9 Third order image structure: T-junction detection ... 127
6.10 Fourth order image structure: junction detection ... 131
6.11 Scale invariance and natural coordinates ... 132
6.12 Irreducible invariants ... 134
Intermezzo: Tensor notation ... 135
6.13 Summary of this chapter ... 136

7. Natural limits on observations 137
7.1 Limits on differentiation: scale, accuracy and order 137
7.2 Summary of this chapter 141

8. Differentiation and regularization 143
8.1 Regularization 143
8.2 Regular tempered distributions and test functions 144
8.3 An example of regularization 147
8.4 Relation regularization $\Longleftrightarrow$ Gaussian scale-space 148
8.5 Summary of this chapter 152

9. The front-end visual system - the retina 153
9.1 Introduction 153
9.2 Studies of vision 154
9.3 The eye 156
9.4 The retina 157
9.1 Retinal receptive fields 160
9.6 Sensitivity profile measurement of a receptive field 162
9.7 Summary of this chapter 165

10. A scale-space model for the retinal sampling 167
10.1 The size and spatial distribution of receptive fields 167
10.2 A scale-space model for the retinal receptive fields 172
10.3 Summary of this chapter 177

11. The front-end visual system - LGN and cortex 179
11.1 The thalamus 179
11.2 The lateral geniculate nucleus (LGN) 181
11.3 Corticofugal connections to the LGN 183
11.4 The primary visual cortex 185
11.4.1 Simple cells 187
11.4.2 Complex cells 188
11.4.3 Directional selectivity 189
11.5 Intermezzo: Measurement of neural activity in the brain 191
Electro-Encephalography (EEG) 191
Magneto-Encephalography (MEG) 192
Functional MRI (fMRI) 193
Optical imaging with voltage sensitive dyes 194
Positron Emission Tomography (PET) 194
11.6 Summary of this chapter 195

12. The front-end visual system - cortical columns 197
12.1 Hypercolumns and orientation structure 197
12.2 Stabilized retinal images 200
12.3 The concept of local sign 202
12.4 Gaussian derivatives and Eigen-images 204

12.5 Plasticity and self-organization ... 208
12.6 Higher cortical visual areas ... 210
12.7 Summary of this chapter ... 211
12.8 Vision dictionary ... 211
12.8.1 Further reading on the web: ... 212

13 Deep structure I. watershed segmentation ... 215
13.1 Multi-scale measurements ... 215
13.2 Scale selection ... 216
13.3 Normalized feature detection ... 218
13.4 Automatic scale selection ... 219
13.4.1 λ-Normalized scale selection ... 220
13.4.2 Is this really deep structure? ... 220
13.5 Edge focusing ... 221
13.5.1 Simplification followed by focusing ... 221
13.5.2 Linking in 1D ... 222
13.6 Follicle detection in 3D ultrasound ... 225
13.6.1 Fitting spherical harmonics to 3D points ... 229
13.7 Multi-scale segmentation ... 231
13.7.1 Dissimilarity measure in scale-space ... 231
13.7.2 Watershed segmentation ... 232
13.7.3 Linking of regions ... 234
13.7.4 The multi-scale watershed segmentation ... 237
13.8 Deep structure and nonlinear diffusion ... 239
13.8.1 Non-linear diffusion watershed segmentation ... 239

14. Deep structure II. catastrophe theory ... 241
14.1 Catastrophes and singularities ... 241
14.2 Evolution of image singularities in scale-space ... 242
14.3 Catastrophe theory basics ... 243
14.3.1 Functions ... 243
14.3.2 Characterization of points ... 243
14.3.3 Structural equivalence ... 244
14.3.4 Local characterization of functions ... 244
14.3.5 Thom's theorem ... 245
14.3.6 Generic property ... 246
14.3.7 Dimensionality ... 246
14.3.8 Illustration of the concepts ... 247
14.4 Catastrophe theory in scale-space ... 250
14.4.1 Generic events for differential operators ... 251
14.4.2 Generic events for other differential operators ... 254
14.4.3 Annihilations and creations ... 255
14.5 Summary of this chapter ... 256

15. Deep structure III. topological numbers 257
15.1 Topological numbers 257
15.1.1 Topological numbers in scale-space 258
15.1.2 Topological number for a signal 259
15.1.3 Topological number for an image 259
15.1.4 The winding number on 2D images 260
15.2 Topological numbers and catastrophes 263
15.3 The deep structure toolbox 265
15.3.1 Detection of singularities 265
15.3.2 Linking of singularities 265
15.3.3 Linking of contours 268
15.3.4 Detection of catastrophes 268
15.3.5 General discrete geometry approach 269
15.4 From deep structure to global structure 271
15.4.1 Image representations 271
15.4.2 Hierarchical pre-segmentation 272
15.4.3 Perceptual grouping 273
15.4.4 Matching and registration 274
15.4.5 Image databases 274
15.4.6 Image understanding 275
15.5 Summary of this chapter 275

16. Deblurring Gaussian blur 277
16.1 Deblurring 277
16.2 Deblurring with a scale-space approach 277
16.3 Less accurate representation, noise and holes 281
16.4 Summary of this chapter 284

17. Multi-scale optic flow 285
17.1 Introduction 285
17.2 Motion detection with pairs of receptive fields 286
17.3 Image deformation by a discrete vectorfield 289
17.4 The optic flow constraint equation 290
17.5 Scalar and density images 292
17.6 Derivation of multi-scale optic flow constraint equation 292
17.6.1 Scalar images, normal flow. 296
17.6.2 Density images, normal flow. 301
17.7 Testing the optic flow constraint equations 303
17.8 Cleaning up the vector field 305
17.9 Scale selection 307
17.10 Discussion 309
17.11 Summary of this chapter 310

18. Color differential structure 311
18.1 Introduction 311
18.2 Color image formation and color invariants 311
18.3 Koenderink's Gaussian derivative color model 314
18.4 Implementation 320
18.5 Combination with spatial constraints 325
18.6 Summary of this chapter 327

19. Steerable kernels 329
19.1 Introduction 329
19.2 Multi-scale orientation 330
19.3 Orientation analysis with Gaussian derivatives 331
19.4 Steering with self-similar functions 332
19.5 Steering with Cartesian partial derivatives 336
19.6 Detection of stellate tumors 338
19.7 Classical papers and student tasks 342
19.8 Summary of this chapter 343

20. Scale-time 345
20.1 Introduction 345
20.2 Analysis of prerecorded time-sequences 346
20.3 Causal time-scale is logarithmic 349
20.4 Other derivations of logarithmic scale-time 351
20.5 Real-time receptive fields 353
20.6 A scale-space model for time-causal receptive fields 354
20.7 Conclusion 359
20.8 Summary of this chapter 360

21. Geometry-driven diffusion 361
21.1 Adaptive Smoothing and Image Evolution 361
21.2 Nonlinear Diffusion Equations 362
21.3 The Perona & Malik Equation 364
21.4 Scale-space implementation of the P&M equation 366
21.5 The P&M equation is ill-posed. 370
21.6 Von Neumann stability of numerical PDE's 372
21.7 Stability of Gaussian linear diffusion 373
21.8 A practical example of numerical stability 376
21.9 Euclidean shortening flow 378
21.10 Grayscale invariance 379
21.11 Numerical examples shortening flow 379
21.12 Curve Evolution 382
21.13 Duality between PDE- and curve evolution 383
21.14 Mathematical Morphology 386
21.15 Mathematical morphology on grayvalued images 389
21.16 Mathematical morphology versus scale-space 390
21.17 Summary of this chapter 390

22. Epilog 393

A. Introduction to *Mathematica* 395
A.1 Quick overview of using Mathematica 395
A.2 Quick overview of the most useful commands 397
A.3 Pure functions 401
A.4 Pattern matching 401
A.5 Some special plot forms 404
A.6 A faster way to read binary 3D data 405
A.7 What often goes wrong 407
A.8 Suggested reading 410
A9. Web resources 412

B. The concept of convolution 413
B.1 Convolution 413
B.2 Convolution is a product in the Fourier domain 416

C. Installing the book and packages 419
C.1 Content 419
C.2 Installation for all systems 420
C.3 Viewing the book in the Help Browser 420
C.4 Sources of additional applications 421

D. First Start with *Mathematica*: Tips & Tricks 423
1. Evaluation 423
2. Images 423
3. Programming 424
4. 3D 424

References 425

Index 455

Front-End Vision and Multi-Scale Image Analysis

Bart M. ter Haar Romeny, PhD

Department of Biomedical Engineering
Eindhoven University of Technology
The Netherlands

Tell me, and I will forget. Show me, and I will remember.
Involve me, and I will understand.
Old Chinese proverb

The purpose of this book

Scale is not an important parameter in computer vision research. It is an *essential* parameter. It is an immediate consequence of the process of observation, of *measurements*. This book is about scale, and its fundamental notion in computer vision, as well as human vision.

Scale-space theory is the theory of apertures, through which we and machines observe the world. The apertures come in an astounding variety. They can be exploited to model the first stages of human vision, and they appear in all aspects of computer vision, such as the extraction of features, the measurement of optic flow and stereo disparity, to do orientation analysis, segmentation, image enhancement etc. They have an essential role in the fundamental processes of differentiation and regularization.

Scale-space theory is named after the space that is formed by looking at an image at many different scales simultaneously.

When stacked, we get one dimension extra, i.e. the scale dimension. The scale-space is the space of the spatial and scale dimensions, see figure 1.

This book is a tutorial course. The level of the book is undergraduate and first level graduate. Its main purpose is to be used as a coursebook in computer vision and front-end vision entry courses. It may also be useful as an entry point for research in biological vision.

Although there are excellent texts appearing on the notion of scale space, most of them are not easy reading for people just entering this field or lacking a solid mathematical background. This book is intended partly to fill this gap, to act as an entry point for the

growing literature on scale-space theory. Throughout the book we will work steadily through the necessary mathematics.

The book discusses the many classical papers published over the last two decades, when scale-space theory became mature. The different approaches and findings are put into context. First, linear scale-space theory is derived from first principles, giving it a sound mathematical basis.

The notion that a multi-scale approach is a natural consequence of the process of observation is interwoven in every chapter of this book. E.g. Horn and Schunck's famous optic flow equation gets a new meaning when we 'look at the data'. The concept of a point and local point operators like the derivative operator diffuse into versions with a Gaussian extent, making the process of differentiation well posed. It immediately makes large and mature fields like differential geometry, invariant theory, tensor analysis and singularity theory available for analysis on discrete data, such as images.

We develop ready-to-use applications of differential invariants of second, third and fourth order. The relation between accuracy, differential order and scale of the operator is developed, and an example of very high order derivatives is worked out in the analytical deblurring of Gaussian blur.

Practical examples are also developed in the chapters on multi-scale optic flow and multi-scale differential structure of color images. Again, the physics of the observation process forces the analytical solution to be multi-scale. Several examples of ways to come to proper scale-selection are treated underway.

Figure 1. A scale-space of a sagittal MR image. The image is blurred with a Gaussian kernel with variable width σ, which is the third dimension in this image.

Scale-space theory is biologically motivated computer vision

We consider it very important to have many cross-links between multi-scale computer vision theory, and findings in the human (mammalian) visual system. We hope the reader will appreciate the mutual cross-fertilization between these fields. For that reason we elaborate the current state of the art in neurophysiological and psychophysical findings of the first stages of the visual system.

The chapters on time-scale and multi-scale orientation analysis are directly inspired by findings from biological vision. The grouping of local properties into meaningful larger subgroups (perceptual grouping) is treated both on the level of establishing neighborhood relationships through all measured properties of the points, and through

the study of the deep structure of images, where topology comes in as a mathematical toolkit. The natural hierarchical ordering is exploited in a practical application, where we discuss multi-scale watershed segmentation.

This book is meant to be a practical and interactive book. It is written as a series of notebooks in *Mathematica* 4, a modern computer algebra language/system.

For every technique discussed the complete code is presented. The reader can run and adapt all experiments himself, and learn by example and prototyping. The most effective way to master the content is to go through the notebooks on a computer running *Mathematica* and play with variations.

This book is a tribute to Jan Koenderink, professor at Utrecht University in the Netherlands, chairman of the Physics Department 'Physics of Man'. He can be considered the 'godfather' of modern scale-space theory. A brilliant physicist, combining broad knowledge on human visual perception with deep insight in the mathematics and physics of the problems.

Figure 2. Prof. dr. hon.dr. Jan Koenderink.

This book is just a humble introduction to his monumental oeuvre and the offspin of it. Many papers he wrote together with his wife, Ans van Doorn. They published on virtually every aspect of front-end vision *and* computer vision with a strong perceptually based inspiration, and the physical modeling of it.

This book is written for both the computer vision scientist with an interest in multi-scale approaches to image processing, and for the neuroscientist with an appeal for mathematical modeling of the early stages of the visual system. One of the purposes of this book is to bridge the gap between both worlds. To accommodate a wide readership, both from physics and biology, sometimes mathematical rigor is lacking (but can be found in the indicated references) in favor of clarity of the exposition.

Figure 3. Attendants of the first international Scale-Space conference, Summer 1997 in Utrecht, the Netherlands, chaired by the author (standing fourth from right).

Figure 4. Attendants of the second international Scale-Space conference, Summer 1999 in Corfu, Greece, chaired by Mads Nielsen, PhD (IT-Univ. Copenhagen, foreground fifth from right). See for the conference series: www.scalespace.org.

This book has been written in *Mathematica*

This book is written as a series of *Mathematica* notebooks. *Mathematica* is a high level interactive mathematical programming language, developed and marketed by Stephen Wolfram (www.wolfram.com). Notebooks are interactive scientific documents, containing both the text as the code.

Mathematica consists of a two separate programs, the kernel (the computing engine) and a front-end which handles all information for and from the user. The structure of 'cells' in the front-end enables the efficient mix of explaining text, computer code and graphics in an intuitive way. The reasons to write this book in *Mathematica* are plentiful:

- We can now do *mathematical* prototyping with computer vision principles/techniques on images. The integration of both symbolic and fast numerical capabilities, and the powerful pattern matching techniques make up for a new and efficient approach to apply and teach computer vision, more suitable for human mathematical reasoning. For, computer vision *is* mathematics on images. It is now easy to do *rapid prototyping*.

- Reading a scientific method now has something extra: the code of every method discussed is available, ready for testing, with modifications and applications to the reader's images. The

gap between the appreciation of a method in a theoretical paper and one's own working software that applies the method is now closing. David Donoho writes about his WaveLab package: "An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures. (http://www-stat.stanford.edu/~wavelab/).

- *Mathematica* stays close to the traditional notation. The mathematical notation of e.g. symbols, operators and rules are virtually identical to traditional math

The notebooks are WYSIWYG. Notebooks can easily be saved as LaTeX or HTML/MathML documents. Notebooks are portable ASCII documents, they appear virtually identical on a wide range of computer platforms.

- All programs become compact. In this book no example exceeds 20 lines of code. There are no **`for`**, **`while`** or **`do`** loops. Most commands are **`Listable`**, i.e. operate on any member of the operand list. The language contains hardly abbreviations, and is so intuitive that learning the language may be mastered during reading the book. In the appendix a list of tutorial books on *Mathematica* is given, and a summary of the command structure of the most popular commands used in this book.

- Wolfram Research Inc. indicates that over 2 million licenses are sold. It may serve as a (WWW-based) starting set in exchangeable *Mathematica* computer vision routines.

- *Mathematica* is complete. Over 2500 highly optimized mathematical routines are on board, which relieves the computer vision programmer from searching for routines in Numerical Recipes, IMSL etc. It has graphical capabilities for 1D to 4D (animations). It is now integrated with Java (**`JLink`**), which is available anywhere and ideally suited for further development of the GUI and real-time manipulation with the data. *Mathematica* code can be compiled for further speed increase.

The parallel version of *Mathematica* now enables the easy distribution over a range of kernels on different computers on a network.

- Last but not least: the present version of *Mathematica* is fast. From release 4 it has reached a point where, from being an ideal rapid prototyping tool, it is now turning into an all-round prototyping *and* application tool. The run-time of most experiments described in this book is within fractions of seconds to tens of seconds on a typical 1.7 GHz 256 MB Pentium IV system under Windows.

It is platform independent, and is available on any type of computer.

Acknowledgements

The contents of this book is based on work of: Jan J. Koenderink, Luc M. J. Florack, Tony Lindeberg, Wiro J. Niessen, Alfons H. Salden, Mads Nielsen, Jon Sporring, Ole F. Olsen, Jan-Mark Geuzebroek, Erik Dam, Peter Johansen, Avan Suinesiaputra, Hans J. Blom, Bram van Ginneken, Stiliyan Kalitzin, Joes Staal, Robert Maas, Max Viergever, Antonio López Peña, Nico Karssemeijer, Stephen Pizer, Dave Eberley, Jean-Philippe Thirion, Stephen Zucker, Wim van de Grind, Ans Koenderink-van Doorn, Alex Frangi, Arjan Kuijper, Remco Duits, Bram Platel, Frans Kanters, Taizo Iijima, Ted Adelson, Pietro Perona, Jitendra Malik, David Mumford, Luis Alvarez, Markus van Almsick, William Freeman, Izumi Ohzawa, Ralph Freeman, Russell DeValois, David Hubel, Torsten Wiesel, Semir Zeki, Erik Kandel, Amiram Grinvald, Brian Wandell, Robert Rodieck and many more, as well asmany of the 3D Computer Vision / Image Sciences Institute master's students.

The careful proofreading and remarks by Michel Bister were very helpful. Markus van Almsick gave many *Mathematica* hints, insights and valuable help with editing. I thank my wife Hetty, for the patience and loving support during the endeavor to write this book.

This book originated from coursenotes of the annual graduate course on "Multiscale Image Analysis and Font-End Vision" at the Image Sciences Institute of Utrecht University 1996-2000 and at Faculty of Biomedical Engineering at Eindhoven University of Technology in the Netherlands since 2001. This course has also been given at a summerschool at CIMAT in Guanajuato, Mexico, at IMPA in Rio de Janeiro, Brazil, at the Multimedia University in Cyberjaya, Malaysia and as a tutorial at several international conferences (VBC 98, CVPR 1999, MICCAI 2001). The author was kindly hosted for two months at the new IT University at Copenhagen, Denmark in the summer of 2000 to get a lot of the writing done. The highly competent and kind atmosphere of the collaborating labs there considerably contributed to the pleasure it was to write and to program this interactive book. I learned a lot by *doing* multi-scale computer vision, and I hope the reader will do the same.

Eindhoven - Utrecht - Copenhagen, February 2003

Email: B.M.terHaarRomeny@tue.nl

1. Apertures and the notion of scale

Nothing that is seen is perceived at once in its entirety.
Euclid (~300 B.C.), Theorem I

1.1 Observations and the size of apertures

Observations are always done by *integrating* some physical property with a measurement device. Integration can be done over a spatial area, over an amount of time, over wavelengths etc. depending on the task of the physical measurement. For example, we can integrate the emitted or reflected light intensity of an object with a CCD (charge-coupled device) detector element in a digital camera, or a grain in the photographic emulsion in a film, or a photoreceptor in our eye. These 'devices' have a sensitive area, where the light is collected. This is the *aperture* for this measurement. Today's digital cameras have several million 'pixels' (*pic*ture *el*ements), very small squares where the incoming light is integrated and transformed into an electrical signal. The size of such pixels/apertures determines the maximal sharpness of the resulting picture.

An example of integration over time is sampling of a temporal signal, for example with an analog-digital converter (ADC). The integration time needed to measure a finite signal is the size of the *temporal aperture*. We always need a *finite* integration area or a *finite* integration time in order to measure a signal. It would be nice to have infinitely small or infinitely fast detectors, but then the integrated signal is zero, making it useless.

Looking with our visual system is making measurements. When we look at something, we have a range of possibilities to do so. We can look with our eyes, the most obvious choice.

We can zoom in with a microscope when things are too small for the unaided eye, or with a telescope when things are just very big.The smallest distance we can see with the naked eye is about 0.5 second of arc, which is about the distance between two neighboring cones in the center of our visual field. And, of course, the largest object we can see fills the whole retina.

It seems that for the eye (and any other measurement device) the *range* of possibilities to observe certain sizes of objects is *bounded* on two sides: there is a minimal size, about the size of the smallest aperture, and there is a maximal size, about the size of the whole detector array.

Spatial resolution is defined as the diameter of the local integration area. It is the size of the *field of view* divided by the number of samples taken over it. The spatial resolution of a Computer Tomography (CT) scanner is about 0.5 mm, which is calculated from the measurement of 512 samples over a field of view with a diameter of 25 cm.

The *temporal resolution* of a modern CT scanner is about 0.5 second, which is 2 images per second.

It seems that we are always trying to measure with the highest possible sharpness, or highest resolution. Reasons to accept lower resolution range from costs, computational efficiency, storage and transmission requirements, to the radiation dose to a patient etc. We can always reduce the resolution by taking together some pixels into one, but we cannot make a coarse image into a sharp one without the introduction of extra knowledge.

The resulting measurement of course strongly depends on the size of the measurement aperture. We need to develop strict criteria that determine objectively what aperture size to apply. Even for a fixed aperture the results may vary, for example when we measure the same object at different distances (see figure 1.1).

```
<< FrontEndVision`FEV`;
Show[GraphicsArray[
   {{Import["cloud1.gif"]}, {Import["cloud2.gif"]}}], ImageSize → 400];
```

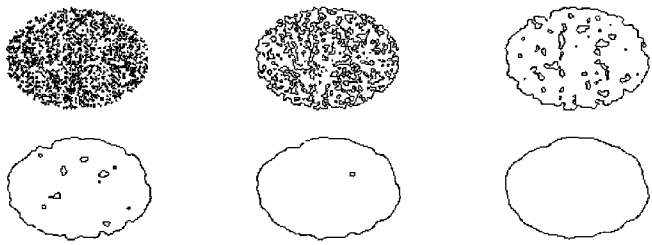


Figure 1.1 A cloud observed at different scales, simulated by the blurring of a random set of points, the 'drops'. Adapted from [Koenderink1992a].

1.2 Mathematics, physics, and vision

In *mathematics* objects are allowed to have no size. We are familiar with the notion of points, that really shrink to zero extent, and lines of zero width. No metrical *units* (like meters, seconds, amperes) are involved in mathematics, as in physics.

Neighborhoods, like necessary in the definition of differential operators, are taken into the limit to zero, so for such operators we can really speak of *local operators*. We recall the definition for the derivative of $f(x)$: $\lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$, where the limit makes the operation confined to a mathematical point.

In *physics* however this is impossible. We saw before that objects live on a bounded range of scales. When we measure an object, or look at it, we use an instrument to do this observation

(our eye, a camera) and it is the range that this instrument can see that we call the scale range. The scale range is bounded on two sides:
- the smallest scale the instrument can see is the *inner scale*. This is the smallest sampling element, such as a CCD element in a digital camera, rod or cone on our retina;

- the largest scale the instrument can see is the *outer scale*. This is the field of view. The dimension is expressed as the ratio between the outer scale and the inner scale, or how often the inner scale fits into the outer scale. Of course the bounds apply both to the detector and the measurement: an image can have a 2D dimension of 256 x 256 pixels.

Dimensional units are *essential* in physics: we express any measurement in dimensional units, like: 12 meters, 14.7 seconds, 0.02 candela/m^2 etc. When we measure (observe, sample) a physical property, we need to choose the 'stepsize' with which we should investigate the measurement. We scrutinize a microscope image in microns, a global satellite image in kilometers. In measurements there is no such thing as a physical 'point': the smallest 'point' we have is the physical sample, which is defined as the *integrated* weighted measurement over the detector area (which we call the aperture), where area is *always finite*.

How large should the sampling element be? It depends on the task at hand in what scale range we should measure: "Do we like to see the leaves or the tree"? The range of scales applies not only to the objects in the image, but also to the scale of the features. In chapter 5 we discuss in detail many such features, and how they can be constructed. We give just one example here: in figure 1.2 we see a *hierarchy* in the range of scales, illustrated here for a specific feature (the gradient).

```
im = Import["Utrecht256.gif"][[1, 1]];
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[im];
  p2 =
   ListDensityPlot[Sqrt[gD[im, 1, 0, #]^2 + gD[im, 0, 1, #]^2] & /@ {1, 2, 4}];
Show[GraphicsArray[Prepend[p2, p1]], ImageSize → 500];
```

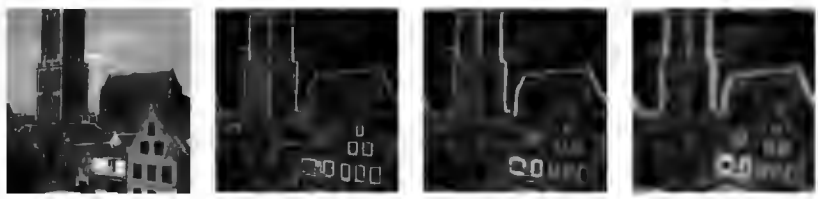


Figure 1.2 Picture of the city of Utrecht. The right three pictures show the *gradient*: the strength of borders, at a scale of 1, 2 resp. 4 pixels. At the finest scale we can see the contours of almost every stone, at the coarsest scale we see the most important edges, in terms of outlines of the larger structures. We see a *hierarchy* of structures at different scales. The *Mathematica* code and the gradient will be explained in detail in later chapters.

To expand the range say of our eye we have a wide armamentarium of instruments available, like scanning electron microscopes and a Hubble telescope. The scale range known to

humankind spans about 50 decades, as is beautifully illustrated in the book (and movie) "Powers of Ten" [Morrison1985].

```
Show[Import["Powersof10sel.gif"], ImageSize → 500];
```

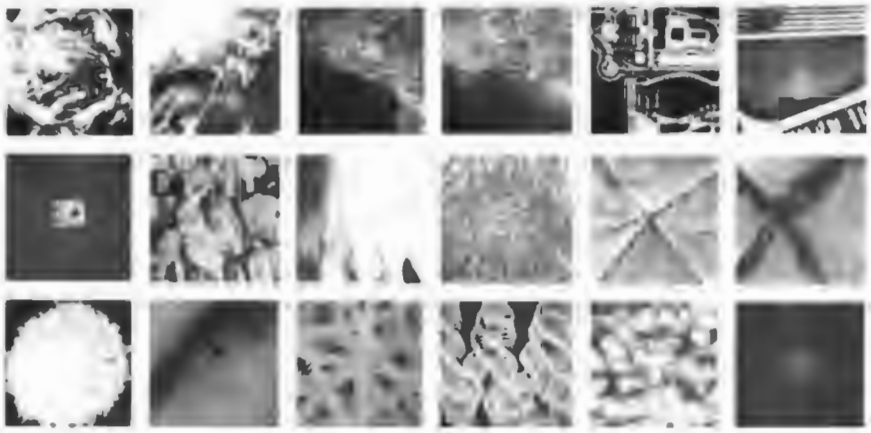


Figure 1.3 Selection of pictures from the journey through scale from the book [Morrison1985], where each page zooms in a factor of ten. Starting at a cosmic scale, with clusters of galaxies, we zoom in to the solar system, the earth (see the selection above), to a picknicking couple in a park in Chicago. Here we reach the 'human' (antropometric) scales which are so familiar to us. We then travel further into cellular and molecular structures in the hand, ending up in the quark structure of the nuclear particles. For the movie see: http://www.micro.magnet.fsu.edu/primer/java/scienceopticsu/powersof10/index.html.

In *vision* we have a system evolved to make visual observations of the outside world. The *front-end* of the (human) visual system is defined as the very first few layers of the visual system. Here a special representation of the incoming data is set up where subsequent processing layers can start from. At this stage there is no memory involved or cognitive process.

Later we will define the term 'front-end' in a more precise way. We mean the retina, lateral geniculate nucleus (LGN, a small nucleus in the thalamus in our mid-brain), and the primary visual cortex in the back of our head. In the chapter on human vision we fully elaborate on the *visual pathway*.

The front-end sampling apparatus (the receptors in the retina) is designed just to extract multi-scale information. As we will see, it does so by applying sampling apertures, at a wide range of sizes simultaneously.

There is no sampling by individual rods and cones, but by well-structured assemblies of rods and cones, the so-called '*receptive fields*'.

In chapters 6 - 9 we will study the neuroanatomy of the human front-end visual system in more detail. The concept of a receptive field was introduced in the visual sciences by

Hartline [Hartline1940] in 1940, who studied single fibre recordings in the horseshoe crab (Limulus polyphemus).

Psychophysically (psychophysics is the art of measuring the performance of our perceptual abilities through perceptual tasks) it has been shown that when viewing sinusoidal gratings of different spatial frequency the threshold modulation depth is constant (within 5%) over more than two decades.

This indicates that the visual system is indeed equipped with a large range of sampling apertures. Also, there is abundant electro-physiological evidence that the receptive fields come in a wide range of sizes. In the optic nerve leaving each eye one optic-nerve-fibre comes from one receptive field, not from an individual rod or cone.

In a human eye there are about 150 million receptors and one million optic nerve fibres.So a typical receptive field consists of an *average* of 150 receptors. Receptive fields form the elementary 'multi-scale apertures' on the retina. In the chapter on human vision we will study this neuroanatomy in more detail.

1.3 We blur by looking

Using a larger aperture reduces the resolution. Sometimes we exploit the blurring that is the result of applying a larger aperture. A classical example is *dithering*, where the eye blurs the little dots printed by a laser printer into a multilevel greyscale picture, dependent on the density of the dots (see figure 1.4).

It nicely illustrates that we can make quite a few different observations of the same object (in this case the universe), with measurement devices having different inner and outer scales. An atlas, of course, is the canonical example.

```
Show[GraphicsArray[{Import["Floyd0.gif"], Import["Floyd1.gif"]}],
  ImageSize -> 330];
```

Figure 1.4 Dithering is the representation of grayvalues through sparse printing of black dots on paper. In this way a tonal image can be produced with a laserprinter, which is only able to print miniscule identical single small high contrast dots. Left: the image as we observe it, with grayscales and no dithering. Right: Floyd-Steinberg dithering with random dot placements. [From http://sevilleta.unm.edu/~bmilne/khoros/html-dip/c3/s7/front-page.html].

A priori we have to decide on how large we should take the inner scale. The front-end vision system has no knowledge whatsoever of what it is measuring, and should be open-minded with respect to the size of the measurement aperture to apply.

```
Show[Import["wales-colordither.gif"], ImageSize -> 400];
```

Figure 1.5 En example of color-dithering in image compression. Left: the original image, 26 KByte. Middle: color dithering, effective spreading of a smaller number of color pixels so that the blurring of our perception blends the colors to the same color as in the original. Filesize 16 Kbyte. Right: enlargement of a detail showing the dithering. From http://www.digital-foundry.com/gif/workshop/dithering.shtml.

As we will see in the next section, the visual front-end measures at a multitude of aperture sizes *simultaneously*. The reason for this is found in the world around us: objects come at all sizes, and at this stage they are all equally important for the front-end.

```
Show[Import["Edlef Romeny - cherry trees.jpg"], ImageSize -> 280];
```

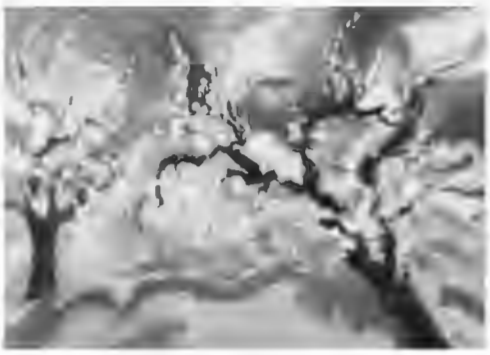


Figure 1.6 In art often perceptual clues are used, like only coarse scale representation of image structures, and dithering. Painting by Edlef ter Haar Romeny [TerHaarRomeny2002b].

```
im = Import["mona lisa face.gif"][[1, 1]];
imr1 = Table[Plus @@ Plus @@ Take[im, {y, y + 9}, {x, x + 9}],
  {y, 1, 300, 10}, {x, 1, 200, 10}];
imr2 = Table[Plus @@ Plus @@ Take[im, {y, y + 14}, {x, x + 9}],
   {y, 1, 290, 15}, {x, 1, 200, 10}];
DisplayTogetherArray[ListDensityPlot /@ {im,
    Join @@ Table[MapThread[Join, Table[imr2 imr1[[y, x]], {x, 1, 20}]],
{y, 1, 30}]}, ImageSize → 250];
```

Figure 1.7 Image mosaic of the Mona Lisa. Image resolution 200x300 pixels. The image is subsampled to 20x30 samples, whose mean intensity modulates a mosaic of the subsampled images.

And that, in a natural way, leads us to the notion of multi-scale observation, and multi-scale representation of information, which is intrinsically coupled to the fact that we can observe in so many ways. The size of the aperture of the measurement will become an extra continuous measurement dimension, as is space, time, color etc. We use it as a *free parameter:* in first instance we don't give it a value, it can take any value.

▲ Task 1.1 Experiment with dithering with circular disks of proper size in each pixel. Calculate the area the disk occupies. Some example code to get started:
```
Show[Graphics[Table[Disk[{x,y},.3+im[[y,x]]/2048],{y,1,128}
,{x,1,128}]],AspectRatio→Automatic];
```

▲ Task 1.2 Experiment with dithering with randomly placed small dots in each pixel.

Mosaics, known since Roman times, employ this multiresolution perceptive effect. There is also artistic interest in replacing a pixel by a complete image (see e.g. figure 1.7). When random images with appropriate average intensity and color (and often intensity gradient) are chosen the technique is called an *image mosaic.*

▲ Task 1.3 One can play with other graphical elements, e.g. text (BasicBlock:→ (Text["FEV", #1,#2]&)) etc. Note that despite the structure in the dithering elements, we still perceive the large scale structure unchanged in depth.

It turns out that there is a very specific reason to *not only* look at the highest resolution. As we will see in this book, a new world opens when we consider a measurement of the outside world at all these sizes simultaneously, at a whole range of sharpnesses. So, not only the smallest possible pixel element in our camera, but a camera with very small ones, somewhat

larger ones, still larger ones and so on. It turns out that our visual system takes this approach. The stack of images taken at a range of resolutions is called a *scale-space*.

Another interesting application of dithering is in the generation of random dot stereograms (RDS), see figure 1.8.

```
Options[RDSPlot] = {BasicBlock ↦ (Rectangle[#1 - #2, #1 + #2] &)};
RDSPlot[expr_, {x_, xmin_, xmax_}, {y_, ymin_, ymax_}, opts___] :=
  Block[{pts = 120, periods = 6, zrange = {-1, 1}, density = .4, depth = 1,
    basicblock = BasicBlock /. {opts} /. Options[RDSPlot], guides = True,
    strip, xpts, ypts, dx, dy, xval, yval, zmin, zmax, exprnorm},
   {zmin, zmax} = zrange; {xpts, ypts} = If[Length[pts] == 2, pts, {pts, pts}];
   dy = (ymax - ymin) / ypts; dx = (xmax - xmin) / xpts; strip = Floor[xpts / periods] dx;
   exprnorm = (.25 depth (xmax - xmin) / (periods (zmax - zmin))) *
     (Max[zmin, Min[zmax, expr]] - (zmax + zmin) / 2);
   Graphics[{RDSArray[basicblock, {dx, dy} / 2, Flatten[Table[If[Random[] < density,
        Thread[{rdsimages[exprnorm /. y → yval, {x, xval, xmax, strip}], yval}], {}],
       {yval, ymin + .5 dy, ymax, dy}, {xval, xmin + .5 dx,
        rdsimage[exprnorm /. y → yval, {x, xmin, strip}], dx}], 2]],
     If[guides, makeguides[{.5 xmax + .5 xmin, 1.1 ymin - .1 ymax}, .5 strip], {}]},
    Sequence @@ Select[{opts}, ! MemberQ[First /@ Options[RDSPlot], First[#]] &]]];

rdsimage[expr_, {x_, xval_, dx_}] := xval + dx - N[expr /. x → xval + dx / 2];
rdsimages[expr_, {x_, xval_, xmax_, dx_}] := If[xval ≤ xmax,
    Prepend[rdsimages[expr, {x, rdsimage[expr, {x, xval, dx}], xmax, dx}], xval], {}];
makeguides[pos_, size_] := Apply[Rectangle,
   Map[pos + size # &, {{{-1.1, -.1}, {-.9, .1}}, {{.9, -.1}, {1.1, .1}}}, {2}], 1];
Unprotect[Display]; Display[channel_, graphics_ ? (! FreeQ[#, RDSArray] &)] := (Display[
   channel, graphics /. (RDSArray[basicblock_, dims_, pts_] :→ (basicblock[#, dims] & /@ pts))];
  graphics); Protect[Display];
```

$$\texttt{Show}\left[\texttt{RDSPlot}\left[-\frac{2}{\sqrt{2\,\pi}}\,\texttt{x Exp}\left[-\frac{\texttt{x}^2+\texttt{y}^2}{2}\right],\ \{\texttt{x, -3, 3}\},\ \{\texttt{y, -3, 3}\}\right],\ \texttt{ImageSize} \to \texttt{400}\right];$$

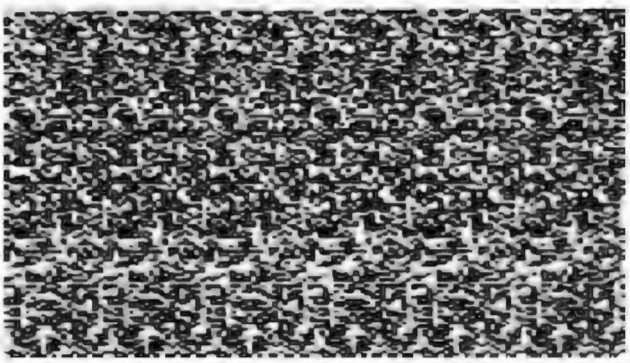


Figure 1.8 Random dot stereogram (of the first derivative with respect to x of the Gaussian function, a function which we will encounter frequently in this book). The dots are replaced by a random draw from the letters A-Z.
Code by Bar-Natan [Bar-Natan1991, www.ma.huji.ac.il/~drorbn/].
See also www.ccc.nottingham.ac.uk/~etzpc/sirds.html. Look with both eyes to a point behind the image, so the dots under the figure blend together. You will then see the function in depth.

See also the peculiar paintings of the Italian painter Giuseppe Arcimboldo (1527-1593). See www.illumin.co.uk/svank/biog/arcim/arcidx.html.

```
Show[Import["Vertumnus.jpg"], ImageSize -> 170];
```

Figure 1.9 Vertumnus (Rudolph II) by Giuseppe Arcimboldo (ca. 1590). Painting in the Skoklosters Slott, Stockholm, Sweden.

1.4 A critical view on observations

Let us take a close look at the process of observation. We note the following:

- Any physical observation is done through an aperture. By necessity this aperture has to be *finite*. If it would be zero size no photon would come through. We can modify the aperture considerably by using instruments, but never make it of zero width. This leads to the fundamental statement: *We cannot measure at infinite resolution*. We only can perceive a 'blurred' version of the mathematical abstraction (infinite resolution) of the outside world.

- In a first 'virginal' measurement like on the retina we like to carry out observations that are *uncommitted*. With uncommitted we mean: not biased in any way, and with no model or any a priori knowledge involved. Later we will fully incorporate the notion of a model, but in this first stage of observation *we know nothing*.

An example: when we know we want to observe vertical structures such as stems of trees, it might be advantageous to take a vertically elongated aperture. But in this early stage we cannot allow such special apertures.

At this stage the system needs to be general. We will exploit this notion of being uncommitted in the sequel of this chapter to the establishment of *linear scale-space theory*.

It turns out that we can express this 'uncommitment' into axioms from which a physical theory can be derived. Extensions of the theory, like nonlinear scale-space theories, follow in a natural way through relaxing these axioms.

- Being uncommitted is a natural requirement for the first stage, but *not* for further stages, where extracted information, knowledge of model and/or task etc. come in. An example: the introduction of feedback enables a multi-scale analysis where the aperture can be made adaptive to properties measured from the data (such as the strength of certain derivatives of the data). This is the field of *geometry-driven diffusion*, a nonlinear scale-space theory. This will be discussed in more detail after the treatment of linear scale-space theory.

```
Show[Import["DottedPR.gif"], ImageSize -> 380];
```

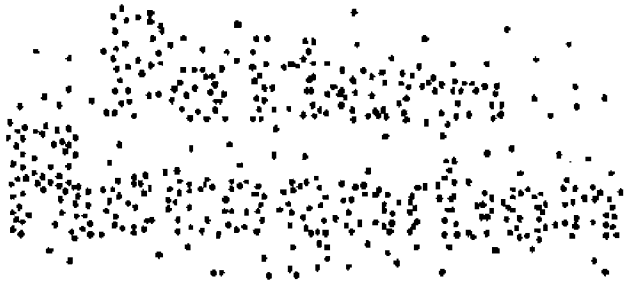


Figure 1.10 At different resolutions we see different information. The meaningful information in this image is at a larger scale then the dots of which it is made. Look at the image from about 2 meters. Source: dr. Bob Duin, Pattern Recognition Group, Delft University, the Netherlands.

- A *single constant size* aperture function may be sufficient in a controlled physical application. An example is a picture taken with a camera or a medical tomographic scanner, with the purpose to replicate the pixels on a screen, paper or film without the need for cognitive tasks like recognition. Note that most man-made devices have a single aperture size. If we need images at a multiple of resolutions we simply blur the images after the measurement.

- The human visual system measures at multiple resolutions *simultaneously*, thus effectively adding scale or resolution as a measurement dimension. It measures a *scale-space* $L(x, y; \sigma)$, a function of space (x, y) and scale σ, where L denotes the measured parameter (in this case luminance) and σ the size of the aperture. In a most general observation *no* a priori size is set, we just don't know what aperture size to take. So, in some way control is needed: we could apply a whole range of aperture sizes if we have no preference or clue what size to take.

- When we observe noisy images we should realize that noise is always part of the observation. The term 'noisy image' already implies that we have some idea of an image with structure 'corrupted with noise'. In a measurement noise can only be separated from the observation if we have a model of the structures in the image, a model of the noise, or a model of both. Very often this is not considered explicitly.

```
im = Table[If[11 < x < 30 && 11 < y < 30, 1, 0] + 2 Random[], {x, 40}, {y, 40}];
ListDensityPlot[im, FrameTicks -> False, ImageSize -> 120];
```

Figure 1.11 A square with additive uniform pixel-uncorrelated noise. Jagged or straight contours? 'We think it is' or 'it looks like' a square embedded in the noise. Without a model one really cannot tell.

- When it is given that objects are human-made structures like buildings or otherwise part of computer vision's 'blocks world', we may assume straight or smoothly curved contours, but often this is not known.

- Things often go wrong when we change the resolution of an image, for example by creating larger pixels.

- If the apertures (the pixels) are square, as they usually are, we start to see blocky tesselation artefacts. In his famous paper "The structure of images" Koenderink coined this *spurious resolution* [Koenderink1984a], the emergence of details that were not there before, and should not be there. The sharp boundaries and right angles are artefacts of the representation, they certainly are not in the outside world data. Somehow we have created structure in such a process. Nearest neighbour interpolation (the name for pixel replication) is of all interpolation methods fastest but the worst. As a general rule we want the structure only to decrease with increasing aperture.

```
Show[Import["Einsteinblocky.gif"], ImageSize -> 120];
```

Figure 1.12 Spurious resolution due to square apertures. Detail of a famous face: Einstein Much unintended 'spurious' information has been added to this picture due to the sampling process. Intuitively we take countermeasures for such artefacts by squeezing our eyes and looking through our eyelashes to blur the image, or we look from a greater distance.

- In the construction of fonts and graphics *anti-aliasing* is well known: one obtains a much better perceptual delineation of the contour if the filling of the pixel is equivalent to the physical integration of the intensity over the area of the detector. See figure 1.13 for a font example.

```
Show[Import["anti_alias.gif"], ImageSize -> 250];
```

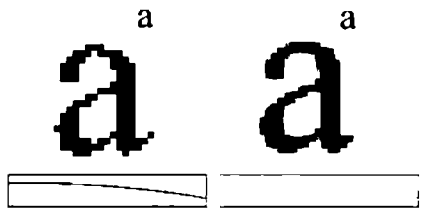


Figure 1.13 Anti-aliasing is the *partial volume effect* at the boundaries of contours. When making physically realistic test images for computer vision applications it is essential to take this sampling effect into account.

1.5 Summary of this chapter

Observations are necessarily done through a finite aperture. Making this aperture infinitesimally small is not a physical reality. The size of the aperture determines a hierarchy of structures, which occur naturally in (natural) images. With the help of instruments (telescopes, microscopes) we are able to see a scale-range of roughly 10^{50}. The visual system exploits a wide range of such observation apertures in the front-end simultaneously, in order to capture the information at all scales. Dithering is a method where the blending/blurring through an observation with a finite aperture is exploited to create grayscale and color nuances which can then be created with a much smaller palet of colors.

Observed noise is part of the observation. There is no way to separate the noise from the data if a model of the data, a model of the noise or a model of both is absent. Without a model noise is considered input which also contains structural geometric information, like edges, corners, etc. at all scales.

The aperture cannot take any form. An example of a wrong aperture is the square pixel so often used when zooming in on images. Such a representation gives rise to edges that were never present in the original image. This artificial extra information is called 'spurious resolution'. In the next chapter we derive from first principles the best and unique kernel for an uncommitted observation.

2. Foundations of scale-space

"There are many paths to the top of the mountain, but the view is always the same" -Chinese proverb.

2.1 Constraints for an uncommitted front-end

To compute any type of representation from the image data, information must be extracted using certain *operators* interacting with the data. Basic questions then are: Which operators to apply? Where to apply them? How should they look like? How large should they be?

Suppose such an operator is the derivative operator. This is a difference operator, comparing two neighboring values at a distance close to each other. In mathematics this distance can indeed become infinitesimally small by taking the limit of the separation distance to zero, but in physics this reduces to the sampling distance as the smallest distance possible. Therefore we may foresee serious problems when we deal with such notions as mathematical differentiation on discrete data (especially for high order), and sub-pixel accuracy.

From this moment on we consider the aperture function as an operator: we will search for *constraints* to pin down the exact specification of this operator. We will find an important result: for an unconstrained front-end there is a *unique* solution for the operator. This is the Gaussian kernel $g(x;\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$, with σ the *width* of the kernel. It is the same bell-shaped kernel we know from probability theory as the probability density function of the *normal distribution*, where σ is the *standard deviation* of the distribution.

Interestingly, there have been many derivations of the front-end kernel, all leading to the unique Gaussian kernel.

This approach was pioneered by Iijima (figure 2.2) in Japan in the sixties [Iijima1962], but was unnoticed for decades because the work was in Japanese and therefore inaccessible for Western researchers.

Independently Koenderink in the Netherlands developed in the early eighties a rather complete multi-scale theory [Koenderink1984a], including the derivation of the Gaussian kernel and the linear diffusion equation.

```
<< FrontEndVision`FEV`;
```

```
σ = 1; Plot[ 1/√(2 π σ²) E^(-x²/(2 σ²)), {x, -4, 4}, ImageSize -> 200];
```

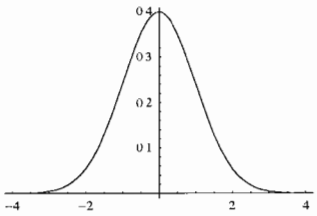


Figure 2.1 The Gaussian kernel with unit standard deviation in 1D.

Koenderink was the first to point out the important relation to the receptive field families in the visual system, as we will discuss in forthcoming chapters. Koenderink's work turned out to be monumental for the development of scale-space theory. Lindeberg pioneered the field with a tutorial book [Lindeberg1994a]. The papers by Weickert, Ishikawa and Imija (who together discovered this Japanese connection) present a very nice review on these early developments [Weickert1997a, Weickert1999a].

```
Show[Import["Iijima.gif"], ImageSize -> 150];
```

Fig. 2.2 Prof. Taizo Iijima, emeritus prof. of Tokyo Technical University, Japan, was the first to publish the axiomatic derivation of 'the fundamental equation of figure'.

We will select and discuss two fundamentally different example approaches to come to the Gaussian kernel in this book:

1. An axiomatic approach based on dimensional analysis and the notion of having 'no preferences' (section 2.2);
2. An approach based on the maximization of local entropy in the observation (section 2.5);

2.2 Axioms of a visual front-end

The line of reasoning presented here is due to Florack et al. [Florack1992a]. The requirements can be stated as *axioms*, or postulates for an uncommitted visual front-end. In essence it is the mathematical formulation for being uncommitted: "we know nothing", or "we have no preference whatsoever".

• **linearity**: we do not allow any nonlinearities at this stage, because they involve knowledge of some kind. So: no knowledge, no model, no memory;

• **spatial shift invariance**: no preferred location. Any location should be measured in the same fashion, with the same aperture function;

• **isotropy**: no preferred orientation. Structures with a particular orientation, like vertical trees or a horizontal horizon, should have no preference, any orientation is just as likely. This necessitates an aperture function with a circular integration area.

• **scale invariance**: no preferred size, or scale of the aperture. Any size of structure, object, texture etc. to be measured is at this stage just as likely. We have no reason to look only through the finest of apertures. The visual world consists of structures at any size, and they should be measured at any size.

In order to use these constraints in a theory that sets up the reasoning to come to the aperture profile formula, we need to introduce the concept of dimensional analysis.

2.2.1 Dimensional analysis

Every physical unit has a *physical dimension.*

It is this that mostly discriminates physics from mathematics. It was Baron Jean-Baptiste Fourier who already in 1822 established the concept of dimensional analysis [Fourier1955]. This is indeed the same mathematician so famous for his Fourier transformation.

```
Show[Import["Fourier.jpg"], ImageSize → 140];
```

Figure 2.3 Jean-Baptiste Fourier, 1792-1842.

Fourier described the concept of dimension analysis in his memorable work entitled "Théorie analytique de la chaleur" [Fourier1955] as follows: *"It should be noted that each physical quantity, known or unknown, possesses a dimension proper to itself and that the terms in an equation cannot be compared one with another unless they possess the same dimensional exponent"*.

When a physicist inspects a new formula he invariably checks first whether the dimensions are correct. It is for example impossible to add meters to meters/second. One of the most fundamental laws in physics is that the physical laws should be rock solid, independent of a chosen description, anywhere and anytime. This law is the *law of scale invariance*, which indicates that we have full freedom of reparametrization:

[Law of Scale Invariance] **Physical laws must be independent of the choice of fundamental parameters.**

'Scale invariance' here refers to the notion of scale with respect to dimensional units (remember the microns, kilometers or milliseconds as the aperture size of the measurement instrument).

In essence the law of scale invariance states that the left and right part of the equation of a physical equation should have the same dimensional units, and they should describe the same process, whether expressed in Cartesian or polar coordinates.

Core in dimensional analysis is that when the dimensions in a complex physical system are considered, only a limited number of dimensionless combinations can be made: the basis or null-space of the system. It is an elegant and powerful tool to find out basic relations between physical entities, or even to *solve* a problem. It is often a method of first choice, when no other information is available. It is often quite remarkable how much one can deduct by just using this technique. We will use dimensional analysis to establish the expression defining the basic linear isotropic scale-space kernel. First some examples which illustrate the idea.

2.2.2 The cooking of a turkey

```
Show[Import["Turkey.gif"], ImageSize -> 150];
```

This example is taken from the delightful paper by Geoffrey West [West1988]. When cooking a turkey, or a goose, there is the problem of knowing how long to cook the bird in the oven, given the considerable variation that can exist in its weight and size.

Many (inferior) cookbooks specify simply something like '20 minutes per pound', implying a linear relation. There are superior cookbooks, however, such as the 'Better Homes and

Gardens Cookbook' that recognize the nonlinear nature of this relationship. In figure 1.4 we have adapted the graph from this cookbook, showing a log-log plot of the cooking time as a function of the weight of the turkey. The slope of the linear relation is about 0.6. It turns out that we can predict this relation just by dimensional analysis.

```
data = {{5, 3}, {7, 3.8}, {10, 4.5}, {14, 5}, {20, 6}, {24, 7}};
LogLogListPlot[data, PlotJoined -> False,
  Ticks -> {{5, 10, 15, 20}, {3, 5, 7}}, Frame -> True,
  FrameLabel -> {"Cooking time (hrs)", "Weight (lb)"},
  PlotStyle -> PointSize[0.02], ImageSize -> 220];
```

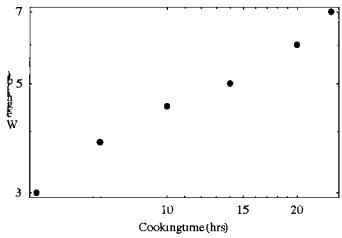


Figure 2.4. Turkey cooking time as a (nonlinear) function of turkey weight. The slope of the log-log plot is about 0.6. (Based on a table in *Better Homes and Gardens Cookbook*, Des Moines: Meridith Corp., 1962, p. 272).

Let us list the physical quantities that are involved:
- the temperature distribution inside the turkey T, in degrees Kelvin,
- the oven temperature T_0 (both measured relative to the outside air temperature), in degrees Kelvin,
- the bird density ρ in kg/m^3,
- the diffusion coefficient of the turkey κ from Fourier's diffusion equation for T: $\frac{\partial T}{\partial t} = \kappa \Delta T$ where Δ is the Laplacian operator $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$, in m^2/sec,
- the weight of the turkey W, in kg,
- and the time t in seconds.

In general, for the dimensional quantities in this problem, there will be a relationship of the form $T = f(T_0, W, t, \rho, \kappa)$. We can make a matrix of the units and their dimensions:

```
m = {{0, 0, 0, 0, -3, 2},
   {0, 0, 0, 1, 0, -1}, {0, 0, 1, 0, 1, 0}, {1, 1, 0, 0, 0, 0}};
TableForm[m, TableHeadings -> {{"length", "time", "mass", "degree"},
   {"T0", "T", "W", "t", "ρ", "κ"}}]
```

	T0	T	W	t	ρ	κ
length	0	0	0	0	-3	2
time	0	0	0	1	0	-1
mass	0	0	1	0	1	0
degree	1	1	0	0	0	0

A matrix can be described with a basis spanned by basis vectors, whose linear combinations satisfy the matrix equation `m.x == 0`. The command `NullSpace` gives us the list of basis vectors:

```
NullSpace[m]
```

```
{{0, 0, -2, 3, 2, 3}, {-1, 1, 0, 0, 0, 0}}
```

The (famous) Pi-theorem in dimensional analysis by Buckingham (see http://www.treasure-troves.com/physics/BuckinghamsPiTheorem.html) states that one can make as many independent dimensionless combinations of the physical variables in the system under study as the number of basis vectors of the nullspace of the dimension matrix **m**. These are determined by the nullspace.

So, for the turkey problem we can only construct two independent dimensionless quantities (just fill in the exponents given by the basis vectors): $\frac{\rho^2 t^3 \kappa^3}{W^2}$ and $\frac{T}{T_0}$.

So, from the nullspace vector **{-1,1,0,0,0,0}** we found $\frac{T}{T_0}$ and from **{0,0,-2,3,2,3}** we found $\frac{\rho^2 t^3 \kappa^3}{W^2}$. Because both these quantities are dimensionless one must be expressible in the other, giving the relationship: $\frac{T}{T_0} = f\left(\frac{\rho^2 t^3 \kappa^3}{W^2}\right)$. Note that since the lefthand side is dimensionless, the arbitrary function f must be a dimensionless function of a dimensionless variable. This equation does not depend on the choice of units, since dimensionless units remain invariant to changes in scale: the scale invariance.
The graph in the cookbook can now be understood: when geometrically similar birds are considered, cooked to the same temperature distribution at the same oven temperature, there will be the following scaling law: $\frac{\rho^2 t^3 \kappa^3}{W^2}$ = constant. If the birds have the same physical characteristics, which means the same ρ and κ, we find that $t^3 = \frac{W^2}{\rho^2 \kappa^3}$, so the cooking time t is proportional to $W^{2/3}$ which nicely explains the slope.

2.2.3 Reynold's number

From [Olver1993] we take the example of the Reynold's number. We study the motion of an object in some fluid.

As physical parameters we have the resistance D of the object (in $\frac{\text{kg}}{ms^2}$), the fluid density ρ (in $\frac{\text{kg}}{m^3}$), the velocity relative to the fluid v (in $\frac{m}{s}$), the object diameter d (in m) and the fluid viscosity μ (in $\frac{\text{kg}}{ms}$). The dimension matrix becomes then:

```
m = {{-3, 1, 1, -1, -1}, {0, -1, 0, -1, -2}, {1, 0, 0, 1, 1}};
TableForm[m,
 TableHeadings -> {{"meter", "second", "kg"}, {"ρ", "v", "d", "μ", "D"}}]
```

	ρ	v	d	μ	D
meter	-3	1	1	-1	-1
second	0	-1	0	-1	-2
kg	1	0	0	1	1

We calculate the nullspace:

```
NullSpace[m]
```

```
{{-1, -2, 0, 0, 1}, {-1, -1, -1, 1, 0}}
```

From the nullspace we easily find the famous Reynolds number: $R = \frac{\rho v d}{\mu}$. The other dimensionless entity $\frac{D}{\rho v^2}$ is the friction factor.

2.2.4 Rowing: more oarsmen, higher speed?

```
Show[Import["Rowerswanted.gif"], ImageSize → 210];
```

Another illuminating example is the problem of the number of oarsmen in a competition rowing boat: Do 8 oarsmen need less time to row a certain distance, say 2000 meter, then a single skiffer, despite the fact that the water displacement is so much bigger? Let's study the physics again: We first find the relation for the drag force F on a ship with length l moving with velocity v through a viscous fluid with viscosity μ and density ρ.

The final term to take into account in this physical setup is the gravity g. Again we can make a dimensional matrix for the six variables involved:

```
m = {{1, 1, 1, -1, -3, 1}, {-2, 0, -1, -1, 0, -2}, {1, 0, 0, 1, 1, 0}};
TableForm[m, TableHeadings ->
  {{"meter", "second", "kg"}, {"F", "l", "v", "μ", "ρ", "g"}}]
```

	F	l	v	μ	ρ	g
meter	1	1	1	-1	-3	1
second	-2	0	-1	-1	0	-2
kg	1	0	0	1	1	0

Figure 2.5 Dimensional matrix for the physics of drag of an object through water. F is the drag force, l resp v are the length resp. the velocity of the ship, μ is the viscosity of the water.

and study the nullspace:

```
NullSpace[m]
```

```
{{0, 1, -2, 0, 0, 1}, {-1, 2, 2, 0, 1, 0}, {-1, 1, 1, 1, 0, 0}}
```

```
rowdata1 = {{1, 6.88}, {2, 6.80}, {4, 6.60}, {8, 5.80}};
rowdata2 = {{1, 7.01}, {2, 6.85}, {4, 6.50}, {8, 5.85}};
rowdata3 = {{1, 7.04}, {2, 6.85}, {4, 6.40}, {8, 5.95}};
rowdata4 = {{1, 7.10}, {2, 6.95}, {4, 6.50}, {8, 5.90}};
```

```
MultipleListPlot[rowdata1, rowdata2,
  rowdata3, rowdata4, Ticks -> {{1, 2, 4, 8}, Automatic},
  AxesLabel -> {"# of\noarsmen", "Time for\n2000 m (min)"},
  PlotJoined -> True, PlotRange -> {Automatic, {5, 8}}];
```

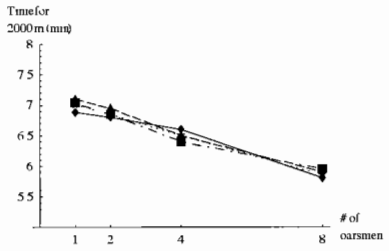


Figure 2.5. The results of best US regatta rowing (2000 m) of Summer 2000 for different numbers of oarsmen. The slope of the graph is about -1/9. Source: http://rowingresults.com/

The dimensionless units are: $\frac{v^2}{lg}$ (Froude's number), $\frac{F}{\rho v^2 l^2}$ (the pressure coefficient) and $\frac{l v \mu}{F}$ (the Poisseuille coefficient). So we have $\frac{F}{\rho v^2 l^2} = f\left(\frac{v^2}{lg}, \frac{l v \mu}{F}\right)$ or $F \approx \rho v^2 l^2 f$ where f is a dimensionless number. The power E produced by the n oarsmen together to overcome the drag force F is given by $F v$. Thus $E = F v = \rho v^3 l^2 f \approx n$ because E is directly proportional to n.

The weight W of a ship is proportional to the volume of displaced water (Archimedes law). so $W \approx l^3$. This implies $\frac{F}{W} \approx \frac{1}{l}$ which means that larger ships have advantages, because, for similar bodies, the ratio $\frac{F}{W}$ decreases as the size of the ship increases. We know $\rho = 1$ for water and $W \approx l^3$ (Archimedes again) and $W \approx n$ in good approximation, we find $v^3 \approx n^{1/3}$ so $v \approx n^{1/9}$. So eight oarsmen indeed go faster. though little, than less oarsmen.

There are several nice references to the technique of dimensional analysis [West1988, Pankhurst1964a, Olver1993], often with quite amusing examples, some of which were presented in this section.

Archimedes' Number	Froude Number	Monin - Obukhov Length
Bingham Number	Grashof Number	Nusselt Number
Biot Number	Internal Froude Number	Péclet Number
Boussinesq Number	Mach Number	Prandtl Number
Critical Rayleigh Number	Magnetic Reynolds Number	Rayleigh Number
Ekman Number	Mason Number	Richardson Number
Fresnel Number	Moment of Inertia Ratio	Timescale Number

Figure 2.6. A list of famous dimensional numbers. From Eric Weisstein's World of Physics. URL: http://scienceworld.wolfram.com/physics/topics/UnitsandDimensionalAnalysis.html.

Many scaling laws exist. In biology scaling laws have become a powerful technique to find surprising relations (see for a delightful easy-to-read overview the book by McMahon and Bonner [McMahon1983] and the classical book by Thompson [Thompson1942]).

2.3 Axiomatic derivation of the Gaussian kernel

The dimensional analysis discussed in the previous section will now be used to derive the Gaussian kernel as the unique solution for the aperture function of the uncommitted visual front-end.

We do the reasoning in the Fourier domain, as this turns out to be easier and leads to smaller equations. We give the theory for 2D. We will see that expansion to other dimensionalities is straightforward. We use scripted symbols for variables in the Fourier domain. We consider 'looking through an aperture'. The matrix m and the nullspace become:

```
m = {{1, -1, -2, -2}, {0, 0, 1, 1}};
TableForm[m,
 TableHeadings -> {{"meter", "candela"}, {"σ", "ω", "L0", "L"}}]
```

	σ	ω	L0	L
meter	1	-1	-2	-2
candela	0	0	1	1

Figure 2.8 Dimensional matrix for the physics of observation through an aperture.

```
NullSpace[m]
```

```
{{0, 0, -1, 1}, {1, 1, 0, 0}}
```

were σ is the size of the aperture, ω the spatial coordinate (frequency in the Fourier domain), $\mathcal{L}_0$ the luminance of the outside world (in candela per square meter: cd/m^2), and $\mathcal{L}$ the luminance as processed in our system. The two dimensionless units $\frac{\mathcal{L}}{\mathcal{L}_0}$ and $\sigma\omega$ can be expressed into each other: $\frac{\mathcal{L}}{\mathcal{L}_0} = \mathcal{G}(\sigma\omega)$, where $\mathcal{G}$ is the kernel (filter, aperture) function in the Fourier domain to be found (the Gaussian kernel we are after). We now plug in our constraints, one by one.

No preference for location, together with the prerequisite for linearity, leads to the recognition of the process as a *convolution*. The aperture function is shifted over the whole image domain, with no preference for location: any location is measured (or filtered, observed) with the same aperture function (kernel, template, filter, receptive field: all the same thing). This is written for the spatial domain as:

$$L(x, y) = L_0(x, y) \otimes G(x, y) \equiv \int_{-\infty}^{\infty} L_0(u, v)\, G(x-u, y-v)\, du\, dv$$

In the Fourier domain, a convolution of functions translates to a regular product between the Fourier transforms of the functions: $\mathcal{L}(\omega_x, \omega_y) = \mathcal{L}_0(\omega_x, \omega_y) \,.\, \mathcal{G}(\omega_x, \omega_y)$

The axiom of isotropy translates into the fact that we now only have to consider the *length* $\|\vec{\omega}\|$ of our spatial frequency vector $\vec{\omega} = \{\omega_x, \omega_y\}$: $\omega = \|\vec{\omega}\| = \sqrt{\omega_x^2 + \omega_y^2}$. This is a scalar.

The axiom of scale-invariance is the core of the reasoning: when we observe (or blur) an observed image again, we get an image which is blurred with the *same* but wider kernel:

$\mathcal{G}(\omega\sigma_1)\mathcal{G}(\omega\sigma_2) = \mathcal{G}(\omega\sigma_1 + \omega\sigma_2)$. Only the exponential function is a general solution of this equation: $\mathcal{G}(\omega\sigma) = \exp((\alpha\omega\sigma)^p)$ where α and p are some arbitrary constants.
We must raise the argument here to some power p because we are dealing with the *dimensionless* parameter $\omega\sigma$. In general, we don't know α or p, so we apply the following constraint: isotropy.
The dimensions are independent, thus separable: $\|\vec{\omega}\sigma\| = (\omega_1\sigma)\vec{e}_1 + (\omega_1\sigma)\vec{e}_2 + \ldots$ where the $\vec{e}_i$ are the basis unit coordinate vectors. Recall that the vector $\vec{\omega}$ ($\vec{\omega} = \omega_x\vec{e}_x + \omega_y\vec{e}_y + \omega_x\vec{e}_z$) in the Fourier domain is the set of spatial frequencies in the spatial dimensions. The magnitude of $\|\vec{\omega}\sigma\|$ is calculated by means of Pythagoras from the projections along $\vec{e}_i$, so we add the squares: $p = 2$. We further demand the solution to be real, so α^2 is real. We notice that when we open the aperture fully, we blur everything out, so $\lim_{\sigma\omega \downarrow 0} \mathcal{G}(\omega\sigma) \to 0$. This means that α^2 must be negative. We choose $\alpha = -\frac{1}{2}$. As we will see, this (arbitrary) choice gives us a concise notation of the diffusion equation. So we get: $\mathcal{G}(\vec{\omega}, \sigma) = \exp(-\frac{1}{2}\sigma^2\omega^2)$. We go to the spatial domain with the inverse Fourier transform:

```
Clear[σ];
g[x_, σ_] = Simplify[InverseFourierTransform[Exp[-σ^2 ω^2/2], ω, x], σ > 0]
```

$$\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma}$$

The last notion to use it that we want a *normalized* kernel. The integrated area under the curve must be unity:

```
Simplify[Integrate[e^(-x^2/(2 σ^2))/σ, {x, -∞, ∞}], σ > 0]
```

$$\sqrt{2\pi}$$

We divide by this factor. so we finally find for the kernel: $G(\vec{x}, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{\vec{x}.\vec{x}}{2\sigma^2}\right)$. This is the *Gaussian kernel*, which is the Green's function of the linear, isotropic *diffusion equation* $\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2} = L_{xx} + L_{yy} = \frac{\partial L}{\partial t}$, where $t = 2\sigma^2$ is the variance.

Note that the 'derivative to scale' in the diffusion equation (as it is typically called) is the derivative to $2\sigma^2$, which also immediately follows from a consideration of the dimensionality of the equation. The variance t has the dimensional unit of m^2. The original image is the *boundary condition* of the diffusion: it starts 'diffusing' from there.Green's functions are named in honor of the English mathematician and physicist George Green (1793-1841).

So from the prerequisites 'we know nothing', the axioms from which we started, we have found the Gaussian kernel as the *unique* kernel fulfilling these constraints. This is an important result, one of the cornerstones in scale-space theory. There have been more ways in which the kernel could be derived as the unique kernel. Weickert [Weickert1997a] gives a

systematic and thorough overview of the historical schemes that have been published to derive the Gaussian.

2.4 Scale-space from causality

Koenderink presented in his famous paper "The structure of images" [Koenderink1984a] the elegant and concise derivation of the linear diffusion equation as the generating partial differential equation for the construction of a scale-space.

The arguments were taken from the physics of *causality*: when we increase the scale and blur the image further, we have the situation that the final blurred image is completely *caused by* the image we started from.

The previous level of scale is the cause of events at the next level. We first discuss the situation in 1D.

```
Clear[f]; f[x_] := Sin[x] + Sin[3 x]; gr = Plot[f[x], {x, -3, 3}, Epilog ->
 {Arrow[{x, f[x]}, {x, f[x] + Sign[f''[x] ] .5}] /. Solve[f'[x] == 0, x]},
  AxesLabel -> {"x", "intensity"}, ImageSize -> 200];
```

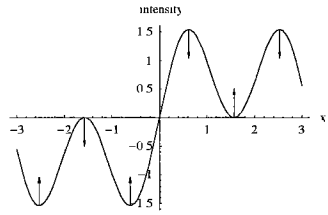


Figure 2.9 Under causal blurring signals can only change in the direction of less structure. Generation of new structure is impossible, so the signal must always be closed to above (seen from both sides of the signal). The arrows indicate the direction the intensity moves under blurring.

A higher level of scale contains always less structure. It is physically not possible that new structure is being generated. This is one of the most essential properties of a scale-space. We will encounter this property again when we consider nonlinear scale-spaces in chapter 19.

The direction of the arrows in figure 2.9 are determined by the fact if the extremum is a maximum or a minimum. In a maximum the intensity is bound to decrease, in a minimum the intensity is bound to increase. The second order derivative determines the *curvature* of the signal, and the sign determines whether the function is locally convex (in a maximum) or concave (in a minimum). We have the following conditions:

maximum: $\frac{\partial^2 u}{\partial x^2} < 0$, $\frac{\partial u}{\partial t} < 0$, intensity always decreasing;
minimum: $\frac{\partial^2 u}{\partial x^2} > 0$, $\frac{\partial u}{\partial t} > 0$, intensity always increasing.

These conditions can be summarized by $\frac{\partial^2 u}{\partial x^2} \frac{\partial u}{\partial t} > 0$.

The most important property to include next is the requirement of linearity: the second order derivative to space $\frac{\partial^2 u}{\partial x^2}$ is linearly related to the first order derivative to scale $\frac{\partial u}{\partial t}$, so: $\frac{\partial^2 u}{\partial x^2} = \alpha \frac{\partial u}{\partial t}$. We may resample any scale axis in such a way that $\alpha = 1$ so we get $\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}$. This is the 1D *linear isotropic diffusion equation*, an important result. The Green's function of the linear diffusion equation is the Gaussian kernel $\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$, which means that any function upon which the diffusion is applied, is convolved with this Gaussian kernel.

We can check (with $t = \frac{1}{2}\sigma^2$, the double == means equation, test of equality, not assignment):

```
Clear[x, t]; ∂x,x (1/√(4 π t) E^(-x^2/(4 t))) == ∂t (1/√(4 π t) E^(-x^2/(4 t))) // Simplify
```

```
True
```

Also any spatial derivative of the Gaussian kernel is a solution. We test this for the first order derivative:

```
Clear[x, t]; ∂x,x (∂x (1/√(4 π t) E^(-x^2/(4 t)))) == ∂t (∂x (1/√(4 π t) E^(-x^2/(4 t))))
```

```
True
```

▲ Task 2.1 Show that this holds true for any order of derivative, including mixed derivatives for 2- or higher dimensional Gaussians.

In 2D and higher dimensions the reasoning is the same. Again we demand the function to be closed to the top. No new structure can emerge.

The requirement for the sign of the second order derivative is now replaced by the requirement on the sign of the rotation invariant *Laplacian*, $\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$.

The reasoning leads to $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial u}{\partial t}$, the 2D linear isotropic diffusion equation, or $\Delta u = \frac{\partial u}{\partial t}$ in any dimension (the Laplacian *operator* $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is often indicated as Δ).

In the following chapters we will study the Gaussian kernel and the Gaussian derivatives in detail. First we present in the next section an alternative and particularly attractive alternative approach to derive the scale-space kernel starting from the maximization of *entropy*.

2.5 Scale-space from entropy maximization

An alternative way to derive the Gaussian kernel as the scale-space kernel of an uncommitted observation is based on the notion that the 'uncommittedness' is expressed in a statistical way using the entropy of the observed signal. The reasoning is due to Mads Nielsen, IT-University Copenhagen [Nielsen1995, Nielsen1997a]:

First of all, we want to do a measurement. We have a *device* which has some integration area over which the measurement is done. As we have seen before, the area (or length or volume) of this detector should have a finite width. It cannot be brought down to zero size, because then nothing would be measured anymore.

The measurement should be done *at all locations* in the same way, with either a series of identical detectors, or the same detector measuring at all places. In mathematical language this is stating that the measurement should be *invariant* for translation.

We also want the measurement to be *linear* in the signal to be measured, for example the intensity. This means that when we measure a signal twice as strong, also the output of the measurement should be doubled, and when we measure two signals, the measurement of the sum of the signals should be equal to the sum of the individual measurements. In mathematical language again this is called invariance for translation along the intensity axis.

These requirements lead automatically to the formulation that the observation must be a *convolution*: $h(x) = \int_{-\infty}^{\infty} L(\alpha)\, g(x-\alpha)\, d\alpha$.

$L(x)$ is the observed variable, in this example the luminance, $g(x)$ is our aperture, $h(x)$ the result of our measurement.

The aperture function $g(x)$ should be a *unity* filter. Such a filter is called a *normalized* filter. Normalization means that the integral over its weighting profile should be unity: $\int_{-\infty}^{\infty} g(x)\, dx = 1$. The filter should not multiply the data with something other than 1.

The *mean* of the filter $g(x)$ should be at the location where we measure (say at x_0), so the expected value (or *first moment*) should be x_0: $\int_{-\infty}^{\infty} x\, g(x)\, dx = x_0$. Because we may take *any* point for x_0, we may take for our further calculations as well the point $x_0 = 0$, which makes life somewhat easier.

The *size* of the aperture is a very essential element. We want to be free in choice of this size, so at least we want to find a *family* of filters where this size is a free parameter. We can then monitor the world at all these sizes by 'looking through' the complete set of kernels simultaneously. We call this 'size' σ. It has the dimension of length, and is the yardstick of our measurement. We call it the inner scale. Every physical measurement has an inner scale. It can be μm, milliseconds, light-years, anything, but for every dimension we need a yardstick. Here σ is our yardstick. We can express distances in a measurement in "number of σ's that we stepped around".

If we weight distances quadratically with our kernel we separate the dimensions: two *orthogonal* vectors fulfill $(a+b)^2 = a^2 + b^2$. Distances (or lengths) add up quadratically by Pythagoras' law. We call the weighted metric σ^2: $\int_{-\infty}^{\infty} x^2\, g(x)\, dx = \sigma^2$.

The last equation we add to the set that will lead to the final formula of the kernel, comes from the incorporation of the request to be as uncommitted as possible. We want no filter that has a preference for something, such as vertical structures, or squares or circles. Actually, we want, in statistical terms, the 'orderlessness' or disorder of the measurement as large as possible, there should be no ordering, ranking, structuring or whatsoever. Physically, this is expressed through the *entropy*, a measure for disorder. The entropy of very regular data is low, we just want maximal entropy. The formula for entropy of our filter is: $H = \int_{-\infty}^{\infty} g(x) \ln g(x)\, dx$ where $\ln(x)$ is the natural logarithm.

We look for the $g(x)$ for which the entropy is maximal, *given the constraints* that we derived before:

$$\int_{-\infty}^{\infty} g(x)\, dx = 1, \quad \int_{-\infty}^{\infty} x\, g(x)\, dx = 0 \text{ and } \int_{-\infty}^{\infty} x^2\, g(x)\, dx = \sigma^2.$$

When we want to find a maximum under a set of given constraints, we apply a standard mathematical technique named the *method of Euler-Lagrange equations* (see for an intuitive explanation of this method Petrou and Bosdogianni [Petrou1999a, page 258]).

This is a technique from the calculus of variations. We first make the Euler-Lagrange equation, or *Lagrangian*, by adding to the entropy term the constraints above, each multiplied with an unknown constant λ, which we are going to determine. The Lagrangian E becomes:

$$E = \int_{-\infty}^{\infty} g(x) \ln g(x)\, dx + \lambda_1 \int_{-\infty}^{\infty} g(x)\, dx + \lambda_2 \int_{-\infty}^{\infty} x\, g(x)\, dx + \lambda_3 \int_{-\infty}^{\infty} x^2\, g(x)\, dx$$

The condition to be minimal for a certain $g(x)$ is given by the vanishing of the first variation (corresponding to the first derivative, but in this case with respect to a function) to $g(x)$: $\frac{\partial E}{\partial g} = 0$. This gives us: $-1 + \lambda_1 + x\lambda_2 + x^2\lambda_3 - \ln g(x) = 0$ from which we can easily solve $g(x)$: $g(x) = e^{-1+\lambda_1+x\lambda_2+x^2\lambda_3}$. So, $g(x)$ is beginning to get some shape: it is an exponential function with constant, linear and quadratic terms of x in the exponent. Let us solve for the λ's:

```
g[x_] := E^(-1+λ1+x λ2+x^2 λ3);
```

From the equation we see that at least λ_3 must be negative, otherwise the function explodes, which is physically unrealistic. We then need the explicit expressions for our constraints, so we make the following set of constraint equations, simplified with the condition of $\lambda_3 < 0$:

```
eqn1 = Simplify[∫_-∞^∞ g[x] dx == 1, λ3 < 0]
```

$$\frac{e^{-1+\lambda 1-\frac{\lambda 2^2}{4\lambda 3}}\sqrt{\pi}}{\sqrt{-\lambda 3}} == 1$$

```
eqn2 = Simplify[∫_-∞^∞ x g[x] dx == 0, λ3 < 0]
```

$$\frac{e^{-1+\lambda 1-\frac{\lambda 2^2}{4\lambda 3}}\sqrt{\pi}\ \lambda 2}{2\ (-\lambda 3)^{3/2}} == 0$$

```
eqn3 = Simplify[∫_-∞^∞ x^2 g[x] dx == σ^2, λ3 < 0]
```

$$\frac{e^{-1+\lambda 1-\frac{\lambda 2^2}{4\lambda 3}}\sqrt{\pi}\ (\lambda 2^2 - 2\lambda 3)}{4\ (-\lambda 3)^{5/2}} == \sigma^2$$

Now we can solve for all three λ's:

```
solution = Solve[{eqn1, eqn2, eqn3}, {λ1, λ2, λ3}]
```

$$\left\{\left\{\lambda 1 \to \frac{1}{4}\,\text{Log}\left[\frac{e^4}{4\pi^2\sigma^4}\right],\ \lambda 2 \to 0,\ \lambda 3 \to -\frac{1}{2\sigma^2}\right\}\right\}$$

```
g[x_, σ_] = Simplify[E^(-1+λ1+x λ2+x^2 λ3) /. Flatten[solution], σ > 0]
```

$$\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi}\ \sigma}$$

which is the Gaussian function. A beautiful result. Again, we have found the Gaussian as the *unique* solution to the set of constraints, which in principle are a formal statement of the *uncommittment* of the observation.

2.6 Derivatives of sampled, observed data

All *partial derivatives* of the Gaussian kernel are solutions too of the diffusion equation.

So the first important result is that we have found the Gaussian kernel and all of its partial derivatives as the *unique* set of kernels for a front-end visual system that satisfies the constraints: no preference for location, scale and orientation, and linearity. We have found a one-parameter *family* of kernels, where the scale σ is the free parameter.

Here are the plots of some members of the Gaussian derivative family:

```
g := 1/(2 π σ^2) Exp[-(x^2 + y^2)/(2 σ^2)]; σ = 1;

Block[{$DisplayFunction = Identity},
  graphs = Plot3D[Evaluate[#], {x, -3.5, 3.5}, {y, -3.5, 3.5}] & /@
    {g, ∂_x g, ∂_x ∂_y g, ∂_{x,x} g + ∂_{y,y} g}];
```

```
Show[GraphicsArray[graphs], ImageSize -> 400];
```

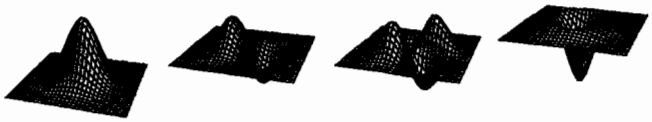


Figure 2.10 Upper left: the Gaussian kernel G(x,y;σ) as the zeroth order point operator; upper right: $\frac{\partial G}{\partial x}$; lower left: $\frac{\partial^2 G}{\partial x \partial y}$; lower right: the Laplacian $\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$ of the Gaussian kernel.

Because of their importance, we will discuss properties of the Gaussian kernel and its derivatives in detail in the next chapters. In chapters 6 and 7 we will see how sensitivity profiles of cells in the retina closely resemble the Laplacian of the Gaussian, and in the primary visual cortex they closely resemble Gaussian derivatives, as was first noticed by Young [Young1985, Young1986, Young1986b, Young1987a] and Koenderink [Koenderink1984a].

The derivative of the observed data $L_0(x, y) \otimes G(x, y; \sigma)$ (the convolution is the observation) is given by $\frac{\partial}{\partial x}\{L_0(x, y) \otimes G(x, y; \sigma)\}$, which can be rewritten as $L_0(x, y) \otimes \frac{\partial}{\partial x} G(x, y; \sigma)$. Note that we cannot apply the chainrule of differentiation here: the operator between $L_0(x, y)$ and $G(x, y; \sigma)$ is a convolution, not a product. The *commutation* (exchange) of the convolution operator and the differential operator is possible because of their linearity. It is best appreciated when we consider the equation $\frac{\partial}{\partial x}\{L_0(x, y) \otimes G(x, y; \sigma)\}$ in the Fourier domain. We need the two rules:

- The Fourier transform of the derivative of a function is equal to $-i\omega$ times the Fourier transform of the function, where $i \equiv \sqrt{-1}$, and
- convolution in the spatial domain is a product in the Fourier domain:

```
Clear[f]; FourierTransform[f[x], x, ω]
```

```
FourierTransform[f[x], x, ω]
```

```
FourierTransform[D[f[x], x], x, ω]
```

```
-i ω FourierTransform[f[x], x, ω]
```

So we get ($\hat{L}$ denotes the Fourier transform of L): $\frac{\partial}{\partial x}\{L_0(x, y) \otimes G(x, y; \sigma)\} \xrightarrow{\mathcal{F}} -i\omega\{\hat{L} . \hat{G}\} = \hat{L} . \{-i\omega \hat{G}\} \xrightarrow{\mathcal{F}^{-1}} L_0(x, y) \otimes \frac{\partial}{\partial x} G(x, y; \sigma)$

The commutation of the convolution and the derivative operators , which is easily shown in the Fourier domain. From this we can see the following important results:

- Differentiation and observation are done in a single step: convolution with a Gaussian derivative kernel.

- Differentiation is now done by *integration*, namely by the convolution integral.

This is a key result in scale-space theory. We can now apply differentiation (even to high order) to *sampled data* like images.

We just convolve the image with the appropriate Gaussian derivative kernel. But where do we need the derivatives, and where do we need higher order derivatives?

An important area of application is the exploitation of *geometric* information from images. The most basic example is the first order derivative, which gives us *edges*.

Edges are defined as a sudden change of intensity L when we walk over the image and this is exactly what a derivative captures: $\frac{\partial L}{\partial x}$.

Derivatives abound in the detection of *differential features* (features expressed as some (polynomial) expression in image derivatives). They also show up with the detection of motion, of stereo disparity to find the depth, the detection of structure in color images, segmentation, image enhancement and denoising, and many other application areas as we will see in the rest of the book.

Some more implications of the theory so far:

- The Gaussian kernel is the *physical* analogue of a *mathematical* point, the Gaussian derivative kernels are the physical analogons of the mathematical differential operators. Equivalence is reached for the limit when the scale of the Gaussian goes to zero:

- $\lim_{\sigma \to 0} G(x;\sigma) = \delta(x)$, where $\delta(x)$ is the Dirac delta function, and $\lim_{\sigma \to 0} \left\{ f(x) \otimes \frac{\partial G(x;\sigma)}{\partial x} \right\} = \lim_{\sigma \to 0} \int_{-\infty}^{\infty} f(\alpha)\ \partial_x G(x-\alpha;\sigma)\, d\alpha = \int_{-\infty}^{\infty} f(\alpha)\ \delta(x-\alpha)\, d\alpha = \partial_x f(x)$.

```
∫_-∞^∞ f[α] D[DiracDelta[α - x], x] dα

f'[x]
```

- There is an intrinsic and unavoidable relation between differentiation and blurring. By its definition, *any* differentiation on discrete (observed) data blurs the data somewhat, with the amount of the scale of the differential operator. There is no way out, this increase of the inner scale is a physical necessity. We can only try to minimize the effect by choosing small scales for the differentiation operator. However, this minimal scale is subject to constraints (as is the maximal scale). In chapter 7 we develop the fundamental relation between the scale of the operator, its differential order and the required amount of accuracy.

The *Mathematica* function `gD[im,nx,ny,σ]` implements a convolution with a Gaussian derivative on the image `im`, with order of differentiation `nx` with respect to x resp. `ny` with respect to y. Figure 2.12 shows the derivative to x and y of a simple test image of a square:

```
im = Table[If[80 < x < 170 && 80 < y < 170, 1, 0], {y, 1, 256}, {x, 1, 256}];
Block[{$DisplayFunction = Identity},
  imx = gD[im, 1, 0, 1]; imy = gD[im, 0, 1, 1]; grad = Sqrt[imx^2 + imy^2];
  p1 = ListDensityPlot /@ {im, imx, imy, grad}];
Show[GraphicsArray[p1], ImageSize → 400];
```

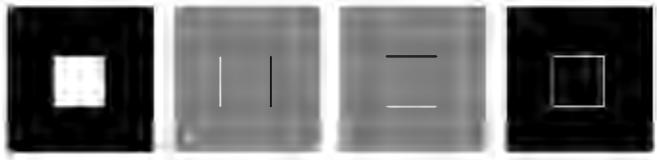


Figure 2.11 The first order derivative of an image gives edges. Left: original test image $L(x, y)$, resolution 256^2. Second: the derivative with respect to x: $\frac{\partial L}{\partial x}$ at scale $\sigma = 1$ pixel. Note the positive and negative edges. Third: the derivative with respect to y: $\frac{\partial L}{\partial y}$ at scale $\sigma = 1$ pixel. Right: the gradient $\sqrt{(\frac{\partial L}{\partial x})^2 + (\frac{\partial L}{\partial x})^2}$ at a scale of $\sigma = 1$ pixel which gives all edges.

- The Gaussian kernel is the unique kernel that generates no *spurious resolution*. It is the blown-up physical *point operator*, the Gaussian derivatives are the blown-up physical *multi-scale derivative operators*.

```
Show[Import["blown-up ddx.jpg"], ImageSize -> 300];
```

Figure 2.12 Convolution with a Gaussian derivative is the blown-up version of convolution with the Delta Dirac function. Taking the limit of the scale to zero ('letting the air out") leads to the 'regular' mathematical formulation.

- Because convolving is an integration, the Gaussian kernel has by definition a strong *regularizing* effect. It was shown by Schwartz [Schwartz1951] that differentiation of *distributions* of data ('wild' data, such as discontinuous or sampled data) has to be accomplished by convolution with a smooth testfunction. This smooth testfunction is our Gaussian kernel here. So, we recognize that the process of observation *is* the regularizer. So there is no need to smooth the data first. Actually, one should never change the input data, but only make modifications to the process of observation where one has access: the filter through which the measurement is done. The visual system does the same: it employs filters at many sizes and shapes, as we will see in the chapter on human vision.

- Recently some interesting papers have shown the complete equivalence of Gaussian scale space regularization with a number of other methods for regularization such as splines,

thin plate splines, graduated convexity etc. [Scherzer2000a, Nielsen1996b, Nielsen1997b]. In chapter 10 we will discuss the aspects of differentiation of discrete data (it is 'ill-posed') and the property of regularization in detail.

- The set of Gaussian derivative kernels (including the zeroth order derivative: the Gaussian kernel itself) forms a *complete* set of derivatives. This set is sometimes referred to as the *N-jet.*

Now the basic toolkit is there to do differential geometry, tensor analysis, invariant theory, topology and apply many more mathematical tools on our discrete data. This will be the topic of much of the rest of this book.

2.7 Scale-space stack

A scale-space is a stack of 2D images, where scale is the third dimension. One can make a scale-space of any measurement, so one can measure an intensity scale-space, a gradient magnitude scale-space, a Laplacian scale-space etc.

```
im = Import["mr64.gif"][[1, 1]];
Block[{$DisplayFunction = Identity, xres, yres, max},
 {yres, xres} = Dimensions[im]; max = Max[im];
 gr = Graphics3D[ListPlot3D[Table[0, {yres}, {xres}],
    Map[GrayLevel, im / max, {2}], Mesh → False, Boxed → False]];
gb = Table[blur = gD[im, 0, 0, i]; Graphics3D[ListPlot3D[
     Table[i 10, {yres}, {xres}], Map[GrayLevel, blur / max, {2}],
     Mesh → False, Boxed → False]], {i, 1, 6}]];
Show[{gr, gb}, BoxRatios → {1, 1, 1}, ViewPoint -> {1.190, -3.209, 1.234},
  DisplayFunction -> $DisplayFunction, Boxed -> True, ImageSize -> 240];
```

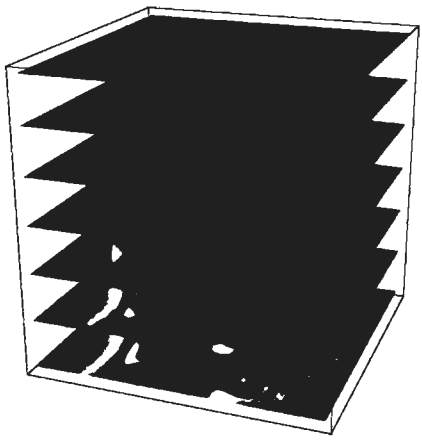


Figure 2.13 A scale-space of a 2D MRI sagittal slice, dimensions 64^2, for a range of scales σ =1,2,3,4,5,and 6 pixels.

We found a family of kernels, with the scale σ as a free parameter. When we don't know what scale to apply in an uncommitted measurement, we just take them all. It is like sampling at spatial locations: we put CCD elements all over our receptor's sensitive area. We

will see that the visual system does just that: it has *groups* of rods and cones in the retina (termed receptive fields) of a wide range of circular diameters, effectively sampling at many different scales.

We will see in the chapters on the 'deep structure' of images (i.e. the structure along the scale axis), that in the scale-space the *hierarchical, topological* structure of images is embedded. See chapters 13-15.

One can make scale-spaces of any dimension. A scale-space stack of 3D images, such as 3D datasets from medical tomographic scanners, is a 4D space (x,y,z;σ) and is termed a *hyperstack* [Vincken 1990].

And here are two scale-spaces of a real image, a scale-space of the intensity (no derivatives, only blurred) and a scale-space of the Laplacian (the Laplacian is the sum of the second order derivatives of the image, $\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$).

```
im = Import["mr128.gif"][[1, 1]];
DisplayTogetherArray[
  {Table[ListDensityPlot[gD[im, 0, 0, E^τ]], {τ, 0, 2.1, .3}],
   Table[ListDensityPlot[gD[im, 2, 0, E^τ] + gD[im, 0, 2, E^τ]],
    {τ, 0, 2.1, .3}]}, ImageSize -> 390];
```

Figure 2.14 A scale-space is a stack of images at a range of scales. Top row: Gaussian blur scale-space of a sagittal Magnetic Resonance image, resolution 128^2, exponential scale range from $\sigma = e^0$ to $\sigma = e^{2.1}$. Bottom row: Laplacian scale-space of the same image, same scale range.
The function `gD[im,nx,ny,σ]` will be explained later (chapter 4 and 5). It convolves the image with a Gaussian derivative.

2.8 Sampling the scale-axis

From the example from the trip through scale in the "Powers of 10" series we made steps of a *factor* 10 each time we took a new picture. This is an exponential stepping through scale, and we know this as experimental fact. We step in 'orders of magnitude'. The scale parameter σ gives a logical length parameter for the level of resolution.

If we consider how to parametrize scale σ with a dimensionless parameter τ, then we realize that *scale-invariance* (or *self-similarity*) must imply that $d\sigma / d\tau$ must be proportional to σ.

In other words, the change that we see when we step along the scale axis, is proportional to the level of resolution at hand. Without loss of generality we may take $\frac{d\sigma}{d\tau} = \sigma$ with $\sigma|_{\tau=0} = \epsilon$. We call the dimensionless parameter τ the *natural scale parameter*: $\sigma = \epsilon e^{\tau}$

where τ can be any number, even negative. Note that the artificial singularity due to the problematic value of $\sigma = 0$ is now no longer present.

There is a difference between 'zooming' and 'blurring':

zooming is the reparametrization of the spatial axis, $\tilde{x} \mapsto a\,x$, so we get a larger or smaller image by just setting them further apart of farther away. There is no information gained or lost. Blurring is doing an observation with a larger aperture: the image is blurred. Now information *is* lost, and this is exactly what is a requirement for a scale-space: reduction of information. Because we have a larger σ over the same image domain, we can effectively perform a *sampling rate reduction* [Vincken1990].

How much information is lost when we increase scale? Florack [Florack1994b] introduced the following reasoning:

The number of (equidistant) samples on a given domain, given a fixed amount of overlap between neighboring apertures, on scale level σ relative to the number of samples at another scale level σ_0 is given by $\frac{N(\sigma)}{N(\sigma_0)} = (\frac{\sigma_0}{\sigma})^D$, where D is the dimension.
Or, in terms of the natural scale parameter τ with $\sigma = \epsilon\, e^{\tau}$:

$$N(\sigma) = N(\sigma_0)\,(\frac{\epsilon e^{\tau_0}}{\epsilon e^{\tau}})^D = N(\sigma_0)\,e^{D(\tau_0 - \tau)}$$

which is the solution of the differential equation $\frac{dN}{d\tau} + DN = 0$. At the highest scale, we have just a single wide aperture left and we achieved total blurring; the image domain has become a single point. Notice that the sampling rate reduction depends on the dimension D. When we consider natural, generic images, we expect the information in the images to exist on all scales. We could think of a 'density of local generic features' such as intensity maxima, minima, saddle points, corners etc. as relatively homogeneously distributed over the images over all scales when we consider enough images. This 'feature density' $N_F(\tau)$ might then be related to the number of samples $N(\tau)$, so $\frac{dN_F}{d\tau} + DN_F = 0$. In chapter 20 we will see that the number of extrema and saddle points in a scale-space of generic images indeed decreases with a slope of $\frac{d \ln N_F}{d\tau} \approx -2$ for 2D images and a slope of -1 for 1D signals.

The factor ϵ in the equation for natural scale appears for dimensional reasons: it is the scale for $\tau = 0$, and is a property of our imaging device; it is the pixelsize, CCD element size, the sampling width etc.: the *inner scale* of the measurement.

```
Block[{$DisplayFunction = Identity},
  p1 = Graphics[
    Table[Circle[{x, y}, .6], {x, 1, 10}, {y, 1, 10}], AspectRatio -> 1];
p2 = Graphics[Table[Circle[{x, y}, 1.2], {x, 1, 10, 2}, {y, 1, 10, 2}],
    AspectRatio -> 1];
p3 = Graphics3D[Table[{EdgeForm[], TranslateShape[Sphere[.6, 10, 10],
       {x, y, z}]}, {x, 1, 6}, {y, 1, 6}, {z, 1, 6}], Boxed -> False];
p4 = Graphics3D[Table[{EdgeForm[], TranslateShape[
       Sphere[1.2, 10, 10], {x, y, z}]}, {x, 1, 6, 2},
     {y, 1, 6, 2}, {z, 1, 6, 2}], Boxed -> False]];
Show[GraphicsArray[{p1, p2, p3, p4}], ImageSize -> 400];
```

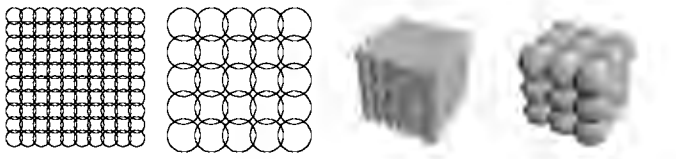


Figure 2.15 The number of samples on a 2D domain, given a fixed amount of overlap decreases with $(\frac{\sigma_0}{\sigma})^2$ (left two figures), on a 3D domain with $(\frac{\sigma_0}{\sigma})^3$ (right two figures). So the number of samples decreases as a function of scale with a slope of $-D$, where D is the dimension (see text). The sampling rate reduction is dependent on the dimensionality of the measurement.

For positive τ we go to larger scale, for negative τ we go to smaller scale. In the expression for the natural scale the singularity at $\sigma = 0$ is effectively removed.

The exponential stepping over the scale axis is also evident in the *Hausdorff dimension*, the number of boxes counted in a quadtree of a binary image (see also [Pedersen2000] and chapter 15, section 15.1.4).

Of course, there is no information within the inner scale, so here problems are to be expected when we try to extract information at sub-pixel scale. Only by taking into account a *context* of voxels through a proper model, we can go to the subpixel domain.

This is an important notion: any observation at a single point is an independent measurement, and we can do a lot of measurements there.

In the next few chapters we will derive many features related to the measurement of derivatives at our pixel. It turns out that we can make lots of specific polynomial combinations, like edge strength, 'cornerness' etc. but they all describe information in that point. It is a 'keyhole observation'. The important 'perceptual grouping' of neighboring points into meaningful sets is accomplished by specifying constraints, like *models*. In this book we first derive many local (differential) features.

In the second part we go a little further in the cascade of visual processing steps, and investigate local neighborhood relations through comparison of local properties like orientation, strength of derivative measurements etc. We also explore the *deep structure* of images (a term first coined by Koenderink), by which we mean the relations over scale. In

the deep structure we may expect the hierarchical, structuring, more topological information: what is 'embedded in' what, what is 'surrounded by' what, what is 'part of' what etc. This takes us to a next level of description in images, which is currently receiving a lot of attention.

Fractals are famous examples of self similar functions. This self-similar fractal shows a tree in three dimensions [Cabrera, www.mathsource.com]. Parameters: α = branch angle; ϵ = scale factor; m = number of branches from previous branch; n = deepness.

2.9 Summary of this chapter

Scale-space theory was discovered independently by Iijima in Japan in the early sixties, and by Koenderink in Europe in the early seventies.

Because we have specific physical constraints for the early vision front-end kernel, we are able to set up a 'first principle' framework from which the exact sensitivity function of the measurement aperture can be derived. There exist many such derivations for an uncommitted kernel, all leading to the same unique result: the Gaussian kernel. We discussed two approaches: the first started with the assumptions of linearity, isotropy, homogeneity and scale-invariance.

With the help of the Pi-theorem from dimensional analysis one is able to derive the Gaussian by plugging in the constraints one by one.

The second derivation started from causality: it is impossible that maxima increase and minima decrease with increasing scale, every blurred version is the causal consequence of the image it was blurred from. This means that the extrema must be closed from above. This leads to s constraint on the sign of the second derivative, from which the diffusion equation emerges.

The third derivation started from the minimization of the entropy at the very first measurement. Through the use of Lagrange multipliers, where the constraints are used one by one, one can again derive the Gaussian kernel as the unique kernel for the front-end.

A crucial result is that differentiation of discrete data is done by the convolution with the derivative of the observation kernel, in other words: by an integration. Differentiation is now possible on discrete data by means of convolution with a finite kernel. In chapter 14 we discuss this important mathematical notion, which is known as *regularization*.

This means that differentiation can never be done without blurring the data somewhat. We find as a *complete* family of front-end kernels the family of all partial derivatives of the Gaussian kernel. The zeroth order derivative is just the Gaussian blurkernel itself.

Scale is parametrized in an exponential fashion (we consider 'orders of magnitude' when scaling). The exponent in this parametrization is called the natural scale parameter.

```
Rotz[t_] = {{Cos[t], Sin[t], 0}, {-Sin[t], Cos[t], 0}, {0, 0, 1}};
Roty[t_] = {{Cos[t], 0, -Sin[t]}, {0, 1, 0}, {Sin[t], 0, Cos[t]}};
Rot3D[ψ_, θ_] = Roty[θ].Rotz[ψ]; SphericalCoordinates[{x_, y_, z_}] =
 { Sqrt[x^2 + y^2 + z^2], ArcTan[z, Sqrt[x^2 + z^2]] , ArcTan[x, y]};
NextBranches[α_, ϵ_, m_][ Branch[rl_List, r0_List, th_] ] :=
    Module[{r, ψ, θ}, {r, θ, ψ} = SphericalCoordinates[rl - r0];
        {Branch[ϵ * (rl - r0) + rl, rl, ϵ*th], Sequence@@ Table[
Branch[rl + ϵ*r {Sin[α] Cos[φ], Sin[α] Sin[φ], Cos[α]}.Rot3D[ψ, θ],
       rl, ϵ*th], {φ, 0, 2 Pi, 2 Pi/m}]}] // N];
NextBranches[α_, ϵ_, m_][w_List] := Map[NextBranches[α, ϵ, m], w];
Tree2D[α_, ϵ_, m_, r_List, th_, n_] :=
NestList[NextBranches[α, ϵ, m], Branch[r, {0, 0, 0}, 1] , n] /.
  Branch[rl_, r0_, t_] :>
   {RGBColor[0, 0.6 (1 - t) + 0.4, 0], Thickness[th*t], Line[{rl, r0}]}
Show[Graphics3D[Tree2D[α, ϵ, m, r, th0, n] /.
    {α -> Pi / 8, ϵ -> 0.6, m -> 5, r -> {0.01, 0, 1}, n -> 4, th0 -> 0.03}],
  PlotRange -> {{-1, 1}, {-1, 1}, {0, 2.5}},
  ViewPoint -> {3.369, -0.040, 0.312}, ImageSize -> 200];
```

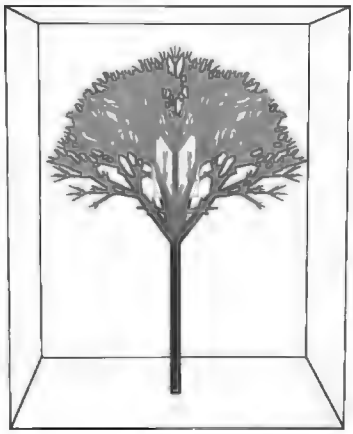


Figure 2.16 Fractals are famous examples of self similar functions. This self-similar fractal shows a tree in three dimensions [Cabrera, www.mathsource.com]. Parameters: α = branch angle; ϵ = scale factor; m = number of branches from previous branch; n = deepness. Source: Renan Cabrera, www.mathsource.com.

3. The Gaussian kernel

Of all things, man is the measure.
Protagoras the Sophist (480-411 B.C.)

3.1 The Gaussian kernel

The Gaussian (better Gaußian) kernel is named after Carl Friedrich Gauß (1777-1855), a brilliant German mathematician. This chapter discusses many of the attractive and special properties of the Gaussian kernel.

```
<< FrontEndVision`FEV`; Show[Import["Gauss10DM.gif"], ImageSize -> 280];
```

Figure 3.1 The Gaussian kernel is apparent on every German banknote of DM 10,- where it is depicted next to its famous inventor when he was 55 years old. The new Euro replaces these banknotes. See also: http://scienceworld.wolfram.com/biography/Gauss.html.

The Gaussian kernel is defined in 1-D, 2D and N-D respectively as

$$G_{1D}(x;\sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}},\quad G_{2D}(x,y;\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},\quad G_{ND}(\vec{x};\sigma) = \frac{1}{(\sqrt{2\pi}\,\sigma)^N} e^{-\frac{|\vec{x}|^2}{2\sigma^2}}$$

The σ determines the *width* of the Gaussian kernel. In statistics, when we consider the Gaussian probability density function it is called the *standard deviation*, and the square of it, σ^2, the *variance*. In the rest of this book, when we consider the Gaussian as an aperture function of some observation, we will refer to σ as the *inner scale* or shortly *scale*.

In the whole of this book the scale can only take positive values, $\sigma > 0$. In the process of observation σ can never become zero. For, this would imply making an observation through an infinitesimally small aperture, which is impossible. The factor of 2 in the exponent is a matter of convention, because we then have a 'cleaner' formula for the diffusion equation, as we will see later on. The semicolon between the spatial and scale parameters is conventionally put there to make the difference between these parameters explicit.

The scale-dimension is *not* just another spatial dimension, as we will thoroughly discuss in the remainder of this book.

The *half width at half maximum* ($\sigma = 2\sqrt{2 \ln 2}$) is often used to approximate σ, but it is somewhat larger:

```
Unprotect[gauss];
gauss[x_, σ_] := 1/(σ Sqrt[2 π]) Exp[-x^2/(2 σ^2)];
Solve[gauss[x, σ]/gauss[0, σ] == 1/2, x]
{{x → -σ Sqrt[2 Log[2]]}, {x → σ Sqrt[2 Log[2]]}}
% // N
{{x → -1.17741 σ}, {x → 1.17741 σ}}
```

3.2 Normalization

The term $\frac{1}{\sqrt{2\pi}\,\sigma}$ in front of the one-dimensional Gaussian kernel is the normalization constant. It comes from the fact that the integral over the exponential function is not unity: $\int_{-\infty}^{\infty} e^{-x^2/2\sigma^2}\,dx = \sqrt{2\pi}\,\sigma$. With the normalization constant this Gaussian kernel is a *normalized* kernel, i.e. its integral over its full domain is unity for every σ.

This means that increasing the σ of the kernel reduces the amplitude substantially. Let us look at the graphs of the normalized kernels for $\sigma = 0.3$, $\sigma = 1$ and $\sigma = 2$ plotted on the same axes:

```
Unprotect[gauss]; gauss[x_, σ_] := 1/(σ Sqrt[2 π]) Exp[-x^2/(2 σ^2)];
Block[{$DisplayFunction = Identity}, {p1, p2, p3} =
   Plot[gauss[x, σ = #], {x, -5, 5}, PlotRange -> {0, 1.4}] & /@
    {.3, 1, 2}];
Show[GraphicsArray[{p1, p2, p3}], ImageSize -> 400];
```

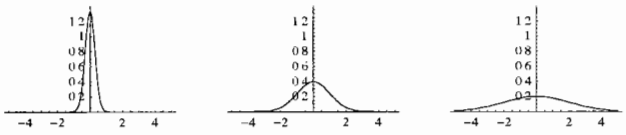


Figure 3.2 The Gaussian function at scales $\sigma = .3$, $\sigma = 1$ and $\sigma = 2$. The kernel is normalized, so the total area under the curve is always unity.

The normalization ensures that the average graylevel of the image remains the same when we blur the image with this kernel. This is known as *average grey level invariance*.

3.3 Cascade property, selfsimilarity

The shape of the kernel remains the same, irrespective of the σ. When we *convolve* two Gaussian kernels we get a new wider Gaussian with a variance σ^2 which is the sum of the variances of the constituting Gaussians: $g_{new}(\vec{x}; \sigma_1^2 + \sigma_2^2) = g_1(\vec{x}; \sigma_1^2) \otimes g_2(\vec{x}; \sigma_2^2)$.

```
σ=.; Simplify[∫_-∞^∞ gauss[α, σ1] gauss[α - x, σ2] dα, {σ1 > 0, σ2 > 0}]
```

$$\frac{e^{-\frac{x^2}{2(\sigma_1^2+\sigma_2^2)}}}{\sqrt{2\pi}\sqrt{\sigma_1^2+\sigma_2^2}}$$

This phenomenon, i.e. that a new function emerges that is similar to the constituting functions, is called *self-similarity*.

The Gaussian is a *self-similar function*. Convolution with a Gaussian is a linear operation, so a convolution with a Gaussian kernel followed by a convolution with again a Gaussian kernel is equivalent to convolution with the broader kernel. Note that the *squares* of σ add, not the σ's themselves. Of course we can concatenate as many blurring steps as we want to create a larger blurring step. With analogy to a cascade of waterfalls spanning the same height as the total waterfall, this phenomenon is also known as the *cascade smoothing property*.
Famous examples of self-similar functions are *fractals*. This shows the famous Mandelbrot fractal:

```
cMandelbrot = Compile[{{c, _Complex}}, -Length[
    FixedPointList[#^2 + c &, c, 50, SameTest -> (Abs[#2] > 2.0 &)]]];
ListDensityPlot[-Table[cMandelbrot[a + b I], {b, -1.1, 1.1, 0.0114},
    {a, -2.0, 0.5, 0.0142}], Mesh -> False, AspectRatio -> Automatic,
  Frame -> False, ColorFunction -> Hue, ImageSize -> 170];
```

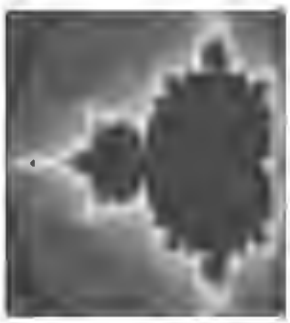


Figure 3.3 The Mandelbrot fractal is a famous example of a self-similar function. Source: www.mathforum.org. See also mathworld.wolfram.com/MandelbrotSet.html.

3.4 The scale parameter

In order to avoid the summing of squares, one often uses the following parametrization: $2\sigma^2 \rightarrow t$, so the Gaussian kernel get a particular short form. In N dimensions: $G_{\mathrm{ND}}(\vec{x}, t) = \frac{1}{(\pi t)^{N/2}} e^{-\frac{r^2}{t}}$.

It is this t that emerges in the diffusion equation $\frac{\partial L}{\partial t} = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2} + \frac{\partial^2 L}{\partial z^2}$. It is often referred to as 'scale' (like in: differentiation to scale, $\frac{\partial L}{\partial t}$), but a better name is *variance*.

To make the self-similarity of the Gaussian kernel explicit, we can introduce a new *dimensionless* spatial parameter, $\tilde{x} = \frac{x}{\sigma\sqrt{2}}$. We say that we have *reparametrized* the x-axis. Now the Gaussian kernel becomes: $g_n(\tilde{x}; \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\tilde{x}^2}$, or $g_n(\tilde{x}; t) = \frac{1}{(\pi t)^{N/2}} e^{-\tilde{x}^2}$. In other words: if we walk along the spatial axis in footsteps expressed in scale-units (σ's), all kernels are of equal size or 'width' (but due to the normalization constraint not necessarily of the same amplitude). We now have a 'natural' size of footstep to walk over the spatial coordinate: a unit step in x is now $\sigma\sqrt{2}$, so in more blurred images we make bigger steps. We call this basic Gaussian kernel the *natural* Gaussian kernel $g_n(\tilde{x}; \sigma)$. The new coordinate $\tilde{x} = \frac{x}{\sigma\sqrt{2}}$ is called the *natural coordinate*. It eliminates the scale factor σ from the spatial coordinates, i.e. it makes the Gaussian kernels similar, despite their different inner scales. We will encounter natural coordinates many times hereafter.

The spatial extent of the Gaussian kernel ranges from -∞ to +∞, but in practice it has negligible values for x larger then a few (say 5) σ. The numerical value at x=5σ, and the area under the curve from x=5σ to infinity (recall that the total area is 1):

```
gauss[5, 1] // N
Integrate[gauss[x, 1], {x, 5, Infinity}] // N
```

```
1.48672 × 10^-6
```

```
2.86652 × 10^-7
```

The larger we make the standard deviation σ, the more the image gets blurred. In the limit to infinity, the image becomes homogenous in intensity. The final intensity is the average intensity of the image. This is true for an image with infinite extent, which in practice will never occur, of course. The boundary has to be taken into account. Actually, one can take many choices what to do at the boundary, it is a matter of consensus. Boundaries are discussed in detail in chapter 5, where practical issues of computer implementation are discussed.

3.5 Relation to generalized functions

The Gaussian kernel is the physical equivalent of the *mathematical point*. It is not strictly local, like the mathematical point, but *semi-local*. It has a *Gaussian weighted extent*, indicated by its inner scale σ.

Because scale-space theory is revolving around the Gaussian function and its derivatives as a physical differential operator (in more detail explained in the next chapter), we will focus here on some mathematical notions that are directly related, i.e. the mathematical notions underlying sampling of values from functions and their derivatives at *selected* points (i.e. that is why it is referred to as sampling). The mathematical functions involved are the *generalized functions*, i.e. the Delta-Dirac function, the Heaviside function and the error function. In the next section we study these functions in detail.

When we take the limit as the inner scale goes down to zero (remember that σ can only take positive values for a physically realistic system), we get the mathematical delta function, or Dirac delta function, $\delta(x)$. This function is everywhere zero except in $x = 0$, where it has infinite amplitude and zero width, its area is unity.

$$\lim_{\sigma \downarrow 0} \left(\frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{x^2}{2\sigma^2}} \right) = \delta(x).$$

$\delta(x)$ is called the *sampling function* in mathematics, because the Dirac delta function adequately samples just one point out of a function when integrated. It is assumed that $f(x)$ is continuous at $x = a$:

$$\int_{-\infty}^{\infty} \texttt{DiracDelta[x - a] f[x] dx}$$

$$\texttt{f[a]}$$

The *sampling property of derivatives* of the Dirac delta function is shown below:

$$\int_{-\infty}^{\infty} \texttt{D[DiracDelta[x], \{x, 2\}] f[x] dx}$$

$$\texttt{f''[0]}$$

The delta function was originally proposed by the eccentric Victorian mathematician Oliver Heaviside (1880-1925, see also [Pickover1998]). Story goes that mathematicians called this function a "monstrosity", but it did work! Around 1950 physicist Paul Dirac (1902-1984) gave it new light. Mathematician Laurent Schwartz (1915-) proved it in 1951 with his famous "theory of distributions" (we discuss this theory in chapter 8). And today it's called "the Dirac delta function".

The integral of the Gaussian kernel from -∞ to x is a famous function as well. It is the *error function*, or *cumulative* Gaussian function, and is defined as:

$$\sigma = .;\ \texttt{err[x\_, σ\_]} = \int_0^x \frac{1}{\sigma\sqrt{2\pi}} \texttt{Exp}\left[-\frac{y^2}{2\sigma^2}\right] \texttt{dy}$$

$$\frac{1}{2}\,\texttt{Erf}\left[\frac{x}{\sqrt{2}\,\sigma}\right]$$

The y in the integral above is just a dummy integration variable, and is integrated out. The *Mathematica* error function is **Erf[x]**.

In our integral of the Gaussian function we need to do the reparametrization $x \to \frac{x}{\sigma\sqrt{2}}$. Again we recognize the natural coordinates. The factor $\frac{1}{2}$ is due to the fact that integration starts halfway, in x = 0.

```
σ = 1.; Plot[1/2 Erf[x/(σ Sqrt[2])], {x, -4, 4}, AspectRatio -> .3,
 AxesLabel -> {"x", "Erf[x]"}, ImageSize -> 200];
```

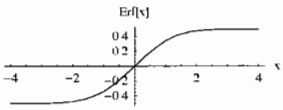


Figure 3.4 The error function **Erf[x]** is the cumulative Gaussian function.

When the inner scale σ of the error function goes to zero, we get in the limiting case the so-called *Heavyside function* or *unitstep function*. The derivative of the Heavyside function is the Delta-Dirac function, just as the derivative of the error function of the Gaussian kernel.

```
σ = .1; Plot[1/2 Erf[x/(σ Sqrt[2])], {x, -4, 4}, AspectRatio -> .3,
 AxesLabel -> {"x", "Erf[x]"}, ImageSize -> 270];
```

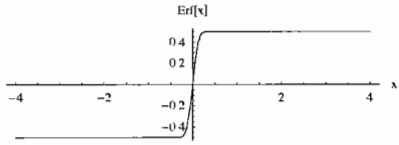


Figure 3.5 For decreasing σ the Error function begins to look like a step function. The Error function is the Gaussian blurred step-edge.

```
Plot[UnitStep[x], {x, -4, 4}, DisplayFunction -> $DisplayFunction,
  AspectRatio -> .3, AxesLabel -> {"x", "Heavyside[x], UnitStep[x]"},
  PlotStyle -> Thickness[.015], ImageSize -> 270];
```

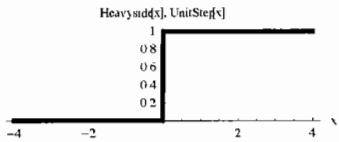


Figure 3.6 The Heavyside function is the generalized unit stepfunction. It is the limiting case of the Error function for $\lim \sigma \to 0$.

The derivative of the Heavyside step function is the Delta function again:

```
D[UnitStep[x], x]

DiracDelta[x]
```

3.6 Separability

The Gaussian kernel for dimensions higher than one, say N, can be described as a regular product of N one-dimensional kernels. Example: $g_{2D}(x, y; \sigma_1^2 + \sigma_2^2) = g_{1D}(x; \sigma_1^2)\, g_{1D}(y; \sigma_2^2)$ where the space in between is the product operator. The regular product also explains the exponent N in the normalization constant for N-dimensional Gaussian kernels in (0). Because higher dimensional Gaussian kernels are regular products of one-dimensional Gaussians, they are called *separable*. We will use quite often this property of *separability*.

```
DisplayTogetherArray[{Plot[gauss[x, σ = 1], {x, -3, 3}],
   Plot3D[gauss[x, σ = 1] gauss[y, σ = 1], {x, -3, 3}, {y, -3, 3}]},
  ImageSize -> 440];
```

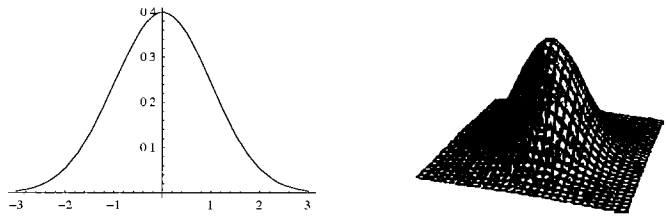


Figure 3.7 A product of Gaussian functions gives a higher dimensional Gaussian function. This is a consequence of the separability.

An important application is the speed improvement when implementing numerical *separable convolution*. In chapter 5 we explain in detail how the convolution with a 2D (or better: N-dimensional) Gaussian kernel can be replaced by a cascade of 1D convolutions, making the process much more efficient because convolution with the 1D kernels requires far fewer multiplications.

3.7 Relation to binomial coefficients

Another place where the Gaussian function emerges is in expansions of powers of polynomials. Here is an example:

```
Expand[(x + y)^30]
```

$$x^{30} + 30\,x^{29}\,y + 435\,x^{28}\,y^2 + 4060\,x^{27}\,y^3 + 27405\,x^{26}\,y^4 + 142506\,x^{25}\,y^5 + 593775\,x^{24}\,y^6 + 2035800\,x^{23}\,y^7 + 5852925\,x^{22}\,y^8 + 14307150\,x^{21}\,y^9 + 30045015\,x^{20}\,y^{10} + 54627300\,x^{19}\,y^{11} + 86493225\,x^{18}\,y^{12} + 119759850\,x^{17}\,y^{13} + 145422675\,x^{16}\,y^{14} + 155117520\,x^{15}\,y^{15} + 145422675\,x^{14}\,y^{16} + 119759850\,x^{13}\,y^{17} + 86493225\,x^{12}\,y^{18} + 54627300\,x^{11}\,y^{19} + 30045015\,x^{10}\,y^{20} + 14307150\,x^{9}\,y^{21} + 5852925\,x^{8}\,y^{22} + 2035800\,x^{7}\,y^{23} + 593775\,x^{6}\,y^{24} + 142506\,x^{5}\,y^{25} + 27405\,x^{4}\,y^{26} + 4060\,x^{3}\,y^{27} + 435\,x^{2}\,y^{28} + 30\,x\,y^{29} + y^{30}$$

The coefficients of this expansion are the *binomial coefficients* $\binom{n}{m}$ ('n over m'):

```
ListPlot[Table[Binomial[30, n], {n, 1, 30}],
  PlotStyle -> {PointSize[.015]}, AspectRatio -> .3];
```

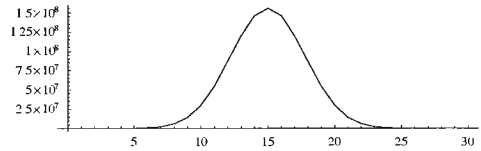


Figure 3.8 Binomial coefficients approximate a Gaussian distribution for increasing order.

And here in two dimensions:

```
BarChart3D[Table[Binomial[30, n] Binomial[30, m], {n, 1, 30}, {m, 1, 30}],
  ImageSize -> 180];
```

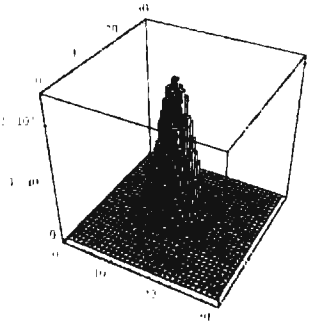


Figure 3.9 Binomial coefficients approximate a Gaussian distribution for increasing order. Here in 2 dimensions we see separability again.

3.8 The Fourier transform of the Gaussian kernel

We will regularly do our calculations in the Fourier domain, as this often turns out to be analytically convenient or computationally efficient. The basis functions of the Fourier transform $\mathcal{F}$ are the sinusoidal functions $e^{i\omega x}$. The definitions for the Fourier transform and its inverse are:

the Fourier transform: $F(\omega) = \mathcal{F}\{f(x)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\, e^{i\omega x}\, dx$

the inverse Fourier transform: $\mathcal{F}^{-1}\{F(\omega)\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(\omega)\, e^{-i\omega x}\, d\omega$

```
σ =.; Fgauss[ω_, σ_] =
 Simplify[1/Sqrt[2 π] Integrate[1/(σ Sqrt[2 π]) Exp[-x^2/(2 σ^2)] Exp[I ω x], {x, -∞, ∞}],
  {σ > 0, Im[σ] == 0}]
```

$$\frac{e^{-\frac{1}{2}\sigma^2\omega^2}}{\sqrt{2\pi}}$$

The Fourier transform is a standard *Mathematica* command:

```
Simplify[FourierTransform[gauss[x, σ], x, ω], σ > 0]
```

$$\frac{e^{-\frac{1}{2}\sigma^2\omega^2}}{\sqrt{2\pi}}$$

Note that different communities (mathematicians, computer scientists, engineers) have different definitions for the Fourier transform. From the *Mathematica* help function:

With the setting **FourierParameters→{a,b}** the discrete Fourier transform computed by **FourierTransform** is $\sqrt{\frac{|b|}{(2\pi)^{1-a}}}\int_{-\infty}^{\infty} f(t)\,e^{ibwt}\,dt$. Some common choices for **{a,b}** are **{0,1}** (default), **{-1,1}** (data analysis), **{1,-1}** (signal processing).

In this book we consistently use the default definition.

So the Fourier transform of the Gaussian function is again a Gaussian function, but now of the frequency ω. The Gaussian function is the *only* function with this property. Note that the scale σ now appears as a multiplication with the frequency. We recognize a well-known fact: a smaller kernel in the spatial domain gives a wider kernel in the Fourier domain, and vice versa. Here we plot 3 Gaussian kernels with their Fourier transform beneath each plot:

```
Block[{$DisplayFunction = Identity},
  p1 = Table[Plot[gauss[x, σ], {x, -10, 10}, PlotRange -> All,
     PlotLabel -> "gauss[x," <> ToString[σ] <> "]"], {σ, 1, 3}];
  p2 = Table[Plot[Fgauss[ω, σ], {ω, -3, 3}, PlotRange -> All,
     PlotLabel -> "Fgauss[x," <> ToString[σ] <> "]"], {σ, 1, 3}]];
Show[GraphicsArray[{p1, p2}], ImageSize -> 400];
```

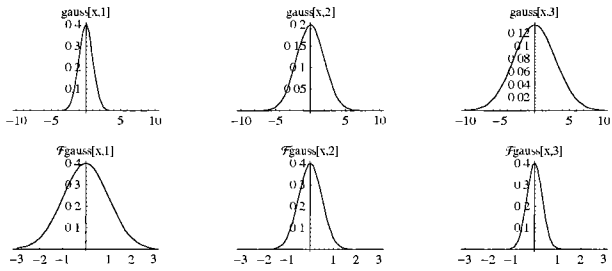


Figure 3.10 Top row: Gaussian function at scales σ=1, σ=2 and σ=3. Bottom row: Fourier transform of the Gaussian function above it. Note that for wider Gaussian its Fourier transform gets narrower and vice versa, a well known phenomenon with the Fourier transform. Also note by checking the amplitudes that the kernel is normalized in the spatial domain only.

There are many names for the Fourier transform $\mathcal{F}\,g(\omega;\sigma)$ of $g(x;\sigma)$: when the kernel $g(x;\sigma)$ is considered to be the point spread function, $\mathcal{F}\,g(\omega;\sigma)$ is referred to as the *modulation transfer function*. When the kernel g(x;σ) is considered to be a signal, $\mathcal{F}\,g(\omega;\sigma)$ is referred to as the *spectrum*. When applied to a signal, it operates as a lowpass *filter*. Let us

plot the spectra of a series of such filters (with a logarithmic increase in scale) on double logarithmic paper:

```
scales = N[Table[ Exp[t / 3], {t, 0, 8}]]
spectra = LogLinearPlot[ℱgauss[ω, #],
     {ω, .01, 10}, DisplayFunction -> Identity] & /@ scales;
Show[spectra, DisplayFunction -> $DisplayFunction, AspectRatio -> .4,
  PlotRange -> All, AxesLabel -> {"ω", "Amplitude"}, ImageSize -> 300];

{1., 1.39561, 1.94773, 2.71828,
 3.79367, 5.29449, 7.38906, 10.3123, 14.3919}
```

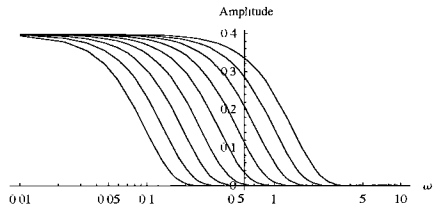


Figure 3.11 Fourier spectra of the Gaussian kernel for an exponential range of scales σ = (most right graph) to σ = 14.39 (most left graph). The frequency ω is on a logarithmic scale. The Gaussian kernels are seen to act as *low-pass* filters.

Due to this behaviour the role of receptive fields as lowpass filters has long persisted. But the retina does *not* measure a Fourier transform of the incoming image, as we will discuss in the chapters on the visual system (chapters 9-12).

3.9 Central limit theorem

We see in the paragraph above the relation with the *central limit theorem*: any repetitive operator goes in the limit to a Gaussian function. Later, when we study the discrete implementation of the Gaussian kernel and discrete sampled data, we will see the relation between interpolation schemes and the binomial coefficients. We study a repeated convolution of two blockfunctions with each other:

```
f[x_] := UnitStep[1 / 2 + x] + UnitStep[1 / 2 - x] - 1;
g[x_] := UnitStep[1 / 2 + x] + UnitStep[1 / 2 - x] - 1;

Plot[f[x], {x, -3, 3}, ImageSize -> 140];
```

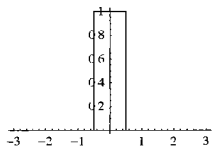


Figure 3.12 The analytical blockfunction is a combination of two Heavyside unitstep functions.

We calculate analytically the convolution integral

```
h1 = Integrate[f[x] g[x - x1], {x, -∞, ∞}]
```

$$\frac{1}{2}\,(-1 + 2\,\text{UnitStep}[1 - x1] - 2\,x1\,\text{UnitStep}[1 - x1] - 2\,x1\,\text{UnitStep}[x1]) +$$
$$\frac{1}{2}\,(-1 + 2\,x1\,\text{UnitStep}[-x1] + 2\,\text{UnitStep}[1 + x1] + 2\,x1\,\text{UnitStep}[1 + x1])$$

```
Plot[h1, {x1, -3, 3}, PlotRange -> All, ImageSize -> 150];
```

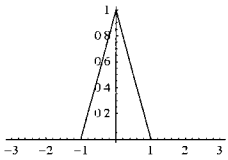


Figure 3.13 One times a convolution of a blockfunction with the same blockfunction gives a triangle function.

The next convolution is this function convolved with the block function again:

```
h2 = Integrate[(h1 /. x1 -> x) g[x - x1], {x, -∞, ∞}]
```

$$-1 + \frac{1}{8}\,(1 - 2\,x1)^2 + \frac{1}{8}\,(1 + 2\,x1)^2 + \frac{1}{8}\,(3 - 4\,x1 - 4\,x1^2) + \frac{1}{8}\,(3 + 4\,x1 - 4\,x1^2) + \frac{1}{8}$$
$$\left(-4 + 9\,\text{UnitStep}\left[\frac{3}{2} - x1\right] - 12\,x1\,\text{UnitStep}\left[\frac{3}{2} - x1\right] + 4\,x1^2\,\text{UnitStep}\left[\frac{3}{2} - x1\right] + \right.$$
$$\left.\text{UnitStep}\left[-\frac{1}{2} + x1\right] - 4\,x1\,\text{UnitStep}\left[-\frac{1}{2} + x1\right] + 4\,x1^2\,\text{UnitStep}\left[-\frac{1}{2} + x1\right]\right) +$$
$$\frac{1}{4}\left(-\text{UnitStep}\left[\frac{1}{2} - x1\right] + 4\,x1\,\text{UnitStep}\left[\frac{1}{2} - x1\right] - 4\,x1^2\,\text{UnitStep}\left[\frac{1}{2} - x1\right] - \right.$$
$$\left.\text{UnitStep}\left[\frac{1}{2} + x1\right] - 4\,x1\,\text{UnitStep}\left[\frac{1}{2} + x1\right] - 4\,x1^2\,\text{UnitStep}\left[\frac{1}{2} + x1\right]\right) +$$
$$\frac{1}{8}\left(-4 + \text{UnitStep}\left[-\frac{1}{2} - x1\right] + 4\,x1\,\text{UnitStep}\left[-\frac{1}{2} - x1\right] + \right.$$
$$4\,x1^2\,\text{UnitStep}\left[-\frac{1}{2} - x1\right] + 9\,\text{UnitStep}\left[\frac{3}{2} + x1\right] +$$
$$\left.12\,x1\,\text{UnitStep}\left[\frac{3}{2} + x1\right] + 4\,x1^2\,\text{UnitStep}\left[\frac{3}{2} + x1\right]\right)$$

$$-1 + \frac{1}{8}\,(1 - 2\,x1)^2 + \frac{1}{8}\,(1 + 2\,x1)^2 + \frac{1}{8}\,(3 - 4\,x1 - 4\,x1^2) + \frac{1}{8}\,(3 + 4\,x1 - 4\,x1^2) +$$
$$\frac{1}{8}\left(-4 + 9\,\text{UnitStep}\left[\frac{3}{2} - x1\right] - 12\,x1\,\text{UnitStep}\left[\frac{3}{2} - x1\right] + 4\,x1^2\,\text{UnitStep}\left[\frac{3}{2} - x1\right] + \right.$$
$$\left.\text{UnitStep}\left[-\frac{1}{2} + x1\right] - 4\,x1\,\text{UnitStep}\left[-\frac{1}{2} + x1\right] + 4\,x1^2\,\text{UnitStep}\left[-\frac{1}{2} + x1\right]\right) + \frac{1}{4}$$
$$\left(-\text{UnitStep}\left[\frac{1}{2} - x1\right] + 4\,x1\,\text{UnitStep}\left[\frac{1}{2} - x1\right] - 4\,x1^2\,\text{UnitStep}\left[\frac{1}{2} - x1\right] - \right.$$
$$\left.\text{UnitStep}\left[\frac{1}{2} + x1\right] - 4\,x1\,\text{UnitStep}\left[\frac{1}{2} + x1\right] - 4\,x1^2\,\text{UnitStep}\left[\frac{1}{2} + x1\right]\right) +$$
$$\frac{1}{8}\left(-4 + \text{UnitStep}\left[-\frac{1}{2} - x1\right] + 4\,x1\,\text{UnitStep}\left[-\frac{1}{2} - x1\right] + 4\,x1^2\,\text{UnitStep}\left[-\frac{1}{2} - x1\right] + \right.$$
$$\left.9\,\text{UnitStep}\left[\frac{3}{2} + x1\right] + 12\,x1\,\text{UnitStep}\left[\frac{3}{2} + x1\right] + 4\,x1^2\,\text{UnitStep}\left[\frac{3}{2} + x1\right]\right)$$

We see that we get a result that begins to look more towards a Gaussian:

```
Plot[{h2, gauss[x1, .5]}, {x1, -3, 3}, PlotRange -> All,
  PlotStyle -> {Dashing[{}], Dashing[{0.02, 0.02}]}, ImageSize -> 150];
```

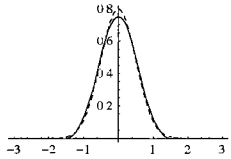


Figure 3.14 Two times a convolution of a blockfunction with the same blockfunction gives a function that rapidly begins to look like a Gaussian function. A Gaussian kernel with $\sigma = 0.5$ is drawn (dotted) for comparison.

The real Gaussian is reached when we apply an infinite number of these convolutions with the same function. It is remarkable that this result applies for the infinite repetition of *any* convolution kernel. This is the *central limit theorem.*

▲ Task 3.1 Show the central limit theorem in practice for a number of other arbitrary kernels.

3.10 Anisotropy

```
PlotGradientField[-gauss[x, 1] gauss[y, 1],
  {x, -3, 3}, {y, -3, 3}, PlotPoints -> 20, ImageSize -> 140];
```

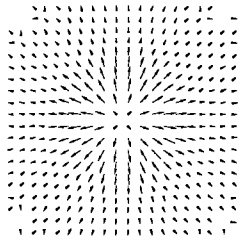


Figure 3.15 The slope of an isotropic Gaussian function is indicated by arrows here. There are circularly symmetric, i.e. in all directions the same, from which the name *isotropic* derives. The arrows are in the direction of the *normal* of the intensity landscape, and are called *gradient vectors.*

The Gaussian kernel as specified above is *isotropic*, which means that the behaviour of the function is in any *direction* the same. For 2D this means the Gaussian function is circular, for 3D it looks like a fuzzy sphere.

It is of no use to speak of isotropy in 1-D. When the standard deviations in the different dimensions are not equal, we call the Gaussian function *anisotropic*. An example is the pointspreadfunction of an astigmatic eye, where differences in curvature of the cornea/lens in different directions occur. This show an anisotropic Gaussian with anisotropy ratio of 2 ($\sigma_x / \sigma_y = 2$):

```
Unprotect[gauss];
gauss[x_, y_, σx_, σy_] := 1/(2 π σx σy) Exp[-(x^2/(2 σx^2) + y^2/(2 σy^2))];
σx = 2; σy = 1; Block[{$DisplayFunction = Identity},
 p1 = DensityPlot[gauss[x, y, σx, σy],
   {x, -10, 10}, {y, -10, 10}, PlotPoints -> 50];
 p2 = Plot3D[gauss[x, y, σx, σy], {x, -10, 10},
   {y, -10, 10}, Shading -> True];
 p3 = ContourPlot[gauss[x, y, σx, σy], {x, -5, 5}, {y, -10, 10}]];
Show[GraphicsArray[{p1, p2, p3}], ImageSize -> 400];
```

Figure 3.16 An anisotropic Gaussian kernel with anisotropy ratio $\sigma_x / \sigma_y = 2$ in three appearances. Left: `DensityPlot`, middle: `Plot3D`, right: `ContourPlot`.

3.11 The diffusion equation

The Gaussian function is the solution of several differential equations. It is the solution of $\frac{dy}{dx} = \frac{y(\mu - x)}{\sigma^2}$, because $\frac{dy}{y} = \frac{(\mu - x)}{\sigma^2} dx$, from which we find by integration $\ln\left(\frac{y}{y_0}\right) = -\frac{(\mu - x)^2}{2\sigma^2}$ and thus $y = y_0 \, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$.

It is the solution of the linear diffusion equation, $\frac{\partial L}{\partial t} = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2} = \Delta L$.

This is a partial differential equation, stating that the first derivative of the (luminance) function $L(x, y)$ to the parameter t (time, or variance) is equal to the sum of the second order spatial derivatives. The right hand side is also known as the Laplacian (indicated by Δ for any dimension, we call Δ the *Laplacian operator*), or the trace of the Hessian matrix of second order derivatives:

```
hessian2D = {{Lxx, Lxy}, {Lxy, Lyy}}; Tr[hessian2D]
```

$$L_{xx} + L_{yy}$$

```
hessian3D = {{Lxx, Lxy, Lxz}, {Lyx, Lyy, Lyz}, {Lzx, Lyz, Lzz}}; Tr[hessian3D]
```

$$L_{xx} + L_{yy} + L_{zz}$$

The diffusion equation $\frac{\partial u}{\partial t} = \Delta u$ is one of some of the most famous differential equations in physics. It is often referred to as the *heat equation*. It belongs in the row of other famous

equations like the Laplace equation $\Delta u = 0$, the wave equation $\frac{\partial^2 u}{\partial t^2} = \Delta u$ and the Schrödinger equation $\frac{\partial u}{\partial t} = i \Delta u$.

The diffusion equation $\frac{\partial u}{\partial t} = \Delta u$ is a *linear* equation. It consists of just linearly combined derivative terms, no nonlinear exponents or functions of derivatives.

The diffused entity is the intensity in the images. The role of time is taken by the variance $t = 2\sigma^2$. The intensity is diffused over time (in our case over scale) in all directions in the same way (this is called *isotropic*). E.g. in 3D one can think of the example of the intensity of an inkdrop in water, diffusing in all directions.

The diffusion equation can be derived from physical principles: the luminance can be considered a *flow*, that is pushed away from a certain location by a force equal to the gradient. The divergence of this gradient gives how much the total entity (luminance in our case) diminishes with time.

```
<< Calculus`VectorAnalysis`
SetCoordinates[Cartesian[x, y, z]];

Div[ Grad[L[x, y, z]]]

L^(0,0,2) [x, y, z] + L^(0,2,0) [x, y, z] + L^(2,0,0) [x, y, z]
```

A very important feature of the diffusion process is that it satisfies a *maximum principle* [Hummel1987b]: the amplitude of local maxima are always decreasing when we go to coarser scale, and vice versa, the amplitude of local minima always increase for coarser scale. This argument was the principal reasoning in the derivation of the diffusion equation as the generating equation for scale-space by Koenderink [Koenderink1984a].

3.12 Summary of this chapter

The normalized Gaussian kernel has an area under the curve of unity, i.e. as a filter it does not multiply the operand with an accidental multiplication factor. Two Gaussian functions can be cascaded, i.e. applied consecutively, to give a Gaussian convolution result which is equivalent to a kernel with the variance equal to the sum of the variances of the constituting Gaussian kernels. The spatial parameter normalized over scale is called the dimensionless 'natural coordinate'.

The Gaussian kernel is the 'blurred version' of the Delta Dirac function, the cumulative Gaussian function is the Error function, which is the 'blurred version' of the Heavyside stepfunction. The Dirac and Heavyside functions are examples of *generalized functions*.

The Gaussian kernel appears as the limiting case of the Pascal Triangle of binomial coefficients in an expanded polynomial of high order. This is a special case of the central limit theorem. The central limit theorem states that any finite kernel, when repeatedly convolved with itself, leads to the Gaussian kernel.

Anisotropy of a Gaussian kernel means that the scales, or standard deviations, are different for the different dimensions. When they are the same in all directions, the kernel is called *isotropic*.

The Fourier transform of a Gaussian kernel acts as a low-pass filter for frequencies. The cut-off frequency depends on the scale of the Gaussian kernel. The Fourier transform has the same Gaussian shape. The Gaussian kernel is the *only* kernel for which the Fourier transform has the same shape.

The diffusion equation describes the expel of the flow of some quantity (intensity, temperature) over space under the force of a gradient. It is a second order parabolic differential equation. The linear, isotropic diffusion equation is the generating equation for a scale-space. In chapter 21 we will encounter a wealth on nonlinear diffusion equations.

4. Gaussian derivatives

A difference which makes no difference is not a difference.
Mr. Spock (stardate 2822.3)

4.1 Introduction

We will encounter the Gaussian derivative function at many places throughout this book. The Gaussian derivative function has many interesting properties. We will discuss them in one dimension first. We study its shape and algebraic structure, its Fourier transform, and its close relation to other functions like the Hermite functions, the Gabor functions and the generalized functions. In two and more dimensions additional properties are involved like orientation (directional derivatives) and anisotropy.

4.2 Shape and algebraic structure

When we take derivatives to x (*spatial derivatives*) of the Gaussian function repetitively, we see a pattern emerging of a polynomial of increasing order, multiplied with the original (normalized) Gaussian function again. Here we show a table of the derivatives from order 0 (i.e. no differentiation) to 3.

```
<< FrontEndVision`FEV`;
Unprotect[gauss]; gauss[x_, σ_] := 1/(σ Sqrt[2 π]) Exp[-x^2/(2 σ^2)];
Table[Factor[Evaluate[D[gauss[x, σ], {x, n}]]], {n, 0, 4}]
```

$$\left\{\frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma},\ -\frac{e^{-\frac{x^2}{2\sigma^2}}\,x}{\sqrt{2\pi}\,\sigma^3},\ \frac{e^{-\frac{x^2}{2\sigma^2}}\,(x-\sigma)\,(x+\sigma)}{\sqrt{2\pi}\,\sigma^5},\right.$$
$$\left.-\frac{e^{-\frac{x^2}{2\sigma^2}}\,x\,(x^2-3\sigma^2)}{\sqrt{2\pi}\,\sigma^7},\ \frac{e^{-\frac{x^2}{2\sigma^2}}\,(x^4-6x^2\sigma^2+3\sigma^4)}{\sqrt{2\pi}\,\sigma^9}\right\}$$

The function **Factor** takes polynomial factors apart.

The function **gauss[x,σ]** is part of the standard set of functions (in **FEV.m**) with this book, and is protected. To modify it, it must be **Unprotected**.

The zeroth order derivative is indeed the Gaussian function itself. The even order (including the zeroth order) derivative functions are even functions (i.e. symmetric around zero) and the odd order derivatives are odd functions (antisymmetric around zero). This is how the graphs of Gaussian derivative functions look like, from order 0 up to order 7 (note the marked increase in amplitude for higher order of differentiation):

```
GraphicsArray[
  Partition[Table[Plot[Evaluate[D[gauss[x, 1], {x, n}]], {x, -5, 5},
    PlotLabel -> StringJoin["Order=", ToString[n]], DisplayFunction ->
     Identity], {n, 0, 7}], 4], ImageSize -> 500] // Show;
```

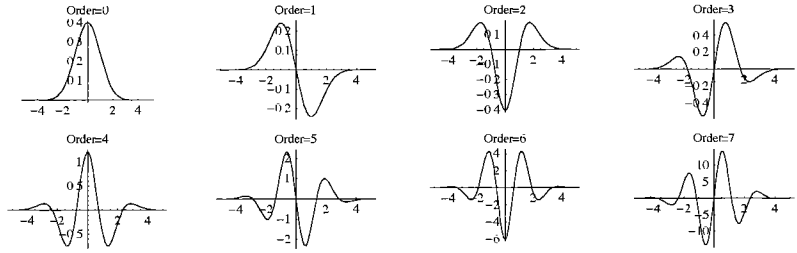


Figure 4.1 Plots of the 1D Gaussian derivative function for order 0 to 7.

The Gaussian function itself is a common element of all higher order derivatives. We extract the polynomials by dividing by the Gaussian function:

```
Table[Evaluate[D[gauss[x, σ], {x, n}] / gauss[x, σ]], {n, 0, 4}] // Simplify
```

$$\left\{1, -\frac{x}{\sigma^2}, \frac{x^2 - \sigma^2}{\sigma^4}, -\frac{x^3 - 3\,x\,\sigma^2}{\sigma^6}, \frac{x^4 - 6\,x^2\,\sigma^2 + 3\,\sigma^4}{\sigma^8}\right\}$$

These polynomials have the same order as the derivative they are related to. Note that the highest order of x is the same as the order of differentiation, and that we have a plus sign for the highest order of x for even number of differentiation, and a minus signs for the odd orders.

These polynomials are the Hermite polynomials, called after Charles Hermite, a brilliant French mathematician (see figure 4.2).

```
Show[Import["Charles Hermite.jpg"], ImageSize -> 150];
```

Figure 4.2 Charles Hermite (1822-1901).

They emerge from the following definition: $\frac{\partial^n e^{-x^2}}{\partial x^n} = (-1)^n H_n(x) e^{-x^2}$. The function $H_n(x)$ is the Hermite polynomial, where n is called the order of the polynomial. When we make the substitution $x \to x/(\sigma\sqrt{2})$, we get the following relation between the Gaussian function $G(x, \sigma)$ and its derivatives: $\frac{\partial^n G(x,\sigma)}{\partial x^n} = (-1)^n \frac{1}{(\sigma\sqrt{2})^n} H_n\left(\frac{x}{\sigma\sqrt{2}}\right) G(x, \sigma)$.

In *Mathematica* the function H_n is given by the function **HermiteH[n,x]**. Here are the Hermite functions from zeroth to fifth order:

```
Table[HermiteH[n, x], {n, 0, 7}] // TableForm
```

$$
\begin{aligned}
&1\\
&2x\\
&-2+4x^2\\
&-12x+8x^3\\
&12-48x^2+16x^4\\
&120x-160x^3+32x^5\\
&-120+720x^2-480x^4+64x^6\\
&-1680x+3360x^3-1344x^5+128x^7
\end{aligned}
$$

The inner scale σ is introduced in the equation by substituting $x \to \frac{x}{\sigma\sqrt{2}}$. As a consequence, with each differentiation we get a new factor $\frac{1}{\sigma\sqrt{2}}$. So now we are able to calculate the 1-D Gaussian derivative functions **gd[x,n,σ]** directly with the Hermite polynomials, again incorporating the normalization factor $\frac{1}{\sigma\sqrt{2\pi}}$:

```
Clear[σ];
gd[x_, n_, σ_] := (-1/(σ Sqrt[2]))^n HermiteH[n, x/(σ Sqrt[2])] 1/(σ Sqrt[2 π]) Exp[-x^2/(2 σ^2)]
```

Check:

```
Simplify[gd[x, 4, σ], σ > 0]
```

$$\frac{e^{-\frac{x^2}{2\sigma^2}}\left(x^4-6x^2\sigma^2+3\sigma^4\right)}{\sqrt{2\pi}\,\sigma^9}$$

```
Simplify[D[1/(σ Sqrt[2 π]) Exp[-x^2/(2 σ^2)], {x, 4}], σ > 0]
```

$$\frac{e^{-\frac{x^2}{2\sigma^2}}\left(x^4-6x^2\sigma^2+3\sigma^4\right)}{\sqrt{2\pi}\,\sigma^9}$$

The amplitude of the Hermite polynomials explodes for large x, but the Gaussian envelop suppresses *any* polynomial function. No matter how high the polynomial order, the exponential function always wins. We can see this graphically when we look at e.g. the 7th order Gaussian derivative without (i.e. the Hermite function, figure left) and with its Gaussian weight function (figure right). Note the vertical scales:

```
f[x_] := (1/Sqrt[2])^7 HermiteH[7, x/Sqrt[2]];
DisplayTogetherArray[{Plot[f[x], {x, -5, 5}],
  p2 = Plot[f[x] Exp[-x^2/2], {x, -5, 5}]}, ImageSize -> 400];
```

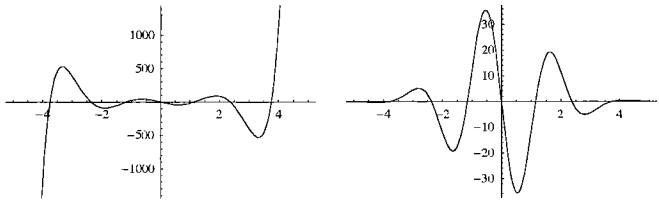


Figure 4.3 Left: The 7th order Hermite polynomial. Right: idem, with a Gaussian envelope (weighting function). This is the 7th order Gaussian derivative kernel.

Due to the limiting extent of the Gaussian window function, the amplitude of the Gaussian derivative function can be negligible at the location of the larger zeros. We plot an example, showing the 20th order derivative and its Gaussian envelope function:

```
n = 20; σ = 1; DisplayTogether[{FilledPlot[gd[x, n, σ], {x, -5, 5}],
  Plot[gd[0, n, σ] gauss[x, σ] / gauss[0, σ], {x, -5, 5}]}, ImageSize -> 200];
```

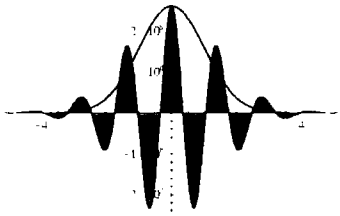


Figure 4.4 The 20th order Gaussian derivative's outer zero-crossings vahish in negligence. Note also that the amplitude of the Gaussian derivative function is not bounded by the Gaussian window. The Gabor kernels, as we will discuss later in section 4.7, are bounded by the Gaussian window.

How fast the Gaussian function goes zero can be seen from its values at $x = 3\sigma$, $x = 4\sigma$ and $x = 5\sigma$, relative to its peak value:

```
Table[gauss[σ, 1] / gauss[0, 1], {σ, 3, 5}] // N
```

```
{0.011109, 0.000335463, 3.72665×10^-6}
```

and in the limit:

```
Limit[gd[x, 7, 1], x -> Infinity]
```

```
0
```

The Hermite polynomials belong to the family of orthogonal functions on the infinite interval (-∞,∞) with the weight function e^{-x^2}, $\int_{-\infty}^{\infty} e^{-x^2} H_n(x) H_m(x) dx = 2^{n+m} n! \sqrt{\pi}\, \delta_{nm}$, where δ_{nm} is the Kronecker delta, or delta tensor. $\delta_{nm} = 1$ for $n = m$, and $\delta_{nm} = 0$ for $n \neq m$.

```
Table[∫_-∞^∞ Exp[-x^2] HermiteH[k, x] HermiteH[m, x] dx,
  {k, 0, 3}, {m, 0, 3}] // MatrixForm
```

$$\begin{pmatrix} \sqrt{\pi} & 0 & 0 & 0 \\ 0 & 2\sqrt{\pi} & 0 & 0 \\ 0 & 0 & 8\sqrt{\pi} & 0 \\ 0 & 0 & 0 & 48\sqrt{\pi} \end{pmatrix}$$

The Gaussian derivative functions, with their weight function $e^{-\frac{x^2}{2}}$ are *not* orthogonal. We check this with some examples:

```
{∫_-∞^∞ gd[x, 2, 1] gd[x, 3, 1] dx, ∫_-∞^∞ gd[x, 2, 1] gd[x, 4, 1] dx}
```

$$\left\{0, -\frac{15}{16\sqrt{\pi}}\right\}$$

Other families of orthogonal polynomials are e.g. Legendre, Chebyshev, Laguerre, and Jacobi polynomials. Other orthogonal families of functions are e.g. Bessel functions and the spherical harmonic functions. The area under the Gaussian derivative functions is *not* unity, e.g. for the first derivative:

```
SetOptions[Integrate, GenerateConditions -> False];
∫_0^∞ gd[x, 1, σ] dx
```

$$-\frac{1}{\sqrt{2\pi}}$$

4.3 Gaussian derivatives in the Fourier domain

The Fourier transform of the derivative of a function is $(-i\omega)$ times the Fourier transform of the function. For each differentiation, a new factor $(-i\omega)$ is added. So the Fourier transforms of the Gaussian function and its first and second order derivatives are:

```
σ=.; Simplify[FourierTransform[
  {gauss[x, σ], ∂_x gauss[x, σ], ∂_{x,2} gauss[x, σ]}, x, ω], σ > 0]
```

$$\left\{\frac{e^{-\frac{1}{2}\sigma^2\omega^2}}{\sqrt{2\pi}}, -\frac{i\, e^{-\frac{1}{2}\sigma^2\omega^2}\,\omega}{\sqrt{2\pi}}, -\frac{e^{-\frac{1}{2}\sigma^2\omega^2}\,\omega^2}{\sqrt{2\pi}}\right\}$$

In general: $\mathcal{F}\left\{\frac{\partial^n G(x,\sigma)}{\partial x^n}\right\} = (-i\omega)^n \mathcal{F}\{G(x,\sigma)\}$.

Gaussian derivative kernels also act as bandpass filters. The maximum is at $\omega = \sqrt{n}$:

```
n=.; σ = 1; Solve[Evaluate[D[(-I ω)^n/√(2 π) Exp[-σ^2 ω^2/2], ω]] == 0, ω]
```

$$\left\{\left\{\omega \to i\, 0^{\frac{1}{n}}\right\}, \{\omega \to -\sqrt{n}\}, \{\omega \to \sqrt{n}\}\right\}$$

The normalized powerspectra show that higher order of differentiation means a higher center frequency for the *bandpass* filter. The bandwidth remains virtually the same.

```
σ = 1; pl = Table[Plot[Abs[((-I ω)^n/√(2 π) Exp[-σ^2 ω^2/2])/((-I n^(1/2))^n/√(2 π) Exp[-σ^2 n/2])],
   {ω, 0, 6}, DisplayFunction -> Identity], {n, 1, 12}];
Show[pl, DisplayFunction -> $DisplayFunction, PlotRange -> All,
  AxesLabel -> {"ω", ""}, ImageSize -> 400];
```

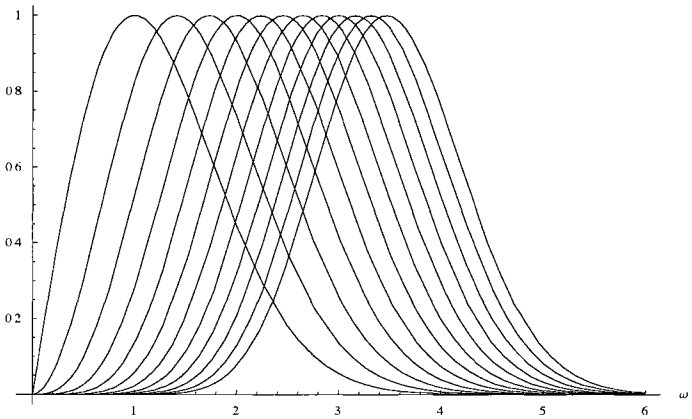


Figure 4.5 Normalized power spectra for Gaussian derivative filters for order 1 to 12, lowest order is left-most graph, $\sigma = 1$. Gaussian derivative kernels act like bandpass filters.

▲ Task 4.1 Show with partial integration and the definitions from section 3.10 that the Fourier transform of the derivative of a function is $(-i\omega)$ times the Fourier transform of the function.

▲ Task 4.2 Note that there are several definitions of the signs occurring in the Fourier transform (see the Help function in *Mathematica* under Fourier). Show that with the other definitions it is possible to arrive to the result that the Fourier

transform of the derivative of a function is $(i\,\omega)$ times the Fourier transform of the function. In this book we stick to the default definition.

4.4 Zero crossings of Gaussian derivative functions

```
gd[x_, n_, σ_] := (-1/(σ √2))^n HermiteH[n, x/(σ √2)] 1/(σ √(2 π)) Exp[-x^2/(2 σ^2)];
nmax = 20; σ = 1;
Show[Graphics[Flatten[Table[{PointSize[0.015], Point [{n, x}]} /.
    Solve[HermiteH[n, x/√2] == 0, x], {n, 1, nmax}], 1]],
 AxesLabel -> {"Order", "Zeros of\nHermiteH"}, Axes -> True,
 ImageSize -> 350];
```

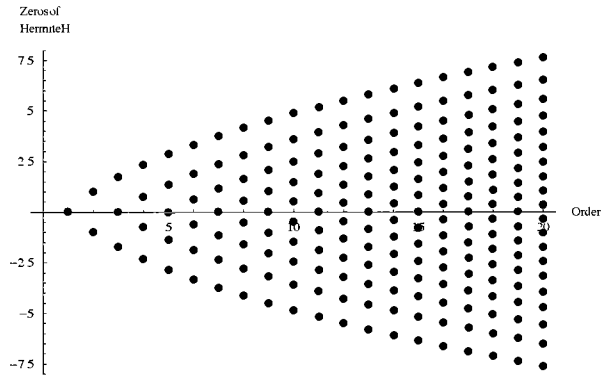


Figure 4.6 Zero crossings of Gaussian derivative functions to 20th order. Each dot is a zero-crossing.

How *wide* is a Gaussian derivative? This may seem a non-relevant question, because the Gaussian envelop often completely determines its behaviour. However, the number of zero-crossings is equal to the order of differentiation, because the Gaussian weighting function is a positive definite function.

It is of interest to study the behaviour of the zero-crossings. They move further apart with higher order. We can define the 'width' of a Gaussian derivative function as the distance between the outermost zero-crossings. The zero-crossings of the Hermite polynomials determine the zero-crossings of the Gaussian derivatives. In figure 4. 6all zeros of the first 20 Hermite functions as a function of the order are shown. Note that the zeros of the second derivative are just one standard deviation from the origin:

```
σ=.; Simplify[Solve[D[gauss[x, σ], {x, 2}] == 0, x], σ > 0]
{{x → -σ}, {x → σ}}
```

An exact analytic solution for the largest zero is not known. The formula of Zernicke (1931) specifies a range, and Szego (1939) gives a better estimate:

```
Block[{$DisplayFunction = Identity},
  p1 = Plot[2 Sqrt[n + 1 - 3.05 ∛(n + 1)], {n, 5, 50}];
  (* Zernicke upper limit *)
  p2 = Plot[2 Sqrt[n + 1 - 1.15 ∛(n + 1)], {n, 1, 50}];
  (* Zernicke lower limit *)
  p3 = Plot[2 √(n + .5) - 2.338098 / ∜(n + .5),
    {n, 1, 50}, PlotStyle -> Dashing[{.01, .02}]]];
Show[{p1, p2, p3}, AxesLabel -> {"Order",
    "Width of Gaussian\nderivative (in σ)"}, ImageSize -> 260];
```

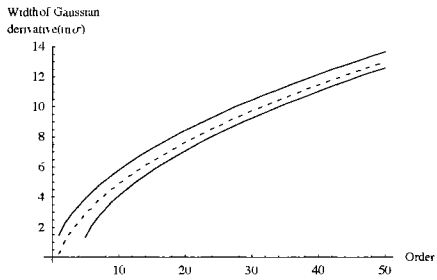


Figure 4.7 Estimates for the width of Gaussian derivative functions to 50th order. Width is defined as the distance between the outmost zero-crossings. Top and bottom graph: estimated range by Zernicke (1931), dashed graph: estimate by Szego (1939).

For very high orders of differentiation of course the numbers of zero-crossings increases, but also their mutual distance between the zeros becomes more equal. In the limiting case of infinite order the Gaussian derivative function becomes a sinusoidal function:

$$\lim_{n\to\infty} \frac{\partial^n G}{\partial^n x}(x, \sigma) = \mathrm{Sin}\left(x \sqrt{\frac{1}{\sigma}\left(\frac{n+1}{2}\right)}\right).$$

4.5 The correlation between Gaussian derivatives

Higher order Gaussian derivative kernels tend to become more and more similar. This makes them not very suitable as a basis. But before we investigate their role in a possible basis, let us investigate their similarity.

In fact we can express exactly how much they resemble each other as a function of the difference in differential order, by calculating the *correlation* between them. We derive the correlation below, and will appreciate the nice mathematical properties of the Gaussian function. Because the higher dimensional Gaussians are just the product of 1D Gaussian functions, it suffices to study the 1D case.

Compare e.g. the 20th and 24nd derivative function:

```
Block[{$DisplayFunction = Identity},
  g1 = Plot[gd[x, 20, 2], {x, -7, 7}, PlotLabel -> "Order 20"];
  g2 = Plot[gd[x, 24, 2], {x, -7, 7}, PlotLabel -> "Order 24"]];
Show[GraphicsArray[{g1, g2}, ImageSize -> 400]];
```

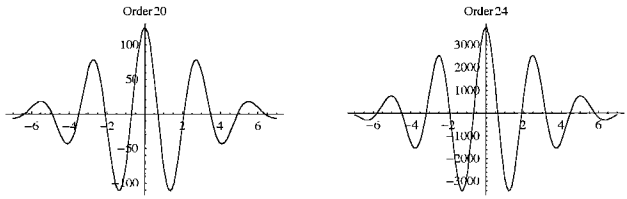


Figure 4.8 Gaussian derivative functions start to look more and more alike for higher order. Here the graphs are shown for the 20th and 24th order of differentiation.

The correlation coefficient between two functions is defined as the integral of the product of the functions over the full domain (in this case $-\infty$ to $+\infty$).

Because we want the coefficient to be unity for complete correlation (when the functions are identical by an amplitude scaling factor) we divide the coefficient by the so-called autocorrelation coefficients, i.e. the correlation of the functions with themselves.

We then get as definition for the correlation coefficient r between two Gaussian derivatives of order n and m:

$$r_{n,m} = \frac{\int_{-\infty}^{\infty} g^{(n)}(x)\, g^{(m)}(x)\, dx}{\sqrt{\int_{-\infty}^{\infty} [g^{(n)}(x)]^2\, dx \int_{-\infty}^{\infty} [g^{(m)}(x)]^2\, dx}}$$

with $g^{(n)}(x) = \frac{\partial^n g(x)}{\partial x^n}$. The Gaussian kernel $g(x)$ itself is an even function, and, as we have seen before, $g^{(n)}(x)$ is an even function for n is even, and an odd function for n is odd. The correlation between an even function and an odd function is zero. This is the case when n and m are both not even or both not odd, i.e. when $(n-m)$ is odd. We now can see already two important results:

$r_{n,m} = 0$ for $(n-m)$ odd;
$r_{n,m} = 1$ for $n = m$.

The remaining case is when $(n-m)$ is even. We take $n > m$. Let us first look to the nominator, $\int_{-\infty}^{\infty} g^{(n)}(x)\, g^{(m)}(x)\, dx$. The standard approach to tackle high exponents of functions in integrals, is the reduction of these exponents by partial integration:

$$\int_{-\infty}^{\infty} g^{(n)}(x)\, g^{(m)}(x)\, dx =$$
$$g^{(n-1)}(x)\, g^{(m)}(x)\,|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} g^{(n-1)}(x)\, g^{(m+1)}(x)\, dx = (-1)^k \int_{-\infty}^{\infty} g^{(n-k)}(x)\, g^{(m+k)}(x)\, dx$$

when we do the partial integration k times. The 'stick expression' $g^{(n-1)}(x)\, g^{(m)}(x)\,|_{-\infty}^{\infty}$ is zero because any Gaussian derivative function goes to zero for large x. We can choose k such that the exponents in the integral are equal (so we end up with the square of a Gaussian derivative

function). So we make $(n-k)=(m+k)$, i.e. $k=\frac{(n-m)}{2}$. Because we study the case that $(n-m)$ is even, k is an integer number. We then get:

$$(-1)^k \int_{-\infty}^{\infty} g^{(n-k)}(x)\, g^{(m+k)}(x)\, dx = (-1)^{\frac{n-m}{2}} \int_{-\infty}^{\infty} g^{(\frac{n+m}{2})}(x)\, g^{(\frac{n+m}{2})}(x)\, dx$$

The *total energy* of a function in the spatial domain is the integral of the square of the function over its full extent. The famous theorem of Parceval states that the total energy of a function in the spatial domain is equal to the total energy of the function in the Fourier domain, i.e. expressed as the integral of the square of the Fourier transform over its full extent. Therefore

$$(-1)^{\frac{n-m}{2}} \int_{-\infty}^{\infty} g^{(\frac{n+m}{2})}(x)\, g^{(\frac{n+m}{2})}(x)\, dx \overset{\text{Parceval}}{=} (-1)^{\frac{n-m}{2}} \frac{1}{2\pi} \int_{-\infty}^{\infty} \left|(i\omega)^{\frac{n+m}{2}}\, \hat{g}(\omega)\right|^2 d\omega =$$
$$(-1)^{\frac{n-m}{2}} \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega^{n+m}\, \hat{g}^2(\omega)\, d\omega = (-1)^{\frac{n-m}{2}} \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega^{n+m}\, e^{-\sigma^2\omega^2}\, d\omega$$

We now substitute $\omega' = \sigma\omega$, and get finally: $(-1)^{\frac{n-m}{2}} \frac{1}{2\pi} \frac{1}{\sigma^{n+m+1}} \int_{-\infty}^{\infty} \omega'^{(n+m)}\, e^{-\omega'^2}\, d\omega'$.

This integral can be looked up in a table of integrals, but why not let *Mathematica* do the job (we first clear n and m):

```
Clear[n, m]; ∫_-∞^∞ x^(m+n) e^(-x^2) dx
```

$$\frac{1}{2}\,(1+(-1)^{m+n})\,\text{Gamma}\left[\frac{1}{2}\,(1+m+n)\right]\text{Log}[e]^{\frac{1}{2}(-1-m-n)}$$

The function **Gamma** is the Euler gamma function. In our case **Re[m+n]>-1**, so we get for our correlation coefficient for $(n-m)$ even:

$$r_{n,m} = \frac{(-1)^{\frac{n-m}{2}} \frac{1}{2\pi} \frac{1}{\sigma^{n+m+1}} \Gamma\left(\frac{m+n+1}{2}\right)}{\sqrt{\frac{1}{2\pi}\frac{1}{\sigma^{2n+1}}\Gamma\left(\frac{2n+1}{2}\right)\frac{1}{2\pi}\frac{1}{\sigma^{2m+1}}\Gamma\left(\frac{2m+1}{2}\right)}} = \frac{(-1)^{\frac{n-m}{2}}\Gamma\left(\frac{m+n+1}{2}\right)}{\sqrt{\Gamma\left(\frac{2n+1}{2}\right)\Gamma\left(\frac{2m+1}{2}\right)}}$$

Let's first have a look at this function for a range of values for n and m (0-15):

```
r[n_, m_] := (-1)^((n-m)/2) Gamma[(m+n+1)/2] / Sqrt[Gamma[(2n+1)/2] Gamma[(2m+1)/2]];
ListPlot3D[Table[Abs[r[n, m]], {n, 0, 15}, {m, 0, 15}],
  Axes -> True, AxesLabel -> {"n", "m", "Abs\nr[n,m]"},
  ViewPoint -> {-2.348, -1.540, 1.281}, ImageSize -> 220];
```

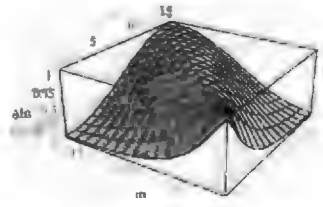


Figure 4.9 The magnitude of the correlation coefficient of Gaussian derivative functions for $0 < n < 15$ and $0 < m < 15$. The origin is in front.

Here is the function tabulated:.

```
Table[NumberForm[r[n, m] // N, 3], {n, 0, 4}, {m, 0, 4}] // MatrixForm
```

$$\begin{pmatrix} 1. & 0.-0.798\,i & -0.577 & 0.+0.412\,i & 0.293 \\ 0.+0.798\,i & 1. & 0.-0.921\,i & -0.775 & 0.+0.623\,i \\ -0.577 & 0.+0.921\,i & 1. & 0.-0.952\,i & -0.845 \\ 0.-0.412\,i & -0.775 & 0.+0.952\,i & 1. & 0.-0.965\,i \\ 0.293 & 0.-0.623\,i & -0.845 & 0.+0.965\,i & 1. \end{pmatrix}$$

The correlation is unity when $n = m$, as expected, is negative when $n - m = 2$, and is positive when $n - m = 4$, and is complex otherwise. Indeed we see that when $n - m = 2$ the functions are even but of opposite sign:

```
Block[{$DisplayFunction = Identity},
  p1 = Plot[gd[x, 20, 2], {x, -5, 5}, PlotLabel -> "Order 20"];
  p2 = Plot[gd[x, 22, 2], {x, -5, 5}, PlotLabel -> "Order 22"]];
Show[GraphicsArray[{p1, p2}, ImageSize -> 450]];
```

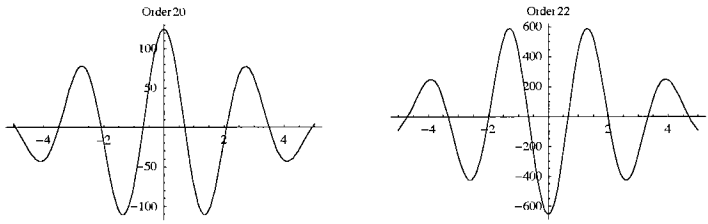


Figure 4.10 Gaussian derivative functions differing two orders are of opposite polarity.

and when $n - m = 1$ they have a phase-shift, leading to a complex correlation coefficient:

```
Block[{$DisplayFunction = Identity},
 p1 = Plot[gd[x, 20, 2], {x, -5, 5}, PlotLabel -> "Order 20"];
 p2 = Plot[gd[x, 21, 2], {x, -5, 5}, PlotLabel -> "Order 21"]];
Show[GraphicsArray[{p1, p2}, ImageSize -> 450]];
```

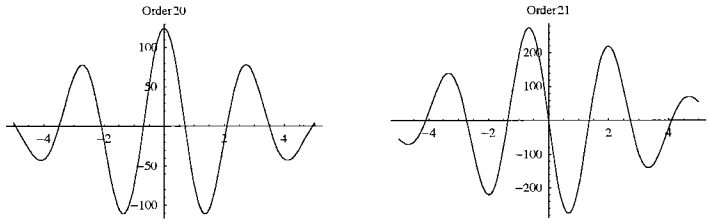


Figure 4.11 Gaussian derivative functions differing one order display a phase shift.

Of course, this is easy understood if we realize the factor $(-i\omega)$ in the Fourier domain, and that $i = e^{-i\frac{\pi}{2}}$. We plot the behaviour of the correlation coefficient of two close orders for large n. The asymptotic behaviour towards unity for increasing order is clear.

```
Plot[-r[n, n + 2], {n, 1, 20}, DisplayFunction -> $DisplayFunction,
  AspectRatio -> .4, PlotRange -> {.8, 1.01},
  AxesLabel -> {"Order", "Correlation\ncoefficient"}];
```

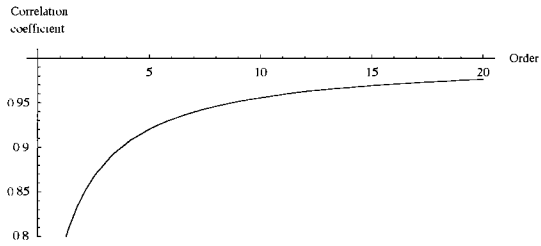


Figure 4.12 The correlation coefficient between a Gaussian derivative function and its ever neighbour up quite quickly tends to unity for high differential order.

4.6 Discrete Gaussian kernels

```
σ = 2; Plot[{1/Sqrt[2 π σ^2] Exp[-x^2/(2 σ^2)], 1/Sqrt[2 π σ^2] BesselI[x, σ^2] / BesselI[0, σ^2]},
 {x, 0, 8}, PlotStyle → {RGBColor[0, 0, 0], Dashing[{0.02, 0.02}]},
 PlotLegend → {"Gauss", "Bessel"}, LegendPosition → {1, 0},
 LegendLabel → "σ = 2", PlotRange → All, ImageSize → 400];
```

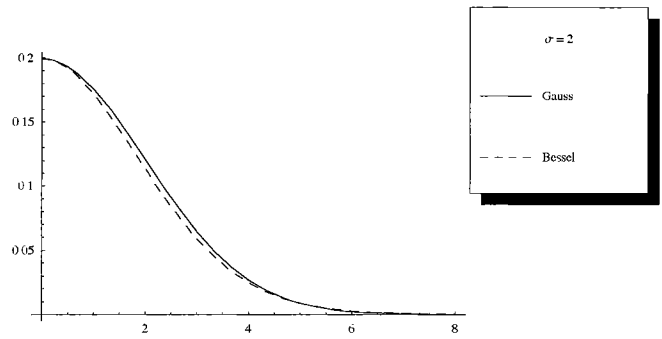


Figure 4.13 The graphs of the Gaussian kernel and the modified Bessel function of the first kind are very alike.

Lindeberg [Lindeberg1990] derived the optimal kernel for the case when the Gaussian kernel was discretized and came up with the "modified Bessel function of the first kind". In *Mathematica* this function is available as **BesselI**. This function is almost equal to the Gaussian kernel for $\sigma > 1$, as we see in the figure on the previous page. Note that the Bessel function has to be normalized by its value at $x = 0$. For larger σ the kernels become rapidly very similar.

4.7 Other families of kernels

The first principles that we applied to derive the Gaussian kernel in chapter 2 essentially stated "we know nothing" (at this stage of the observation). Of course, we can relax these principles, and introduce some knowledge. When we want to derive a set of apertures tuned to a *specific spatial frequency* $\vec{k}$ in the image, we add this physical quantity to the matrix of the dimensionality analysis:

```
m = {{1, -1, -2, -2, -1}, {0, 0, 1, 1, 0}};
TableForm[m,
 TableHeadings -> {{"meter", "candela"}, {"σ", "ω", "L0", "L", "k"}}]
```

	σ	ω	L0	L	k
meter	1	-1	-2	-2	-1
candela	0	0	1	1	0

The nullspace is now:

```
NullSpace[m]
```

```
{{1, 0, 0, 0, 1}, {0, 0, -1, 1, 0}, {1, 1, 0, 0, 0}}
```

Following the exactly similar line of reasoning, we end up from this new set of constraints with a new family of kernels, the *Gabor family of receptive fields*, with are given by a sinusoidal function (at the specified spatial frequency) under a Gaussian window.

In the Fourier domain: $\mathcal{G}abor(\omega, \sigma, k) = e^{-\omega^2 \sigma^2} e^{\imath k \omega}$, which translates into the spatial domain:

$$\texttt{gabor[x\_, σ\_] := Sin[x]} \frac{1}{\sqrt{2\pi\sigma^2}} \texttt{Exp}\left[-\frac{x^2}{2\sigma^2}\right];$$

The Gabor function model of cortical receptive fields was first proposed by Marcelja in 1980 [Marcelja1980]. However the functions themselves are often credited to Gabor [Gabor1946] who supported their use in communications.

Gabor functions are defined as the sinus function under a Gaussian window with scale σ. The phase ϕ of the sinusoidal function determines its detailed behaviour, e.g. for $\phi = \pi/2$ we get an even function.Gabor functions can look very much like Gaussian derivatives, but there are essential differences:

- Gabor functions have an infinite number of zero-crossings on their domain.

- The amplitudes of the sinusoidal function never exceeds the Gaussian envelope.

```
gabor[x_, ϕ_, σ_] := Sin[x + ϕ] gauss[x, σ];
Block[{$DisplayFunction = Identity, p, pg},
  p = Plot[gabor[x, #, 10], {x, -30, 30}, PlotRange -> {-.04, .04}] &;
  pg = Plot[gauss[x, 10], {x, -30, 30}, PlotRange -> {-.04, .04}];
  p13 = Show[p[0], pg]; p23 = Show[p[π / 2], pg]];

Show[GraphicsArray[{p13, p23}], ImageSize -> 450];
```

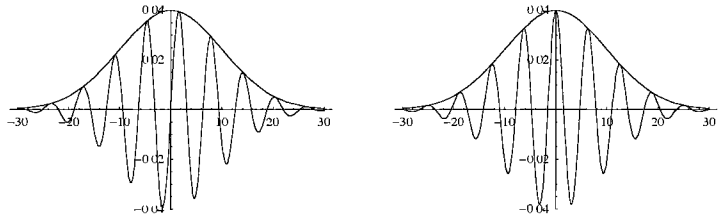


Figure 4.14 Gabor functions are sinusoidal functions with a Gaussian envelope. Left: Sin[x] G[x,10]; right: Sin[x+π/2] G[x,10].

Gabor functions can be made to look very similar by an appropriate choice of parameters:

```
σ = 1; gd[x_, σ_] = D[1 / Sqrt[2 π σ^2] Exp[-x^2 / (2 σ^2)], x];
Plot[{-1.2 gabor[x, 1.2], gd[x, 1]}, {x, -4, 4},
  PlotStyle → {Dashing[{0.02, 0.02}], RGBColor[0, 0, 0]},
PlotLegend → {"Gabor", "Gauss"},
  LegendPosition → {1.2, -0.3}, ImageSize -> 320];
```

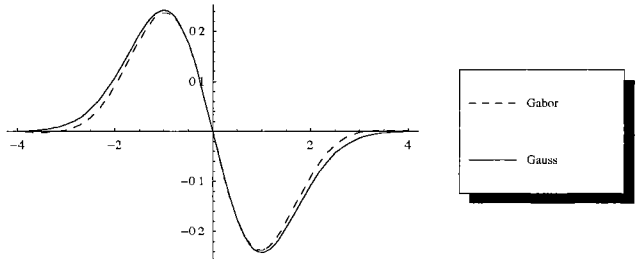


Figure 4.15 Gabor functions can be made very similar to Gaussian derivative kernels. In a practical application then there is no difference in result. Dotted graph: Gaussian first derivative kernel. Continuous graph: Minus the Gabor kernel with the same σ as the Gaussian kernel. Note the necessity of sign change due to the polarity of the sinusoidal function.

If we relax one or more of the first principles (leave one or more out, or add other axioms), we get other families of kernels. E.g. when we add the constraint that the kernel should be tuned to a specific spatial frequency, we get the family of *Gabor kernels* [Florack1992a, Florack1997a]. It was recently shown by Duits et al. [Duits2002a], extending the work of Pauwels [Pauwels1995], that giving up the constraint of separability gives a new family of interesting *Poisson* scale-space kernels, defined by the solution of the Dirichlet problem

$\frac{\partial L}{\partial s} = -(-\Delta)^{\alpha} L$. For $\alpha = 1$ we find the Gaussian scale-space, for $\alpha = \frac{1}{2}$ we get the Poisson scale-space. In this book we limit ourselves to the Gaussian kernel.

We conclude this section by the realization that the front-end visual system at the retinal level must be uncommitted, no feedback from higher levels is at stake, so the Gaussian kernel seems a good candidate to start observing with at this level. At higher levels this constraint is released.

The extensive feedback loops from the primary visual cortex to the LGN may give rise to 'geometry-driven diffusion' [TerHaarRomeny1994f], nonlinear scale-space theory, where the early differential geometric measurements through e.g. the simple cells may modify the kernels LGN levels. Nonlinear scale-space theory will be treated in chapter 21.

▲ Task 4.2 When we have noise in the signal to be differentiated, we have two counterbalancing effect when we change differential order and scale: for higher order the noise is amplified (the factor $(-i\omega)^n$ in the Fourier transform representation) and the noise is averaged out for larger scales. Give an explicit formula in our *Mathematica* framework for the propagation of noise when filtered with Gaussian derivatives. Start with the easiest case, i.e. pixel-uncorrelated (white) noise, and continue with correlated noise. See for a treatment of this subject the work by Blom et al. [Blom1993a].

▲ Task 4.3 Give an explicit formula in our *Mathematica* framework for the propagation of noise when filtered with a *compound function* of Gaussian derivatives, e.g. by the Laplacian $\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$. See for a treatment of this subject the work by Blom et al. [Blom1993a].

4.8 Higher dimensions and separability

Gaussian derivative kernels of higher dimensions are simply made by multiplication. Here again we see the separability of the Gaussian, i.e. this *is* the separability. The function **gd2D[x,y,n,m,σx,σy]** is an example of a Gaussian partial derivative function in 2D, first order derivative to x, second order derivative to y, at scale 2 (equal for x and y):

```
gd2D[x_, y_, n_, m_, σx_, σy_] := gd[x, n, σx] gd[y, m, σy];
Plot3D[gd2D[x, y, 1, 2, 2, 2], {x, -7, 7},
  {y, -7, 7}, AxesLabel -> {x, y, ""}, PlotPoints -> 40,
  PlotRange -> All, Boxed -> False, Axes -> True, ImageSize -> 190];
```

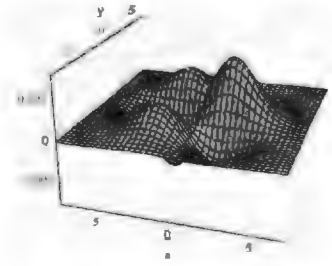


Figure 4.16 Plot of $\frac{\partial^3 G(x,y)}{\partial x \partial y^2}$. The two-dimensional Gaussian derivative function can be constructed as the product of two one-dimensional Gaussian derivative functions, and so for higher dimensions, due to the separability of the Gaussian kernel for higher dimensions.

The ratio $\frac{\sigma_x}{\sigma_y}$ is called the anisotropy ratio. When it is unity, we have an *isotropic* kernel, which diffuses in the x and y direction by the same amount. The Greek word 'isos' ($\iota\sigma o\varsigma$) means 'equal', the Greek word 'tropos' ($\tau\rho o\pi o\varsigma$) means 'direction' (the Greek word 'topos' ($\tau o\pi o\varsigma$) means 'location, place').

In 3D the iso-intensity surfaces of the Gaussian kernels are shown (and can be interactively manipulated) with the command MVContourPlot3D from the OpenGL viewer 'MathGL3D' by J.P.Kuska
(phong.informatik.uni-leipzig.de/~kuska/mathgl3dv3):

```
<< MathGL3d`OpenGLViewer`;

MVClear[]; σ = 1;

p1 = Table[MVContourPlot3D[Evaluate[D[E^(-(x^2+y^2+z^2)/(2 σ^2)), {x, n}]], {x, -6, 6},
   {y, -4, 4}, {z, -3, 0}, Contours → Range[-.6, .6, .1], PlotPoints → 60,
   BoxRatios → {2, 2, 1}, DisplayFunction → Identity], {n, 1, 3}];
Show[GraphicsArray[p1], ImageSize → 400];
```

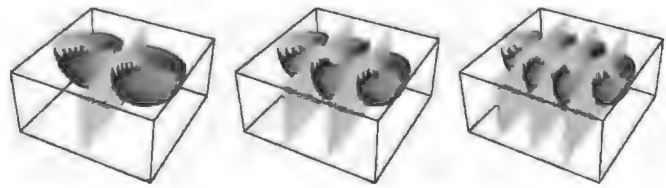


Figure 4.17 Iso-intensity surface for Gaussian derivative kernels in 3D. Left: $\frac{\partial G}{\partial x}$; middle: $\frac{\partial^2 G}{\partial x^2}$ right: $\frac{\partial^3 G}{\partial x^3}$.

The sum of 2 of the three 2nd order derivatives is called the 'hotdog' detector:

```
MVClear[]; σ = 1;
p1 = MVContourPlot3D[Evaluate[∂x,x E^-(x^2+y^2+z^2)/(2σ^2) + ∂z,z E^-(x^2+y^2+z^2)/(2σ^2)], {x, -6, 6},
  {y, -4, 4}, {z, -3, 0}, Contours → Range[-.6, .6, .1],
  PlotPoints → 60, BoxRatios → {2, 2, 1}, ImageSize -> 150];
```

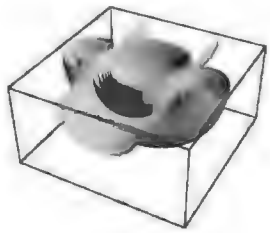


Figure 4.18 Iso-intensity surface for the Gaussian derivative in 3D $\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial z^2}$.

4.9 Summary of this chapter

The Gaussian derivatives are characterized by the product of a polynomial function, the Hermite polynomial, and a Gaussian kernel. The order of the Hermite polynomial is the same as the differential order of the Gaussian derivative. Many interesting recursive relations exist for Hermite polynomials, making them very suitable for analytical treatment. The shape of the Gaussian derivative kernel can be very similar to specific Gabor kernels. One essential difference is the number of zerocrossing: this is always infinite for Gabor kernels, the number of zerocrossings of Gaussian derivatives is equal to the differential order. The envelope of the Gaussian derivative amplitude is not the Gaussian function, as is the case for Gabor kernels.

The even orders are symmetric kernels, the odd orders are asymmetric kernels. The normalized zeroth order kernel has unit area by definition, the Gaussian derivative kernels of the normalized kernel have no unit area.

Gaussian derivatives are not orthogonal kernels. They become more and more correlated for higher order, if odd or even specimens are compared. The limiting case for infinite order leads to a sinusoidal (for the odd orders) or cosinusoidal (for the even orders) function with a Gaussian envelope, i.e. a Gabor function.

In the vision chapters we will encounter the Gaussian derivative functions as suitable and likely candidates for the receptive fields in the primary visual cortex.

5. Multi-scale derivatives: implementations

Three people were at work on a construction site. All were doing the same job, but when each was asked what the job was, the answers varied. "Breaking rocks," the first replied. "Earning my living," the second said. "Helping to build a cathedral," said the third.
-Peter Schultz

In order to get a good feeling for the interactive use of *Mathematica*, we discuss in this section three implementations of convolution with a Gaussian derivative kernel (in 2D) in detail:
1. implementation in the spatial domain with a 2D kernel;
2. through two sequential 1D kernel convolutions (exploiting the separability property);
3. implementation in the Fourier domain.
Just blurring is done through convolution with the zero order Gaussian derivative, i.e. the Gaussian kernel itself.

5.1 Implementation in the spatial domain

Mathematica 4 has a fast implementation of a convolution: **ListConvolve[kernel, list]** forms the convolution of the kernel **kernel** with **list**. This function is N-dimensional, and is internally optimized for speed. It can take any *Mathematica* expression, but its greatest speed is for **Real** (floating) numbers. We first define the 1D Gaussian function **gauss[x,σ]**:

```
<< FrontEndVision`FEV`;
Unprotect[gauss];
```

$$\texttt{gauss[x\_, σ\_ /; σ > 0] :=} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} ;$$

We explain in detail what happens here:
The function **gauss[x_, σ_]** is defined for the variables **x_** and **σ_**. The underscore **_** means that **x_** is a **Pattern** with the name **x**, it can be anything. This is one of the most powerful features in *Mathematica*: it allows pattern matching. In the appendix a number of examples are given. The variable **σ_** has the condition (indicated with **/;**) that **σ** should be positive. If this condition is not met, the function will not be evaluated. The function is defined with delayed assignment (**:=** in stead of **=** for direct assignment). In this way it will be evaluated only when it is called. The semicolon is the separator between statements, and in general prevents output to the screen, a handy feature when working on images.

The function **gDc[im,nx,ny,σ]** implements the same function in the spatial domain. The parameters are the same as above. This function is much faster, as it exploits the internal

function **ListConvolve**, and applies Gaussian derivative kernels with a width truncated to +/- 4 standard deviations, which of course can freely be changed.

```
gDc[im_, nx_, ny_, σ_ /; σ > 0] := Module[{x, y, kernel},
 kernel = N[Table[Evaluate[
       D[gauss[x, σ] * gauss[y, σ], {x, nx}, {y, ny}]],
      {y, -4 * σ, 4 * σ}, {x, -4 * σ, 4 * σ}]];
  ListConvolve[kernel, im, Ceiling[Dimensions[kernel] / 2]]];
```

Module[{vars}, ...] is a construct to make a block of code where the vars are shielded from the global variable environment. The derivative of the function **gauss[]** is taken with **D[f,{x,nx},{y,ny}]** where **nx** is the number of differentiations to **x** and **ny** the number of differentiations to **y**. The variable **kernel** is a **List**, generated by the **Table** command, which tabulates the function **gauss[]** over the range $\pm 4\sigma$ for both **x** and **y**. The derivative function must be evaluated with **Evaluate[]** before it can be tabulated. The function **N[]** makes the result a numerical value, a **Real** number.

ListConvolve is an optimized internal *Mathematica* command, that *cyclically* convolves the kernel **kernel** with the image **im**. The **Dimensions[]** of the kernel are a **List** containing the x- and y-dimension of the square kernel matrix. Finally, the upwards rounded (**Ceiling**) list of dimensions is used by **ListConvolve** to fix that the kernel starts at the first element of **im** and returns an output image with the same dimension as the input image.

```
im = Table[If[x^2 + y^2 < 7000, 100, 0], {x, -128, 127}, {y, -128, 127}];
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[#] & /@ {im, gDc[im, 1, 0, 1]}];
Show[GraphicsArray[p1], ImageSize -> 350];
```

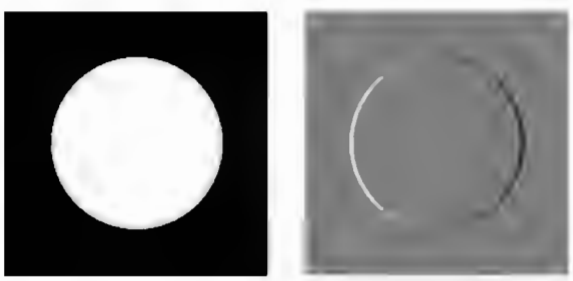


Figure 5.1 The derivative to x (right) at scale σ = 1 pixel on a 256^2 image of a circle (left).

The wider the kernel, the more points we include for calculation of the convolution, so the more computational burden we get. When the kernel becomes wider than half of the domain of the image, it becomes more efficient to apply the Fourier implementation discussed below. This trade-off has been worked out in detail by Florack [Florack2000a].

5.2 Separable implementation

The fastest implementation exploits the separability of the Gaussian kernel, and this implementation is mainly used in the sequel:

```
Options[gD] = {kernelSampleRange → {-6, 6}};
gD[im_List, nx_, ny_, σ_, (opts___)?OptionQ] :=
 Module[{x, y, kpleft, kpright, kx, ky, mid, tmp},
  {kpleft, kpright} = kernelSampleRange /. {opts} /. Options[gD];
  kx = N[Table[Evaluate[D[gauss[x, σ], {x, nx}]],
    {x, kpleft*σ, kpright*σ}]];
  ky =
    If[nx == ny, kx, N[Table[Evaluate[D[gauss[y, σ], {y, ny}]],
    {y, kpleft*σ, kpright*σ}]]]; mid = Ceiling[Length[#1]/2] &;
  tmp =
    Transpose[ListConvolve[{kx}, im, {{1, mid[kx]}, {1, mid[kx]}}]];
  Transpose[ListConvolve[{ky}, tmp, {{1, mid[ky]}, {1, mid[ky]}}]]];
```

The function **gD[im, nx, ny, σ, options]** implements first a convolution per row, then transposes the matrix of the image, and does the convolution on the rows again, thereby effectively convolving the columns of the original image. A second **Transpose** returns the image back to its original orientation. This is the default implementation of multi-scale Gaussian derivatives and will be used throughout his book.

```
im = Table[If[x^2 + y^2 < 7000, 100, 0], {x, -128, 127}, {y, -128, 127}];
Timing[imx = gD[im, 0, 1, 2]][[1]]
```

```
0.031 Second
```

```
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[#] & /@ {im, imx}];
Show[GraphicsArray[p1], ImageSize -> 260];
```

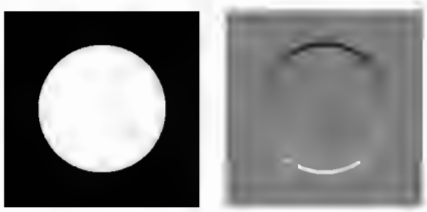


Figure 5.2 The derivative to y (right) at scale $\sigma = 2$ pixels on a 256^2 image of a circle (left).

▲ Task 5.1 Write a *Mathematica* function of the separable Gaussian derivative kernel implementation for 3D. Test the functionality on a 3D test image, e.g. a sphere.

5.3 Some examples

Convolving an image with a single point (a delta function) with the Gaussian derivative kernels, gives the kernels themselves., i.e. the pointspread function. E.g. here is the well known series of all Cartesian partial Gaussian derivatives to 5th order:

```
spike = Table[0., {128}, {128}]; spike[[64, 64]] = 1.;
Block[{$DisplayFunction = Identity},
  array = Table[Table[ListDensityPlot[gD[spike, m - n, n, 20],
      PlotLabel -> "∂x=" <> ToString[m - n] <> ", ∂y=" <> ToString[n]],
    {n, 0, m}], {m, 0, 5}]];
Show[GraphicsArray[array], ImageSize → 330];
```

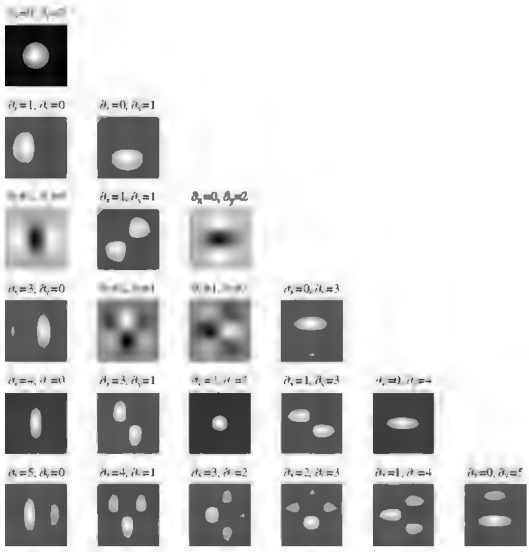


Figure 5.3 Gaussian partial derivative kernels up to 5th order.

$DisplayFunction is the internal variable that determines how things should be displayed. Its normal state (it default has the value **Display[$Display,#1]&**) is to send PostScript to the output cell. Its value is temporarily set to **Identity**, which means: no output. This is necessary to calculate but not display the plots.

We read an image with **Import** and only use the first element **[[1,1]]** of the returned structure as this contains the pixeldata.

```
im = Import["mr128.gif"][[1, 1]];
```

We start with just blurring at a scale of $\sigma = 3$ pixels and show the result as 2D image and 3D height plot:

```
DisplayTogetherArray[
  {ListDensityPlot[gD[im, 0, 0, 3]], ListPlot3D[gD[im, 0, 0, 3],
    Mesh -> False, BoxRatios → {1, 1, 1}]}, ImageSize → 500];
```

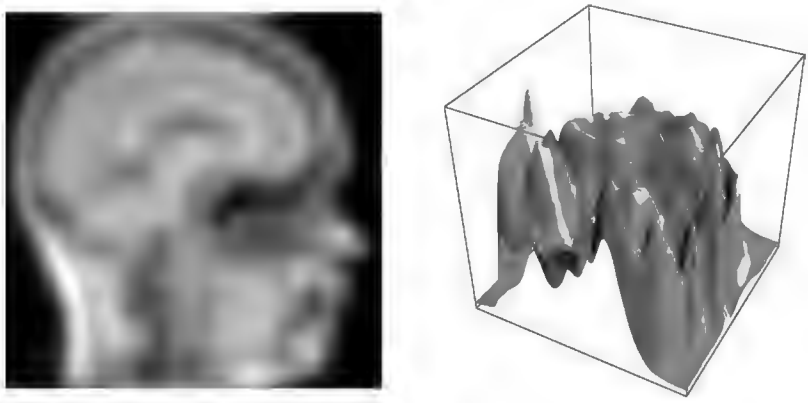


Figure 5.4 Left: a blurred MR image, resolution 128^2, $\sigma_{blur} = 3$ pixels. Right: The intensity surface as a height surface shows the blurring of the surfaces.

A movie of a (in this example) logarithmically sampled intensity scale-space is made with the **Table** command. Close the group of cells with images by double-clicking the group bracket. Double-clicking one of the resulting images starts the animation. Controls are on the bottom windowbar.

```
ss = Table[ListDensityPlot[gDf[im, 0, 0, E^τ], ImageSize -> 150],
  {τ, 0, 2.5, .25}];
```

Figure 5.5 Animation of a blurring sequence, with exponential scale parametrization. Double-click the image to start the animation (only in the electronic version). Controls appear at the lower window bar.

This animation is only available in the electronic version. Here are the images:

```
Show[GraphicsArray[Partition[ss, 5]], ImageSize -> 450];
```

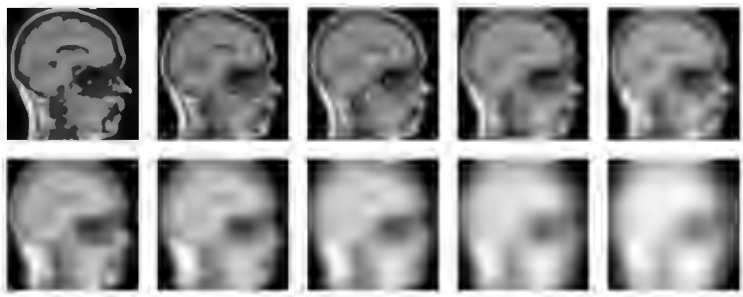


Figure 5.6 Frames of the animation of a blurring sequence above.

The sequence can be saved as an animated GIF movie (e.g. for use in webpages) with:

```
Export["c:\\scalespace.gif", ss, "GIF"];
```

The gradient of an image is defined as $\sqrt{L_x{}^2 + L_y{}^2}$. On a scale $\sigma = 0.5$ pixel for a 256^2 CT image of chronic cocaine abuse (EuroRAD teaching file case #1472, www.eurorad.org):

```
im = Import["Cocaine septum.gif"][[1, 1]];
DisplayTogetherArray[{ListDensityPlot[im],
   grad = ListDensityPlot[Sqrt[gD[im, 1, 0, .5]^2 + gD[im, 0, 1, .5]^2]]},
  ImageSize -> 370];
```

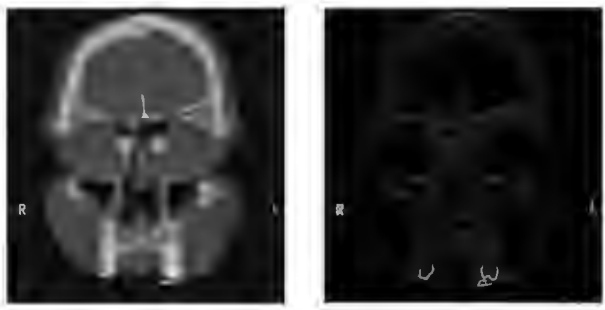


Figure 5.7 The gradient at a small scale σ = 0.5 pixels. Due to the letters R and L in the image with steep gradients the gradient image is not properly scaled in intensity. Note the completely missing septum in this patient (From www.eurorad.org, EuroRAD authors: D. De Vuyst, A.M. De Schepper, P.M. Parizel, 2002).

To change the window/level (contrast/brightness) settings one can change the displayed range of intensity values:

```
Show[grad, PlotRange → {0, 20},
  DisplayFunction -> $DisplayFunction, ImageSize -> 150];
```

Figure 5.8 The gradient at a small scale σ = 0.5 pixels, now with an intensity window of 0 (black) to 30 (white).

We can also transfer the image into its histogram equalized version, by substituting its grayvalues by the values given by its cumulative lookup table:

```
Unprotect[heq];
heq[im_List] := Module[{min, max, freq, cf, lcf, maxcf, lut, int},
   min = Min[im]; max = Max[im];
   freq = BinCounts[Flatten[im], {min, max, (max - min) / 256}];
   cf = FoldList[Plus, First[freq], Drop[freq, 1]];
  maxcf = Max[cf]; lcf = Length[cf];
  lut = Table[N[{(i - 1) / lcf, cf[[i]] / maxcf}], {i, 1, lcf}];
  lut[[lcf]] = {1., 1.};
   int = Interpolation[lut]; max int[(im - min) / (max - min)]];

ListDensityPlot[
  heq[√(gD[im, 1, 0, .5]^2 + gD[im, 0, 1, .5]^2), ImageSize -> 150];
```

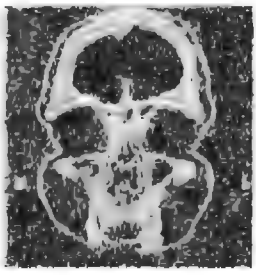


Figure 5.9 Histogram equalization of the gradient image of figure 5.7. By many radiologists this is considered too much enhancement. 'Clipped' adaptive histogram equalization admits different levels of enhancement tuning [Pizer1987].

The cumulative lookup table is applied for the intensity transform. Small contrasts have been stretched to larger contrasts, and reverse. We next compare the histograms of the gradient image with the histogram of the histogram-equalized gradient image. The total histogram of this image is indeed reasonably flat now.

```
grad = √(gD[im, 1, 0, .5]^2 + gD[im, 0, 1, .5]^2) ; DisplayTogetherArray[
 Histogram[Flatten[#]] & /@ {grad, heq[grad]}, ImageSize → 380];
```

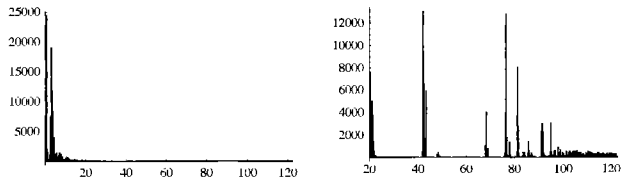


Figure 5.10 Left: Histogram of the gradient image of figure 5.7. Right: Histogram of the histogram-equalized gradient image. Note the equalizing or marked stretching of the histogram.

To conclude this introduction to multi-scale derivatives, let us look at some edges detected at different scales. It is clear from the examples below that the larger scale edges denote the more 'important' edges, describing the coarser, hierarchically higher structure:

```
im = Import["Utrecht256.gif"][[1, 1]];
DisplayTogetherArray[
 ListDensityPlot[√(gD[im, 1, 0, #]^2 + gD[im, 0, 1, #]^2) ] & /@ {.5, 2, 5},
 ImageSize -> 400];
```

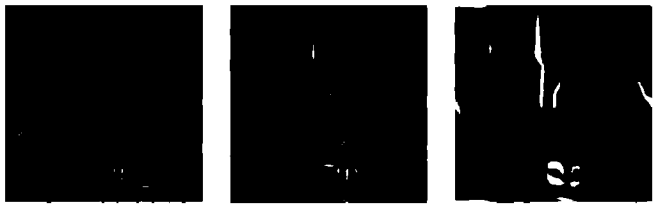


Figure 5.11 Gradient edges detected at different scales (σ = 0.5, 2, 5 pixels resp.). The coarser edges (right) indicate hierarchically more 'important' edges.

Other sources of different scales for edges are shadows and diffuse boundaries [Elder1996].

5.4 N-dim Gaussian derivative operator implementation

One of the powerful capabilities of *Mathematica* as a programming language is the relative ease to write numerical functions on N-dimensional data. In scale-space theory often high dimensional data occur: 3D and 3D-time medical images, such as 3D cardiovascular time sequences, orientation bundles (see chapter 16 where an extra dimension emerges from the inclusion of orientation as the output of *measurements* by oriented filters), high dimensional feature spaces for texture analysis, etc. Here is the separable implementation for N-dimensions:

```
Unprotect[gDn]; gDn[im_, orderlist_, σlist_, opts___?OptionQ] :=
 Module[{gaussd, dim = Length[Dimensions[im]], out = N[im], l, r, gder, x,
   kernel, cat, mid, lc, tl, td}, td = Dimensions/@{orderlist, σlist};
  tl = Length/@td; {l, r} = kernelSampleRange/. {opts} /. Options[gD];
  gaussd = 1/(#2 Sqrt[2 π]) (-1/(#2 Sqrt[2]))^#1 HermiteH[#1, x/(#2 Sqrt[2])] e^(-x^2/(2 #2^2)) &;
  gder = Table[N[gaussd[#1, #2]], {x, Floor[l #2], Ceiling[r #2]}] &;
  kernel = RotateRight[MapThread[gder, {orderlist, σlist}]];
  mid = (Ceiling[Length[#1] / 2] &) /@ kernel;
  cnt = Append[Table[l, {dim - 1}], mid[[#1]]] &;
  lc =
   Transpose[ListConvolve[Nest[List, kernel[[#2]], dim - 1], #1, {cnt[#2], cnt[#2]}],
     RotateRight[Range[dim]]] &; Do[out = lc[out, i], {i, dim}]; out]
```

The function makes use of the possibility to **Nest** functions to large depth, and the universality of the **ListConvolve** function. The function is fast. Note the specification of orders and scales as lists, and note the specific, *Mathematica*-intrinsic ordering with the fastest running variable last: {z,y,x}.

Example: **gDn[im,{0,2,1},{2,2,2}]** calculates $\frac{\partial^3 L}{\partial x \partial y^2}$ of the input image **im** at an isotropic scale of $\sigma_z = \sigma_y = \sigma_x = 2$ pixels.

Here is the time it takes to calculate the first order derivative in 3 directions at scales of 1 pixel of a 128^3 random array (more than 2 million pixels, 1.7 GHz, 512 MB, Windows XP):

```
im = Table[Random[], {128}, {128}, {128}];
Timing[gDn[im, {1, 1, 1}, {1, 1, 1}]] // First
```

```
5.094 Second
```

This gives help on how to call the function:

```
? gDn
```

```
gDn[im,{...,ny,nx},{...,σy,σx},options] calculates the Gaussian
  derivative of an N-dimensional image by approximated spatial
  convolution. It is optimized for speed by 1D convolutions per
  dimension. The image is considered cyclic in each direction.
  Note the order of the dimensions in the parameter lists.
     im = N-dimensional input image [List]
     nx = order of differentiation to x [Integer, nx ≥ 0]
     σx = scale in x-dimension [in pixels, σ > 0]
     options = <optional> kernelSampleRange: range of kernel
  sampled in multiples of σ. Default: kernelSampleRange->{-6,6}

 Example: gDn[im,{0,0,1},{2,2,2}] calculates the x-
  derivative of a 3D image at an isotropic scale of σz=σy=σx=2.
```

5.5 Implementation in the Fourier domain

The spatial convolutions are not exact. The Gaussian kernel is truncated. In this section we discuss the implementation of the convolution operation in the Fourier domain.

In appendix B we have seen that a convolution of two functions in the spatial domain is a multiplication of the Fourier transforms of the functions in the Fourier domain, and take the inverse Fourier transform to come back to the spatial domain. We recall the processing scheme (e.g. with 1D functions):

$$\begin{array}{ccccc} f(x) & = & h(x) & \otimes & g(x) \\ \updownarrow & & \updownarrow & & \updownarrow \\ F(\omega) & = & H(\omega) & \cdot & G(\omega) \end{array}$$

The $\updownarrow$ indicates the Fourier transform in the downwards direction and the inverse Fourier transform in the upwards direction. $f(x)$ is the convolved function, $h(x)$ the input function, and $g(x)$ the convolution kernel.

The function **gDf[im,nx,ny,σ]** implements the convolution of the 2D image with the Gaussian derivative for 2D discrete data in the Fourier domain. This is an exact function, no approximations other than the finite periodic window in both the x- and y-direction. We explicitly give the code of the functions here, so you see *how* it is implemented, the reader may make modifications as required. All information on (always capitalized) internal functions is on board of the *Mathematica* program in the Help Browser (highlight+key F1), as well as on the 'the Mathematica book' internet pages of Wolfram Inc.

Variables: **im** = 2D image (as a **List** structure)
nx, **ny** = order of differentiation to **x** resp. **y**
σ = scale of the Gaussian derivative kernel, in pixels

The underscore in e.g. **im_** means **Blank[im]** and stands for *anything* (a single element) which we name **im**. **im_List** means that **im** is tested if it is a **List**. If not, the function **gDf** will not be evaluated.

```
Unprotect[gDf]; Remove[gDf];

gDf[im_List, nx_, ny_, σ_] :=
  Module[{xres, yres, gdkernel},
          {yres, xres} = Dimensions[im];
    gdkernel =
    N[Table[Evaluate[D[1/(2 π σ^2) Exp[-(x^2 + y^2)/(2 σ^2)], {x, nx}, {y, ny}]], {y,
       -(yres - 1) / 2, (yres - 1) / 2}, {x, -(xres - 1) / 2, (xres - 1) / 2}]];
     Chop[N[Sqrt[xres yres] InverseFourier[Fourier[im]
     Fourier[RotateLeft[gdkernel, {yres / 2, xres / 2}]]]]]];
```

A **Module[{vars}, ...code...]** is a scope construct, where the vars are private variables. The last line determines what is returned. The assignment with **:=** is a delayed assignment, i.e. the function is only evaluated when called. The dimensions of the input image are extracted (note the order!) and the Gaussian kernel is differentiated with the function **D[gauss,{x,nx},{y,ny}]** and symmetrically tabulated over the x- and y-dimensions to get a kernel image with the same dimensions as the input image.

We have now a 2D **List** with the kernel in the center. We shift **gdkernel** with **RotateLeft** over half the dimensions in x- and y-direction in order to put the kernel's center at the origin at {0,0}. We could equally have shifted in this symmetric case with **RotateRight**. We then take the **Fourier** transform of both the image and the kernel, multiply them (indicated by a space) and take the **InverseFourier** transform.

Because we have a finite Fourier transform, we normalize over the domain through the factor $\sqrt{\texttt{xres yres}}$. The function **N[]** makes all output numerical, and the function **Chop[]** removes everything that is smaller then 10^{-10}, so to remove very small round-off errors.

```
im = Import["mr256.gif"][[1, 1]]; imx = gDf[im, 1, 0, 1];
ListDensityPlot[imx, ImageSize -> 240];
```

Figure 5.12 First order Gaussian derivative with respect to x at scale σ = 1 pixel, calculated through the Fourier domain. Resolution 256^2 pixels.

The *Mathematica* function Fourier is highly optimized for any size of the data, and uses sophisticated bases when the number of pixels is not a power of 2.

This function is somewhat slower that the spatial implementation, but is exact. Here is a vertical edge with a lot of additive uniform noise. The edge detection at very small scale only reveals the 'edges of the noise'. Only at the larger scales we discern the true edge, i.e. when the scale of the operator applied is at 'the scale of the edge'.

```
im5 = Table[If[x > 128, 1, 0] + 13 Random[], {y, 256}, {x, 256}];
DisplayTogetherArray[
 Prepend[ListDensityPlot[Sqrt[gDf[im5, 1, 0, #]^2 + gDf[im5, 0, 1, #]^2]] & /@
   {2, 6, 12}, ListDensityPlot[im5]], ImageSize -> 500];
```

Figure 5.13 Detection of a very low contrast step-edge in noise. Left: original image, the step-edge is barely visible. At small scales (second image, σ = 2 pixels) the edge is not detected. We see the edges of the noise itself, cluttering the edge of the step-edge. Only at large scale (right, σ = 12 pixels) the edge is clearly found. At this scale the large scale structure of the edge emerges from the small scale structure of the noise.

▲ Task 5.2 The Fourier implementation takes the Fourier transform of the image and the Fourier transform of a calculated kernel. This seems a waste of calculating time, as we know the *analytical* expression for the Fourier transform of the Gaussian kernel. Write a new *Mathematica* function that takes this into account, and check if there is a real speed increase.

▲ Task 5.3 The spatial implementation has different speed for different size kernels. With increasing kernel size the number of operations increases substantially. How?

▲ Task 5.4 Compare for what kernel size the choice of implementation is computationally more effective: Fourier or spatial domain implementation. See also [Florack 2000a].

There are two concerns we discuss next: what to do at the boundaries? And: the function is slow, so how to speed it up?

5.6 Boundaries

```
DisplayTogetherArray[
  Show /@ Import /@ {"Magritte painting boundary.gif", "Magritte.jpg"},
  ImageSize -> 340];
```

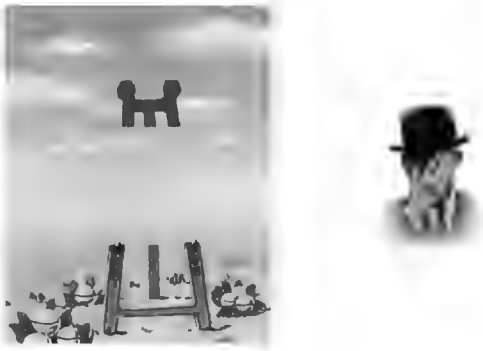


Figure 5.14 It is important to consider what happens at the boundaries of images. It matters what we *model* outside our image. Painting by René Magritte (right: self-portrait, 1898-1967).

At the boundary of the image artefacts may appear when we do convolutions with (by nature) extended kernels. Here is an example: two linear intensity ramps give a constant output when we calculate the first derivative to x, but we see both at the left- and right-hand side strong edge responses, for the Fourier implementation as well as for the spatial implementation:

```
im = Table[If[y > 64, x - 64, 64 - x], {y, 128}, {x, 128}];
DisplayTogetherArray[ListDensityPlot[#] & /@
   {im, gDf[im, 1, 0, 3], gD[im, 1, 0, 3]}, ImageSize -> 400];
```

Figure 5.15 Boundary effects due to the periodicity of the Fourier domain.

This is due to the fact that both in the Fourier domain as the spatial domain implementation of the convolution function the image is regarded as repetitive. A Fourier transform is a *cyclic* function, i.e. $\mathcal{F}(\omega) = \mathcal{F}(\omega + n\,2\pi)$. In 2D: $\mathcal{F}(\omega_x, \omega_y) = \mathcal{F}(\omega_x + n_x\,2\pi, \omega + n_y\,2\pi)$. The boundary effects in the image above are due to the strong edge created by the neighboring pixels at both ends. One can regard the domain of the image as a window cut-out from an infinite tiling of the plane with 2D functions. Figure 4.1 shows a tiling with 20 images, each 64^2 pixels:

```
im = Import["mr64.gif"][[1, 1]];
tiles = Join @@ Table[MapThread[Join, Table[im, {5}]], {4}];
ListDensityPlot[tiles, ImageSize -> 280];
```

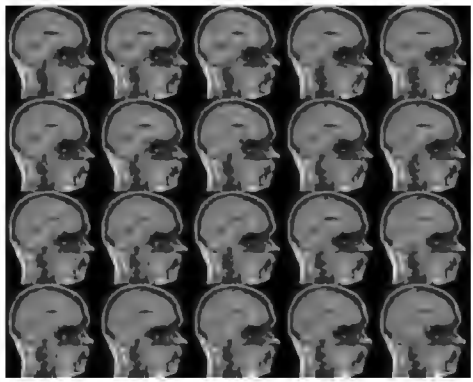


Figure 5.16 A section from the infinite tiling of images when we consider a cyclic operation. The *Mathematica* function `MapThread` maps the function `Join` on the rows of the horizontal row of 5 images to concatenate them into long rows, the function `Join` is then applied (with `Apply` or `@@`) on a table of 4 such resulting long rows to concatenate them into a long vertical image.

```
Clear[a, b, c, d, e, f, h];
MapThread[h, {{a, b, c}, {d, e, f}}]
```

```
{h[a, d], h[b, e], h[c, f]}
```

```
Apply[h, {{a, b, c}, {d, e, f}}]
```

```
h[{a, b, c}, {d, e, f}]
```

```
h @@ {{a, b, c}, {d, e, f}}
```

```
h[{a, b, c}, {d, e, f}]
```

It is important to realize that there is no way out to deal with the boundaries. Convolution is an operation with an *extended* kernel, so at boundaries there is always a choice to be made. The most common decision is on repetitive tiling of the domain to infinity, but other choices are just as valid. One could extend the image with zero's, or mirror the neighboring image at all sides in order to minimize the edge artefacts. In all cases information is put at places where there was *no* original observation. This is no problem, as long as we carefully describe how our choice has been made. Here is an example of mirrored tiling:

```
im = Import["mr128.gif"][[1, 1]]; Off[General::spell];
imv = Reverse[im]; imh = Reverse /@ im; imhv = Reverse /@ Reverse[im];
mirrored = Join @@ (MapThread[Join, #] & /@ ({{imhv, imv, imhv}, {imh, im, imh}, {imhv, imv, imhv}}));
ListDensityPlot[mirrored, ImageSize -> 270];
```

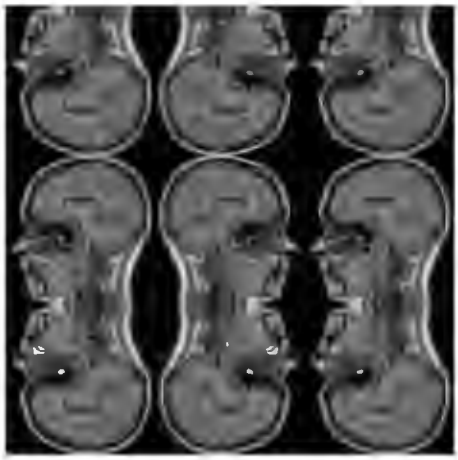


Figure 5.17 A section from the infinite tiling of images when we consider a mirroring operation. Note the rather complex mirroring and concatenation routines for these 2D images.

▲ Task 5.5 Rachid Deriche [Deriche 1992] describes a fast *recursive* implementation of the Gaussian kernel and its derivatives. Make a *Mathematica* routine for recursive implementation.

▲ Task 5.6 A small truncated kernel size involves less computations, and is thus faster. Blurring with a large kernel can also be accomplished by a concatenation of small kernels, e.g. a blurring step with $\sigma = 3$ pixels followed by a blurring step with $\sigma = 4$ pixels gives the same result as a single blurring step with $\sigma = 5$ pixels ($\sigma_1^2 + \sigma_2^2 = \sigma_{new}^2$). What is faster, a large kernel, or a cascade series of smaller kernels? Where is the trade-off?

5.7 Advanced topic: speed concerns in *Mathematica*

This section can be skipped at first reading.

Mathematica is an interpreter, working with symbolic elements, and arbitrary precision. For this reason, care must be taken that computation times do not explode for large datasets. When proper measures are taken, *Mathematica* can be fast, close to compiled C++ code. In

this section we discuss some examples of increasing the speed of the operations on larger datasets.

It pays off to work numerically, and to *compile* a function when it is a repetitive operation of simple functions. *Mathematica*'s internal commands are optimized for speed, so the gain here will be less. We discuss the issue with the example of the generation of a discrete Gaussian derivative kernel. The timings given are for a 1.7 GHz 512 MB PC and *Mathematica* 4.1 under Windows XP.

First of all, exact calculations are slow. Most internal *Mathematica* functions can work both with symbolic and numerical data. These internal functions are fully optimized with respect to speed and memory resources for numerical input. Here is a simple example:

```
σ = 1; m = Table[Exp[-(x^2 + y^2)/(2 σ^2)], {x, -4, 4}, {y, -4, 4}];
Timing[Eigenvalues[m]]
Timing[Eigenvalues[N[m]]]
Timing[Chop[Eigenvalues[N[m]]]]
```

$$\{3.156\ \text{Second},\ \{0, 0, 0, 0, 0, 0, 0, 0, 1 + \frac{2}{e^{16}} + \frac{2}{e^{9}} + \frac{2}{e^{4}} + \frac{2}{e}\}\}$$

```
{0. Second, {1.77264, 5.59373 \ 10^-17, -4.27232 \ 10^-17, -1.82978 × 10^-18,
  3.22688 × 10^-22, -6.5072 × 10^-24, -7.47864 × 10^-34, 1.05492 \ 10^-35, 0.}}

{0. Second, {1.77264, 0, 0, 0, 0, 0, 0, 0, 0}}
```

In the sequel we will develop a very fast implementation for the convolution of a 2D image with a Gaussian derivative in the Fourier domain (see section 4.3). Most of the time is spent in the creation of the 2D Gaussian kernel, e.g. for 256^2:

```
{xres, yres} = {256, 256}; σ = 3;
Timing[
  kernel = Table[1/(2 π σ^2) Exp[-(x^2 + y^2)/(2 σ^2)], {y, -(yres - 1) / 2, (yres - 1) / 2},
    {x, -(xres - 1) / 2, (xres - 1) / 2}]][[1]]
```

```
4.859 Second
```

Mathematica keeps values as long as possible in an exact representation. Here is the pixelvalue at (30, 47):

```
kernel[[30, 47]]
```

$$\frac{1}{18\ e^{32689/36}\ \pi}$$

An additional disadvantage is that the Fourier transform on such symbolic expressions also takes a long time:

```
Timing[fft = Fourier[kernel]] // First
```

```
8. Second
```

It doesn't make much difference when we enter the data as **Real** values (to be done with the insertion of a decimal point in a number, or through the function **N**):

```
{xres, yres} = {256., 256.}; σ = 3.; pi = N[π];
Timing[
  gdkernel = Table[1/(2 π σ^2) Exp[-(x^2 + y^2)/(2 σ^2)], {y, -(yres - 1) / 2, (yres - 1) / 2},
    {x, -(xres - 1) / 2, (xres - 1) / 2}]][[1]]
```

```
5.797 Second
```

The output is now a number, not a symbolic expression:

```
gdkernel[[30, 48]]
```

$$6.379323933059 \times 10^{-393}$$

But still, we have no gain in speed. This is because the internal representation is still in 'arbitrary precision' mode. The smallest and largest number that can be represented as a **Real** is:

```
$MinMachineNumber
$MaxMachineNumber
```

$$2.22507 \times 10^{-308}$$

$$1.79769 \times 10^{308}$$

We have smaller values in our pixels! As soon as *Mathematica* encounters a number smaller or larger then the dynamic range for **Real** numbers, it turns into arbitrary precision mode, which is slow. A good improvement in speed is therefore gained through restricting the output to be in this dynamic range. In our example the parameter for the exponential function **Exp** should be constrained:

```
{xres, yres} = {256., 256.}; σ = 3.; pi = N[π];
Timing[gdkernel = Table[1/(2 π σ^2) Exp[If[-(x^2 + y^2)/(2 σ^2) < -100, -100, -(x^2 + y^2)/(2 σ^2)]],
    {y, -(yres - 1) / 2, (yres - 1) / 2},
    {x, -(xres - 1) / 2, (xres - 1) / 2}]] // First
```

```
2.594 Second
```

Most of the internal commands of *Mathematica* do a very good job on real numbers.

A further substantial improvement in speed can be obtained by *compilation* of the code into fast internal assembler code with the function **Compile[{args}, ...code..., {decl}]**. This generates a pure function, that can be called with the arguments **{args}**. This function generates optimized code based on an idealized register machine. It assumes approximate real or inter numbers, or matrices of these. The arguments in the argumentlist need to have the proper assignment (**_Real**, **_Integer**, **_Complex** or **True/False**).

The assignment `_Real` is default and may be omitted, so `{x, _Real}` is equivalent to `{x}`. An example to calculate the factorial of a sum of two real numbers:

```
gammasum = Compile[{{x, _Real}, {y, _Real}}, (x + y) !]

CompiledFunction[{x, y}, (x + y) !, -CompiledCode-]

gammasum[3, 5]

40320.
```

We now check if the compiled code of our Gaussian kernel gives a speed improvement:

```
gdkernel = Compile[{xres, yres, σ}, xresh = (xres - 1)/2; yresh = (yres - 1)/2;
   p = Table[1/(2 π σ^2) Exp[If[-(x^2 + y^2)/(2 σ^2) < -100, -100, -(x^2 + y^2)/(2 σ^2)]],
     {y, -yresh, yresh}, {x, -xresh, xresh}], {{x, _Real},
    {y, _Real}, {xresh, _Real}, {yresh, _Real}, {p, _Real, 2}}];

Timing[gdkernel[256, 256, 3]] // First

2.532 Second
```

In version 4.2 of *Mathematica* we see no improvement, running the example above, the kernel has been optimized for these calculations. In earlier versions you will encounter some 60% improvement with the strategy above. See the Help browser (shift-F1) for speed examples of the Compile function. We now add the symbolic operation of taking derivatives of the kernel. We force direct generation of the polynomials in the Gaussian derivatives with the Hermite polynomials, generated with `HermiteH`. The symbolic functions are first evaluated through the use of the function `Evaluate`, then compiled code is made:

```
gdkernel = Compile[{xres, yres, σ, {nx, _Integer}, {ny, _Integer}},
   xresh = (xres - 1)/2; yresh = (yres - 1)/2;
   p = Table[Evaluate[
      (-1/(σ Sqrt[2]))^(nx+ny) HermiteH[nx, x/(σ Sqrt[2])] HermiteH[ny, y/(σ Sqrt[2])]
       1/(σ^2 2 π) Exp[If[-(x^2 + y^2)/(2 σ^2) < -100, -100, -(x^2 + y^2)/(2 σ^2)]]],
     {y, -yresh, yresh}, {x, -xresh, xresh}], {{x, _Real},
    {y, _Real}, {xresh, _Real}, {yresh, _Real}, {p, _Real, 2}}];

Timing[gdkernel[256, 256, 3, 10, 10]] // First

4.25 Second
```

Larger kernels are now no problem anymore, e.g. for 512^2:

```
Timing[t = gdkernel[512, 512, 3, 2, 1]] // First
```

```
7.593 Second
```

We adopt this function for our final implementation. Because the output is a matrix of real numbers, also the Fourier transform is very fast. This is the time needed for the Fast Fourier Transform on the 512^2 kernel just generated:

```
Timing[Fourier[t]] // First
```

```
0.172 Second
```

To complete this section we present the final implementation, available throughout the book in the context **FEV`**, which is loaded by default in each chapter.

In the compiled function also complex arrays emerge, such as the result of **Fourier[]** and **InverseFourier[]**. The compiler is told by the declarations at the end that anything with the name **Fourier** and **InverseFourier** working on something (_) should be stored in a complex array with tensorrank 2, i.e. a 2D array. Study the rest of the details of the implementation yourself:

```
Unprotect[gDf]; gDf[im_, nx_, ny_, σ_] :=
 Module[{}, {xres, yres} = Dimensions[im]; gf[im, nx, ny, σ, xres, yres]];

gf =
  Compile[{{im, _Real, 2}, {nx, _Integer}, {ny, _Integer}, σ, xres, yres},
   Module[{x, y}, xresh = (xres - 1)/2; yresh = (yres - 1)/2;
    p = RotateLeft[Table[1/(2 π σ^2) Evaluate[(-1/(σ Sqrt[2]))^(nx+ny)
          HermiteH[nx, x/(σ Sqrt[2])] HermiteH[ny, y/(σ Sqrt[2])]
          E^If[-(x^2+y^2)/(2 σ^2) < -200, -200, -(x^2+y^2)/(2 σ^2)]],
        {y, -yresh, yresh}, {x, -xresh, xresh}], {xres/2, yres/2}}]];
   Sqrt[xres yres] Chop[Re[InverseFourier[Fourier[im] Fourier[p]]]],
   {{x, _Real}, {y, _Real}, {xresh, _Real}, {yresh, _Real}, {p, _Real, 2},
    {Fourier[_], _Complex, 2}, {InverseFourier[_], _Complex, 2}}];
```

5.8 Summary of this chapter

Mathematica is fast when:
- it can use its internal kernel routines as much as possible. They have been optimized for speed and memory use;
- it can calculate on numerical data. Use the function **N[...]** to convert infinite precision representations like **Sin[3/7]** to numerical data;
- it is working in the representation range of real numbers. Otherwise it enters the infinite precision mode again;
- the function is compiled with the function **Compile[...]**;

6. Differential structure of images

"*If I had more time, I would have written you a shorter letter*", Pascal (1623-1662)

6.1 The differential structure of images

In this chapter we will study the differential structure of discrete images in detail. This is the structure described by the local multi-scale derivatives of the image. We start with the development of a toolkit for the definitions of heightlines, local coordinate systems and independence of our choice of coordinates.

```
<< FrontEndVision`FEV`; Off[General::spell];
Show[Import["Spiral CT abdomen.jpg"], ImageSize -> 170];
```

Figure 6.1 An example of a need for segmentation: 3D rendering of a spiral CT acquisition of the abdomen of a patient with Leriche's syndrome (EuroRAD case #745, authors R. Brillo, A. Napoli, S. Vagnarelli, M. Vendola, M. Benedetti Valentini, 2000, www.eurorad.org).

We will use the tools of *differential geometry*, a field designed for the structural description of space and the lines, curves, surfaces etc. (a collection known as *manifolds)* that live there.

We develop strategies for the generation of formulas for the detection of particular *features*, that detect special, semantically circumscribed, local meaningful structures (or properties) in the image. Examples are edges, corners, T-junctions, monkey-saddles and many more. We develop operational detectors in *Mathematica* for all features described.

One can discriminate local and multi-local methods in image analysis. We specifically discuss here local methods, at a particular local neighborhood (pixel). In later chapters we look at multi-local methods, and enter the realm of how to connect local features, both by studying similarity in properties with neighboring pixels ('perceptual grouping'), relations

over scale ('deep structure') and relations given by a particular *model*. We will discuss the use of the local features developed in this chapter into 'geometric reasoning'.

Why do we need to go in detail about local image derivatives? Combinations of derivatives into expressions give nice feature detectors in images. It is well known that $\sqrt{\left(\frac{\partial L}{\partial x}\right)^2 + \left(\frac{\partial L}{\partial y}\right)^2}$ is a good edge detector, and $\left(\frac{\partial L}{\partial y}\right)^2 \frac{\partial^2 L}{\partial x^2} - 2 \frac{\partial L}{\partial x} \frac{\partial L}{\partial y} \frac{\partial^2 L}{\partial x \partial y} + \left(\frac{\partial L}{\partial x}\right)^2 \frac{\partial^2 L}{\partial y^2}$ is a good corner detector. But how do we come to such formula's? We can make an infinite number of such expressions. What constraints can/should we impose to come to a reasonably small set of *basis* descriptors? Is there such a *basis*? It turns out there is, and in this chapter we will derive a formal complete set of such descriptive elements.

A very important constraint in the development of tools for the description of image structure is to be independent of the choice of coordinates. We will discuss coordinate transformations, like translations, rotations, zooming, in order to find a way to detect features *invariant* to such coordinate transformations. In fact, we will discuss three 'languages' in which it is easy to develop a general strategy to come up with quite complex image structure detectors:

gauge coordinates, Cartesian tensors, and algebraic polynomial invariants. All these methods have firm roots in mathematics, specifically differential geometry, and form an ideal substrate for the true understanding of image structure.

We denote the function that describes our landscape (the image) with $L(x, y)$ throughout this book, where L is the physical property measured in the image. Examples of L are luminance, T1 or T2 relaxation time (for MRI images), linear X-ray absorption coefficient (for CT images), depth (for range images) etc. In fact, it can be any scalar value. The coordinates x, y are discrete in our case, and denote the locations of the pixel. If the image is 3-dimensional, e.g. a stack of images from an MRI or CT scanner, we write $L(x, y, z)$. A scale-space of images, observed at a range of scales σ is written as $L(x, y; \sigma)$. We write a semicolon as separator to highlight the fact that σ is *not* just another spatial variable. If images are a function of time as well, we write e.g. $L(x, y, z; t)$ where t is the time parameter. In chapter 17 we will develop scale-space theory for images sampled over time. In chapter 15 we study the extra dimension of color in images and derive differential features in color-space, and in chapter 13 we derive methods for the extraction of motion, a vectorial property with a magnitude and a direction. We firstly focus on static, spatial images.

6.2 Isophotes and flowlines

Lines in the image connecting points of equal intensity are called *isophotes*. They are the heightlines of the intensity landscape when we consider the intensity as 'height'. Isophotes in 2D images are curves, and in 3D surfaces, connecting points with equal luminance.

(Greek: isos ($\iota\sigma o\varsigma$) = equal, photos ($\phi o\tau o\varsigma$) = light): $L(x, y) = \text{constant}$ or $L(x, y, z) = \text{constant}$. This definition however is for a continuous function. But the scale-space paradigm solves this: in discrete images isophotes exist because these are *observed*

images, and thus *continuous* (which means: infinitely differentiable, or C^{∞}). Lines of constant value in 2D are **Contours** in *Mathematica*, which can be plotted with **ContourPlot**. Figure 6.2 illustrates this for a blurred version of a 2D image.

```
im = Import["mr128.gif"][[1, 1]];
Block[{$DisplayFunction = Identity, dp, cp},
 dp = ListDensityPlot[gD[im, 0, 0, #]] & /@ {1, 2, 3};
 cp = ListContourPlot[gD[im, 0, 0, #],
      ContourStyle → List /@ Hue /@ (.1 Range[10])] & /@ {1, 2, 3};
 pa = MapThread[Show, {dp, cp}]]; Show[GraphicsArray[pa],
 ImageSize -> 400];
```

Figure 6.2 Isophotes of an image at various blurring scales: from left to right: $\sigma = 1$, $\sigma = 2$ and $\sigma = 3$ pixels. Image resolution 128^2. Ten isophotes are plotted in each image, equidistant over the available intensity range. Each is shown in a different color, superimposed over the grayvalues. Notice that the isophotes get more 'rounded' when we blur the image. When we consider the intensity distribution of a 2D image as a landscape, where the height is given by the intensity, isophotes are the heightlines.

Isophotes are important elements of an image. In principle, all isophotes together contain the same information as the image itself. The famous and often surprisingly good working segmentation method by thresholding and separating the image in pixels lying within or without the isophote at the threshold luminance is an example of an important application of isophotes. Isophotes have the following properties:

- isophotes are closed curves. Most (but not all, see below) isophotes in 2D images are a so-called Jordan curve: a non-self-intersecting planar curve topologically equivalent to a circle;
- isophotes can intersect themselves. These are the critical isophotes. These always go through a saddlepoint;
- isophotes do not intersect other isophotes;
- any planar curve is completely described by its *curvature*, and so are isophotes. We will define and derive the expression for isophote curvature in the next section.

- isophote shape is independent of grayscale transformations, such as changing the contrast or brightness of an image.

A special class of isophotes is formed by those isophotes that go through a *singularity* in the intensity landscape, thus through a minimum, maximum or saddle point. At these places the intensity landscape is horizontal, the local spatial derivatives are all zero. Only at saddle points isophotes intersect themselves, and just above and below this intersection its neighbor

isophotes have different *topology*: they have split from one curve into two, or merged from two curves into one.

```
blob[x_, y_, μx_, μy_, σ_] := 1/(2 π σ^2) Exp[-((x - μx)^2 + (y - μy)^2)/(2 σ^2)];
blobs[x_, y_] :=
  blob[x, y, 10, 10, 4] + .7 blob[x, y, 15, 20, 4] + 0.8 blob[x, y, 22, 8, 4];
Block[{$DisplayFunction = Identity}, p1 = Plot3D[blobs[x, y] - .00008,
    {x, 0, 30}, {y, 0, 30}, PlotPoints → 30, Mesh → False, Shading -> True];
  c = ContourPlot[blobs[x, y], {x, 0, 30}, {y, 0, 30},
    PlotPoints → 30, ContourShading → False];
  c3d = Graphics3D[Graphics[c][[1]] /.
     Line[pts_] :→ (val = Apply[blobs, First[pts]];
       Line[Map[Append[#, val] &, pts]])]];
Show[p1, c3d, ViewPoint -> {1.393, 2.502, 1.114}, ImageSize -> 250];
```

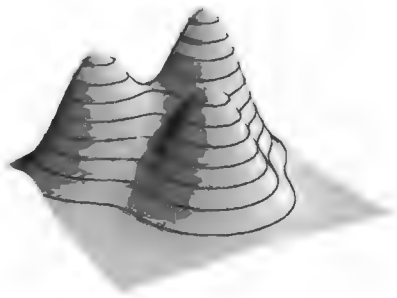


Figure 6.3 Isophote on a 2D 'landscape' image of 3 Gaussian blobs, depicted as heightlines. The height is determined by the intensity. The height plot is depicted slightly lower (-0.0002) in order to show the full extent of the isophotes.

At a minimum or maximum the isophote has shrunk to a point, and going to higher or lower intensity gives rise to the creation or disappearance of isophotes. This is best illustrated with an example of an image where only three Gaussian 'blobs' are present (see figure 6.3). The saddle points are in between the blobs. Isophotes through saddles and extrema are called *critical isophotes*.

We show the dynamic event of a 'split' and a 'merge' of an isophote by the behaviour of a two-parameter family of curves, the Cassinian ovals: $(x^2 + y^2 + a^2) - b^2 - 4\,a^2\,x^2 = 0$.

Famous members of Cassini functions are the circle (`cassini[x,y,a=0,b]`) and the lemniscate of Bernouilli (`cassini[x,y,a=b,b]`). The limaçon function, a generalization of the cardioid function, shows how we can get self-intersection where the new loop is formed within the isophote's inner domain. Here are the plots:

```
cassini[x_, y_, a_, b_] := (x^2 + y^2 + a^2)^2 - b^2 - 4 a^2 x^2;
DisplayTogetherArray[{
   ImplicitPlot[cassini[x, y, #, 4] == 0, {x, -5, 5}] & /@ {1.99, 2., 2.01},
   ParametricPlot[{2 Cos[t] + #) {Cos[t], Sin[t]}, {t, 0, 2 π}] & /@
    {3, 2., 1}}, ImageSize -> 400];
```

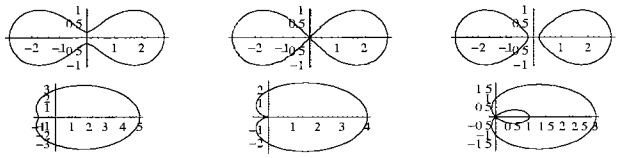


Figure 6.4 Top row: Split and merge of an isophote just under, at and above a saddle point in the image, simulated with a Cassini curve. Bottom row: Self intersection with an inner loop, simulated with the limaçon function. Examples taken from the wonderful book by Alfred Gray [Gray1993].

Isophotes in 3D are surfaces. Here is an example of the plotting of 4 isophote surfaces of a discrete dataset. We use the versatile OpenGL viewer MathGL3d developed by Jens-Peer Kuska: http://phong.informatik.uni-leipzig.de/~kuska/mathgl3dv3/

```
Get["MathGL3d`OpenGLViewer`"]; isos = Compile[{}, 10^3
   Table[Exp[-x^2/18 - y^2/8 - z^2/18], {z, -10, 10}, {y, -10, 10}, {x, -10, 10}]];
MVListContourPlot3D[isos[], Contours -> {.1, 1, 10}, ImageSize -> 150];
```

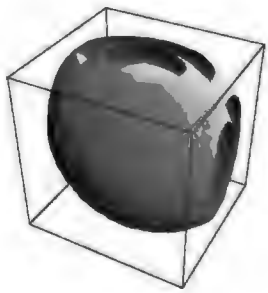


Figure 6.5 Isophotes in 3D are surfaces. Shown are the isophotes connecting all voxels with the values 0.1, 1, 10 and 100 in the discrete dataset of two neighboring 3D Gaussian blobs.

The calculations with the native command `ListContourPlot3D` take take much longer.

Flowlines are the lines everywhere perpendicular to the isophotes. E.g. for a Gaussian blob the isophotes are circles, and the flowlines are radiating lines from the center. Flowlines are the *integral curves* of the gradient, made up of all the small little gradient vectors in each point integrated to a smooth long curve. In 2D, the flowlines and the isophotes together form a *mesh* or *grid* on the intensity surface.

Figure 6.6 shows such a grid of the isophotes and flowlines of a 2D Gaussian blob (we have left out the singularity).

```
DisplayTogether[
  ShadowPlot3D[-gauss[x, 5] gauss[y, 5], {y, -15, 15}, {x, -15, 15}],
CartesianMap[Exp, {-π, π}, {-π, π}],
  ImageSize -> 200, AspectRatio -> 1];
```

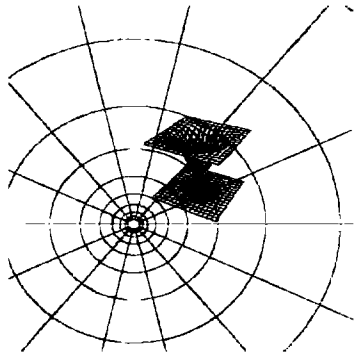


Figure 6.6 Isophotes and flowlines on the slope of a Gaussian blob. The circles are the isophotes, the flowlines are everywhere perpendicular to them. Inset: The height and intensity map of the Gaussian blob.

Just as in principle all isophotes together completely describe the intensity surface, so does the set of all flowlines. Flowlines are the *dual* of isophotes, isophotes are the *dual* of flowlines. One set can be calculated from the other. Just as the isophotes have a singularity at minima and maxima in the image, so have flowlines a singularity in direction in such points.

6.3 Coordinate systems and transformations

We will now apply the complete family of well behaving differential operators developed in the first chapter for the detection of local differential structure in images. The set of derivatives taken at a particular location is a *language* from which we can make a description of a local feature. We can make assemblies of the derivatives to any order, in any combination. Local structure is the local shape of the intensity landscape, like how sloped or curved it is, if there are saddlepoints, etc. The first order derivative gives us the slope, the second order is related to how curved the landscape is, etc.

In mathematical terms the image derivatives show up in the so-called Taylor expansion of our image function.

The Taylor expansion describes the function 'a little further up': If we move a little distance (δx, δy) away from the pixel where we stand, the Taylor expansion -or Taylor series- is given by (we take the expansion in the origin (0, 0) for notational convenience):

$$L(\delta x, \delta y) = L(0,0) + \left(\frac{\partial L}{\partial x}\delta x + \frac{\partial L}{\partial y}\delta y\right) + \frac{1}{2!}\left(\frac{\partial^2 L}{\partial x^2}\delta x^2 + \frac{\partial^2 L}{\partial x \partial y}\delta x \delta y + \frac{\partial^2 L}{\partial y^2}\delta y^2\right) +$$
$$\frac{1}{3!}\left(\frac{\partial^3 L}{\partial x^3}\delta x^3 + \frac{\partial^3 L}{\partial x^2 \partial y}\delta x^2 \delta y + \frac{\partial^3 L}{\partial x \partial y^2}\delta x \delta y^2 + \frac{\partial^3 L}{\partial y^3}\delta y^3\right) + O(\delta x^4, \delta y^4)$$

We see al the partial derivatives appearing. The spatial derivatives are taken at the location (0,0), e.g. $\frac{\partial^2 L}{\partial x^2}\Big|_{(0,0)}$. The first-order, second-order and third-order terms are grouped in brackets. Such groups of all terms of a specific order together are called 'binary forms'. The list goes to infinity, so we have to cut-off somewhere. The above series is an approximation to the third order, and the final expression $O(\delta x^4, \delta y^4)$ indicates that there is more, a rest term of order 4 and higher in δx and δy. *Mathematica* has the command **Series** to make a Taylor expansion. Here is the Taylor series for $L(x, y)$ for δx to second order and then expanded to second order by δy :

```
Series[L[δx, δy], {δx, 0, 2}, {δy, 0, 2}]
```

$$\left(L[0,0] + L^{(0,1)}[0,0]\,\delta y + \frac{1}{2} L^{(0,2)}[0,0]\,\delta y^2 + O[\delta y]^3\right) +$$
$$\left(L^{(1,0)}[0,0] + L^{(1,1)}[0,0]\,\delta y + \frac{1}{2} L^{(1,2)}[0,0]\,\delta y^2 + O[\delta y]^3\right)\delta x +$$
$$\left(\frac{1}{2} L^{(2,0)}[0,0] + \frac{1}{2} L^{(2,1)}[0,0]\,\delta y + \frac{1}{4} L^{(2,2)}[0,0]\,\delta y^2 + O[\delta y]^3\right)\delta x^2 + O[\delta x]^3$$

This expansion says essentially that we get a good approximation of the intensity landscape a little bit (δx, δy) further away from the origin (0, 0), when we first climb up over δx and δy with a slope given by the first derivative, the tangent. Then we come close, but not exactly. We can come somewhat better approximated when we include also the second order derivative, indicating how curved locally our landscape is. Etc. Taking into account more and more higher order terms gives us a better approximation and finally with the infinite series we have an exact description.
Our most important constraint for a good local image descriptor comes from the requirement that we want to be independent of our choice of coordinates. The coordinate system used the most is the Cartesian coordinate system (invented by and named after Descartes, a brilliant French mathematician from the 18th century): this is our familiar orthogonal (x, y) or (x, y, z) coordinate system.

But it should not matter if we describe our local image structure in another coordinate system like a polar, cylindrical or rotated or translated version of our Cartesian coordinate system. Because the Cartesian system is the easiest to understand, we will deal only with changes in this coordinate system. The *frame* of the coordinate system is formed by the unit vectors pointing in the respective dimensions. What changes could occur to a coordinate system? Of course any modification is possible. We will focus on the change of orientation (rotation of the axes frame), translation (x and/or y shift of the axes frame), and zoom (multiplication of the length of the units along the axes with some factor).

The shear transformation (where the axes are no longer orthogonal) will not be discussed here; we limit ourselves to changes of the coordinates where they remain orthogonal.

```
DisplayTogetherArray[
  Show[Graphics[{Arrow[{0, 0}, #] & /@ {{1, 0}, {0, 1}},
     Red, PointSize[.04], Point[{.4, .6}]}],
   Frame -> True, Axes -> True, AspectRatio -> 1],
  Show[Graphics3D[{arrow3D[{0, 0, 0}, #, True] & /@ {{1, 0, 0}, {0, 1, 0},
       {0, 0, 1}}, Red, PointSize[.04], Point[{.4, .6, .7}]}],
   Boxed -> True, BoxRatios -> {1, 1, 1}, Axes -> True], ImageSize -> 250];
```

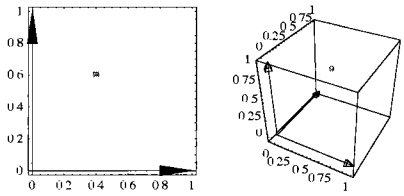


Figure 6.7 Use of graphics primitives in *Mathematica*: the coordinate unit vectors in 2D and 3D.

We call all the possible instantiations of a transformation the transformation *group*. So all rotations form the rotational group, the group of translations is formed by all translations. We now consider the transformation of the frame vectors.

Mathematically, the operation of a transformation is described by a matrix, the transformation matrix. E.g. rotation of a vector over an angle ϕ is described by the rotation matrix in 2D:

```
RotationMatrix2D[ϕ] // MatrixForm
```

$$\begin{pmatrix} \text{Cos}[\phi] & \text{Sin}[\phi] \\ -\text{Sin}[\phi] & \text{Cos}[\phi] \end{pmatrix}$$

The angle ϕ is defined as clockwise for the positive direction. In 3D it gets a little more complicated, as we have three angles to rotate over (these are called the 'Euler' angles):

```
RotationMatrix3D[ψ, θ, ϕ]

{{Cos[ϕ] Cos[ψ] - Cos[θ] Sin[ϕ] Sin[ψ],
   Cos[θ] Cos[ψ] Sin[ϕ] + Cos[ϕ] Sin[ψ], Sin[θ] Sin[ϕ]},
  {-Cos[ψ] Sin[ϕ] - Cos[θ] Cos[ϕ] Sin[ψ],
   Cos[θ] Cos[ϕ] Cos[ψ] - Sin[ϕ] Sin[ψ], Cos[ϕ] Sin[θ]},
  {Sin[θ] Sin[ψ], -Cos[ψ] Sin[θ], Cos[θ]}}
```

In general a transformation is described by a set of equations:

$$\begin{aligned} x'_1 &= f_1(x_1, x_2, \ldots, x_n) \\ &\vdots \\ x'_n &= f_n(x_1, x_2, \ldots, x_n) \end{aligned}$$

When we transform a space, the volume often changes, and the density of the material inside is distributed over a different volume. To study the change of a small volume we need to consider $\frac{\partial \vec{x}'}{\partial \vec{x}}$, which is the matrix of first order partial derivatives.

We have $J = \frac{\partial \vec{x}'}{\partial \vec{x}} = \begin{pmatrix} \frac{\partial (x')_1}{\partial x_1} & \cdots & \frac{\partial (x')_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial (x')_n}{\partial x_1} & \cdots & \frac{\partial (x')_n}{\partial x_n} \end{pmatrix}$. This matrix is called the *Jacobian matrix*, named after Carl Jacobi (1804-1851), a Prussian mathematician. The Jacobian can be computed in *Mathematica* with

```
jacobianmatrix[functions_List, variables_List] :=
 Outer[D, functions, variables]
```

If we consider the change of the infinitesimally small volume $d\,x'_1\,d\,x'_2 \ldots d\,x'_n = \left| \frac{\partial \vec{x}'}{\partial \vec{x}} \right| d\,x_1\,d\,x_2 \ldots d\,x_n$ we see that the determinant of the Jacobian matrix (also called the *Jacobian*) is the factor which corrects for the change in volume. When the Jacobian is unity, we call the transformation a *special* transformation.

The transformation in matrix notation is expressed as $\vec{x}' = A\,\vec{x}$, where $\vec{x}'$ is the transformed vector, $\vec{x}$ is the input vector, and $A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$ is the transformation matrix. When the coefficients of A are constant, we have a *linear* transformation, often called an *affine* transformation. In *Mathematica* (note the dot product between the matrix and the vector):

```
Clear[x, y]; A = {{a11, a12}, {a21, a22}}; x = {x1, x2};
x' = 1/Det[jacobianmatrix[A.x, x]] A.x
{(a11 x1 + a12 x2)/(-a12 a21 + a11 a22), (a21 x1 + a22 x2)/(-a12 a21 + a11 a22)}
```

▲ Task 6.1 Show that the Jacobian of the transformation matrices `RotationMatrix2D[ϕ]` and `RotationMatrix3D[ϕ,θ,ψ]` are unity.

A rotation matrix that rotates over zero degrees is the *identity matrix* or the *symmetric tensor* or δ*-operator*:

```
δ = RotationMatrix2D[0]; δ // MatrixForm
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and the matrix that rotates over 90 degrees ($\pi/2$ radians) is called the *antisymmetric tensor*, the ϵ*-operator* or the *Levi-Civita tensor*:

```
ε = RotationMatrix2D[π / 2]; ε // MatrixForm
```

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Let us study an example of a rotation: a unit vector under 45^0 is rotated over 110^0 clockwise:

```
v = {Sqrt[2]/2, Sqrt[2]/2}; v' = RotationMatrix2D[110 2 π/360].v // N
```

```
{0.422618, -0.906308}
```

```
Show[Graphics[{Arrow[{0, 0}, #] & /@ {v, v'}, Text["v", {.8, .8}],
    Text["v'", {.55, -.8}]}], PlotRange -> {{-1, 1}, {-1, 1}},
  Frame -> True, Axes -> True, AspectRatio -> 1, ImageSize -> 100];
```

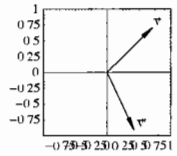


Figure 6.8 The vector $\vec{v}'$ is rotated by the action of the rotation matrix operator on the vector $\vec{v}$.

What we want is *invariance* under the transformations of translation and rotation. A function is said to be invariant under a group of transformations, if the transformation has no effect on the value of the function. The *only* geometrical entities that make physically sense are invariants. In the words of Hermann Weyl: "any invariant has a specific *meaning*", and as such they are widely studied in computer vision theories.

An example: The derivative to x is *not* invariant to rotation; if we rotate the coordinate system, or the image, we get in general a completely different value for the value of the derivative at that point. The same applies to the derivative to y. However, the combination $\sqrt{(\frac{\partial L}{\partial x})^2 + (\frac{\partial L}{\partial y})^2}$ is invariant, as can be seen from the following: We denote derivatives with a lower index: $L_x \equiv \frac{\partial L}{\partial x}$. The *length* of the *gradient vector* (L_x, L_y) is the scalar

```
Sqrt[{Lx, Ly}.{Lx, Ly}]
```

$$\sqrt{Lx^2 + Ly^2}$$

We used here again the **Dot** (.) product of vectors. When we now rotate each vector (L_x, L_y) with the rotation matrix over an arbitrary angle ϕ, we get

```
√((RotationMatrix2D[ϕ].{Lx, Ly}).(RotationMatrix2D[ϕ].{Lx, Ly}))
```

$$\sqrt{(Ly\,Cos[\phi] - Lx\,Sin[\phi])^2 + (Lx\,Cos[\phi] + Ly\,Sin[\phi])^2}$$

```
Simplify[%]
```

$$\sqrt{Lx^2 + Ly^2}$$

Invariance proved for this case. Invariants are so important, that the lower-order ones have a name. E.g. the scalar $\sqrt{\left(\frac{\partial L}{\partial x}\right)^2 + \left(\frac{\partial L}{\partial y}\right)^2}$ is called the *gradient magnitude*, the vector operator $\vec{\nabla} \equiv \left\{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right\}$ is called the *nabla operator*. So $\vec{\nabla} L$ is the gradient of L. $\vec{\nabla}.\left(\vec{\nabla} L\right) = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$ is called the *Laplacian*. Note that the gradient of the gradient $\vec{\nabla}\left(\vec{\nabla} L\right) = \begin{pmatrix} \frac{\partial^2 L}{\partial x^2} & \frac{\partial^2 L}{\partial x \partial y} \\ \frac{\partial^2 L}{\partial x \partial y} & \frac{\partial^2 L}{\partial y^2} \end{pmatrix}$ is the matrix of second order derivatives, or the *Hessian matrix* (this is not an invariant).

▲ Task 6.2 Show that the Laplacian is an invariant under rotation, in 2D and 3D.

In the sequel, we will only consider *orthonormal transformations*. These are also called Euclidean transformations. Orthonormal transformations are *special orthogonal* transformations (the Jacobian is unity). With orthogonal transformations the orthogonality of the coordinate frame is preserved. An orthonormal transformation preserves lengths of vectors and angles between vectors, i.e. it preserves a symmetric inner product $< \vec{x}, \vec{y} >$. When T is the orthogonal transformation, this means that $< \vec{x}, \vec{y} >= < T\vec{x}, T\vec{y} >$.

The transformation matrix of an orthogonal transformation is an *orthogonal matrix*. They have the nice property that they are always invertible, as the inverse of an orthogonal matrix is equal to its transpose: $A^{-1} = A^T$. A matrix m can be tested to see if it is orthogonal using

```
OrthogonalQ[m_List?MatrixQ] :=
  (Transpose[m].m == IdentityMatrix[Length[m]]);
```

Of course, there are many groups of transformations that can be considered, such as projective transformations (projecting a 3D world onto a 2D surface). In biomedical imaging mostly orthogonal transformations are encountered, and on those which will be the emphasis of the rest of this chapter.

Notice that with invariance we mean invariance for the transformation (e.g. rotation) of the coordinate system, not of the image. The value of the local invariant properties is also the same when we rotate the image. There is however an important difference between image rotation, and coordinate rotation. We specifically mean here the *local* independence of rotation, for that particular point. See also figure 6.9. If we study the rotation of the whole image, we apply the same rotation to every pixel.

Here, we want in every point a description which is independent to the rotation of the local coordinates, so we may as well rotate our coordinates in every pixel differently. Invariance for rotation in this way means something different than a rotation of the image. There would be no way otherwise to recognize rotated images from non-rotated ones!

```
Show[Import["Thatcher illusion.jpg"], ImageSize -> 330];
```

Figure 6.9 The "Thatcher illusion", created by P. Thompson [Thompson1980], shows that local rotations of image *patches* are radically different from the local coordinate rotation invariance, and that we are not used to (i.e. have no associative set in our memory) for sights that we seldomly see: faces upside down. Rotate the images 180 degrees to see the effect.

In particular, we will see that specific scalar combinations of local derivatives give descriptions of local image structure invariant under a Euclidean transformation.

6.4 Directional derivatives

The directed first order nabla operator is given in 2D by $\vec{v}.\vec{\nabla}$, where $\vec{v}$ is a unit vector pointing in the specific direction. $\vec{v}.\vec{\nabla}$ is called the *directional derivative*. Let us consider some examples. We calculate the directional derivative for $\vec{v} = \{-\sqrt{2}, -\sqrt{2}\}$ and $\vec{v} = \{\sqrt{3}/2, 1/2\}$:

```
im = Import["mip147.gif"][[1, 1]];
northeast[im_, σ_] := {-√2, -√2}.{gD[im, 1, 0, σ], gD[im, 0, 1, σ]};
southsouthwest[im_, σ_] :=
  {√3/2, 1/2}.{gD[im, 1, 0, σ], gD[im, 0, 1, σ]};
DisplayTogetherArray[ListDensityPlot /@
   {im, northeast[im, 1], southsouthwest[im, 1]}, ImageSize -> 300];
```

Figure 6.10 Directional derivatives. Image from the Eurorad database (www.eurorad.org), case 147.

6.5 First order gauge coordinates

We introduce the notion of *intrinsic geometry*: we like to have every point described in such a way, that if we have the same structure, or local landscape form, no matter the rotation, the description is always the same. This can be accomplished by setting up in each point a dedicated coordinate frame which is determined by some special local directions given by the landscape locally itself.

Consider yourself an ant on a surface, you can only see the direct vicinity, so the world looks locally very simple. We now fix *in each point separately* our local coordinate frame in such a way that one frame vector points to the direction of maximal change of the intensity, and the other perpendicular to it (90 degrees clockwise). The direction of maximal change of intensity is just the gradient vector $\vec{w} = \left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}\right)$. The perpendicular direction is $\vec{v} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} . \vec{w} = \left(\frac{\partial L}{\partial y}, -\frac{\partial L}{\partial x}\right)$. We can check: if we are on a slope going up in the y-direction only (the 'Southern' slope of a hill), we have as gradient $\left\{0, \frac{\partial L}{\partial y}\right\}$, because in the x-direction the slope is horizontal.

```
ContourPlot[x^2 + y^2, {y, 2, 4.5},
  {x, 2, 4.5}, Contours → Range[2, 100, 4], Epilog ->
   {PointSize[.02], Point[{3, 3}], Arrow[{3, 3}, {3 + .5 √2, 3 - .5 √2}],
    Arrow[{3, 3}, {3 + .5 √2, 3 + .5 √2}], Text["v̂", {3.8, 2.2}],
    Text["ŵ", {3.8, 3.8}]}, Frame → False, ImageSize → 100];
```

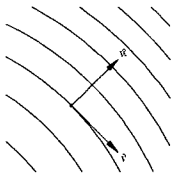


Figure 6.11 Local first order gauge coordinates $\{\hat{v}, \hat{w}\}$. The unit vector $\hat{v}$ is everywhere tangential to the isophote (line of constant intensity), the unit vector $\hat{w}$ is everywhere perpendicular to the isophote and points in the direction of the gradient vector.

We have now *fixed* locally the direction for our new intrinsic local coordinate frame $(\vec{v}, \vec{w})$. This set of local directions is called a *gauge*, the new frame forms the *gauge coordinates* and fixing the frame vectors with respect to the constant direction $\vec{w}$ is called: *fixing the gauge*. Because we discuss first order derivatives here, we call this a *first order gauge*. We can also derive a second order gauge from second order local differential structure, as we will see later.

We want to take derivatives with respect to the gauge coordinates.

As they are fixed to the object, no matter any rotation or translation, we have the following very useful result:

any derivative expressed in gauge coordinates is an orthogonal invariant. E.g. it is clear that $\frac{\partial L}{\partial w}$ is the derivative in the gradient direction, and this is just the gradient itself, an invariant.

And $\frac{\partial L}{\partial v} \equiv 0$, as there is no change in the luminance as we move tangentially along the isophote, and we have chosen this direction by definition.

From the derivatives with respect to the gauge coordinates, we always need to go to Cartesian coordinates in order to calculate the invariant properties on a computer. The transformation from the $(\hat{v}, \hat{w})$ from to the Cartesian $(\hat{x}, \hat{y})$ frame is done by implementing the definition of the directional derivatives. Important is that first a directional partial derivative (to whatever order) is calculated with respect to a *frozen* gradient direction. We call this direction (Lx, Ly). Then the formula is calculated which expresses the gauge derivative into this direction, and finally the frozen direction is filled in from the *calculated* gradient.

In *Mathematica*: The frame vectors $\hat{w}$ and $\hat{v}$ are defined as

```
ŵ = {Lx, Ly} / Sqrt[Lx^2 + Ly^2]; v̂ = {{0, 1}, {-1, 0}}.ŵ;
```

The directional differential operators $\hat{v} \cdot \vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$ and $\hat{v} \cdot \vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$ are defined as:

```
v̂.{∂x #, ∂y #} &;
v̂.{∂x #, ∂y #} &;
```

The notation `(...#)&` is a 'pure function' on the argument `#`, e.g. `(#^2 + #^5) &` gives the sum of second and fifth power of some argument #, `D[#,x]&` (or equivalently `(∂x #) &`) takes the derivative of the variable `#` with respect to `x` (look in the Help browser to `Function` for more examples). So the construct of a pure function is the construct for an *operator*. This pure function can be applied to an argument by the familiar square brackets, e.g.

```
(#^2 + #^5) &[zz]
```

$$zz^2 + zz^5$$

Higher order derivatives are constructed through nesting multiple first order derivatives, as many as needed. The total transformation routine is now:

```
Clear[f, L, Lx, Ly]; Unprotect[gauge2D];
gauge2D[f_, nv_/; nv ≥ 0, nw_/; nw ≥ 0] :=
  Module[{Lx, Ly, v, w}, w = {Lx, Ly} / Sqrt[Lx^2 + Ly^2];
   v = {{0, 1}, {-1, 0}}.w;
   Simplify[
    Nest[(v.{D[#1, x], D[#1, y]} &), Nest[(w.{D[#1, x], D[#1, y]} &),
       f, nw], nv] /. {Lx → D[f, x], Ly → D[f, y]}]];
```

where f is a symbolic function of x and y, and n_w and n_v are the orders of differentiation with respect to w resp v. Here is an example of its output: the gradient $\frac{\partial L}{\partial w}$:

```
Lw = gauge2D[L[x, y], 0, 1]
```

$$\sqrt{L^{(0,1)}[x, y]^2 + L^{(1,0)}[x, y]^2}$$

Using pattern matching with the function **shortnotation** we get more readable output:

```
Lw = gauge2D[L[x, y], 0, 1] // shortnotation
```

$$\sqrt{L_x^2 + L_y^2}$$

```
Lww = gauge2D[L[x, y], 0, 2] // shortnotation
```

$$\frac{L_x^2 L_{xx} + 2 L_x L_{xy} L_y + L_y^2 L_{yy}}{L_x^2 + L_y^2}$$

```
Lv = gauge2D[L[x, y], 1, 0] // shortnotation
```

0

As expected, because it is exactly what we put into the definition of $\frac{\partial L}{\partial v}$: it is the differentiation in the direction perpendicular to the gradient, so along the tangential direction of the isophote, and in this direction there is *no* change of the intensity function. But

```
Lvv = gauge2D[L[x, y], 2, 0] // shortnotation
```

$$\frac{-2 L_x L_{xy} L_y + L_{xx} L_y^2 + L_x^2 L_{yy}}{L_x^2 + L_y^2}$$

is *not* zero, because it is constructed by first applying the directional derivative twice, and then fixing the gauge.

This calculates the Laplacian in gauge coordinates, $L_{vv} + L_{ww}$ (what do you expect?):

```
gauge2D[L[x, y], 0, 2] + gauge2D[L[x, y], 2, 0] // shortnotation
```

$$L_{xx} + L_{yy}$$

▲ Task 6.3 Show and explain that in the definition of the function gauge2D we cannot define $w = \{\partial_x L, \partial_y L\}$. We need to have the direction of the gauge fixed while computing the compound formula. Why?

The next figure shows the $\{\hat{v}, \hat{w}\}$ gauge frame in every pixel of a simple 32^2 image with 3 blobs:

```
blob[x_, y_, μx_, μy_, σ_] := 1/(2 π σ^2) Exp[-((x - μx)^2 + (y - μy)^2)/(2 σ^2)];
blobs[x_, y_] :=
  blob[x, y, 10, 10, 4] + .7 blob[x, y, 15, 20, 4] + 0.8 blob[x, y, 22, 8, 4];
im = Table[blobs[x, y], {y, 30}, {x, 30}];
Block[{$DisplayFunction = Identity, gradient, norm, σ, frame},
  norm = (# / Sqrt[#.#]) &;
σ = 1; gradient = Map[norm,
    Transpose[{gD[im, 1, 0, σ], gD[im, 0, 1, σ]}, {3, 2, 1}], {2}];
frame = Graphics[{White, Arrow[#2 - .5, #2 - .5 + #1], Red,
      Arrow[#2 - .5, #2 - .5 + {#1[[2]], -#1[[1]]}]}] &;
ar = MapIndexed[frame, gradient / 2, {2}];
  lp = ListDensityPlot[gD[im, 0, 0, σ]]];

Show[{lp, ar}, Frame -> True, ImageSize -> 410];
```

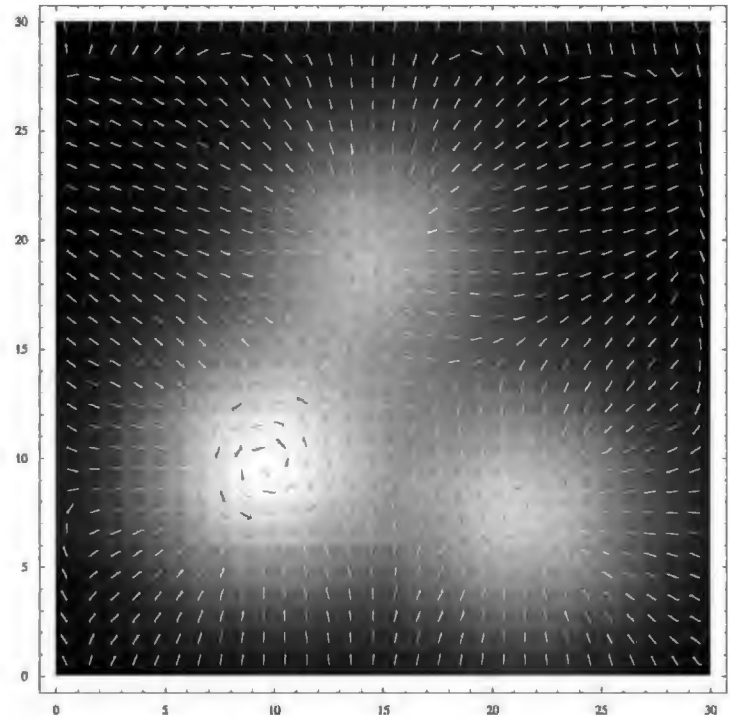


Figure 6.12 The gauge frame $\{w, v\}$ given for every pixel in a 30^2 image of three Gaussian blobs. The gradient direction w, calculated at a scale of σ=1 pixel, is indicated in white, and points always to higher intensity. They are (defined as) everywhere perpendicular to the isophotes and tangential to the flowlines. These vectors always point to extrema and saddle points. The v frame vector (in red) is $\pi/2$ radians rotated clockwise, they encircle the extrema, (defined as) tangential to the isophotes. (The boundary effects, most notably on the right, are due to the cyclic interpretation of the gradient calculation, which causes the image to be interpreted as infinitely repeated in all directions: the gradient direction changes over π, no artefact, but now well understood).

The gauge coordinates are not defined at 'horizontal points' in the intensity landscape, i.e. locations where $\sqrt{{L_x}^2 + {L_y}^2} = 0$, as is clear from the definition of the gauge coordinates. This occurs in saddle points and extrema (minima and maxima) of the intensity landscape, where both $L_x = 0$ and $L_y = 0$. In practice however this is not a problem: we have a finite number of such points, typically just a few, and we know from *Morse theory* that we can get rid of such a *singularity* by an infinitesimally small local change in the intensity landscape.

Due to the fixing of the gauge by removing the degree of freedom for rotation (that is why $L_v \equiv 0$), we have an important result: *every derivative to v and w is an orthogonal invariant.*

In other words: it is an invariant property where translation and/or rotation of the coordinate frame is irrelevant. This also means that polynomial combinations of these gauge derivative terms are invariant. We now have the toolkit to make invariants expressed in gauge derivatives to any order.

Here are a few other differential invariants of the image, which are now easily constructed:

```
gauge2D[L[x, y], 4, 0] // shortnotation
```

$$\frac{-4\,L_x^3\,L_{xyyy}\,L_y + 6\,L_x^2\,L_{xxyy}\,L_y^2 - 4\,L_x\,L_{xxxy}\,L_y^3 + L_{xxxx}\,L_y^4 + L_x^4\,L_{yyyy}}{(L_x^2 + L_y^2)^2}$$

```
gauge2D[L[x, y], 2, 1] // shortnotation
```

$$\frac{L_x^3\,L_{xyy} + L_x\,(L_{xxx} - 2\,L_{xyy})\,L_y^2 + L_{xxy}\,L_y^3 + L_x^2\,L_y\,(-2\,L_{xxy} + L_{yyy})}{(L_x^2 + L_y^2)^{3/2}}$$

In conclusion of this section, we have found a *complete family* of differential invariants, that are invariant for rotation and translation of the coordinate frame. They are called differential invariants, because they consist of polynomials with as coefficients partial derivatives of the image. In the next section we discuss some important members of this family. Only the lowest order invariants have a *name*, the higher orders become more and more exotic.

The final step is the operational implementation of the gauge derivative operators for discrete images. This is simply done by applying pattern matching:
- first calculate the symbolic expression
- then replace any derivative with respect to x and y by the numerical derivative `gD[im,`n_x`,`n_y`,`σ`]`
- and then insert the pixeldata in the resulting polynomial function;
as follows:

```
Unprotect[gauge2DN];
gauge2DN[im_, nv_, nw_, σ_ /; σ > 0] :=
  Module[{im0}, gauge2D[L[x, y], nv, nw] /.
     Derivative[nx_, ny_][L_][x_, y_] → gD[im0, nx, ny, σ] /. im0 → im];
```

This *writes our numerical code automatically*. Here is the implementation for L_{vv}. If the image is not defined, we get the formula returned:

```
Clear[im, σ]; gauge2DN[im, 2, 0, 2]

(gD[im, 0, 2, 2] gD[im, 1, 0, 2]^2 -
   2 gD[im, 0, 1, 2] gD[im, 1, 0, 2] gD[im, 1, 1, 2] +
   gD[im, 0, 1, 2]^2 gD[im, 2, 0, 2]) / (gD[im, 0, 1, 2]^2 + gD[im, 1, 0, 2]^2)
```

If the image is available, the invariant property is calculated in each pixel:

```
im = Import["thorax02.gif"][[1, 1]];
DisplayTogetherArray[
  ListDensityPlot /@ {im, -gauge2DN[im, 0, 1, 1], -gauge2DN[im, 2, 0, 4]},
  ImageSize -> 400];
```

Figure 6.13 The gradient L_w (middle) and L_{vv}, the second order directional derivative in the direction tangential to the isophote (right) for a 256^2 X-thorax image at a small scale of 0.5 pixels. Note the shadow of the coins in the pocket of his shirt in the lower right.

6.6 Gauge coordinate invariants: examples

6.6.1 Ridge detection

L_{vv} is a good ridge detector, since at ridges the curvature of isophotes is large (see figure 6.13).

```
f[x_, y_] := (Sin[x] + 1/3 Sin[3 x]) (1 + .1 y);
DisplayTogetherArray[Plot3D[f[x, y], {x, 0, π}, {y, 0, π}],
 ContourPlot[f[x, y], {x, 0, π}, {y, 0, π}, PlotPoints -> 50],
 ImageSize -> 370];
```

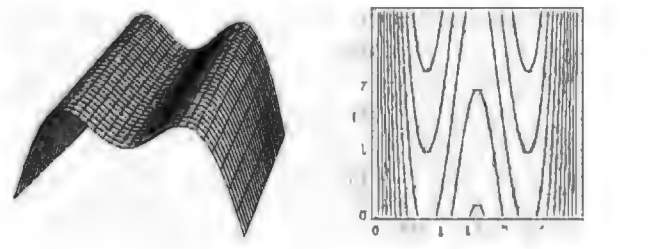


Figure 6.14 Isophotes are much more curved at the top of ridges and valleys then along the slopes of it. Left: a slightly sloping artificial intensity landscape with two ridges and a valley, at right the contours as isophotes.

Let us test this on an X-ray image of fingers and calculate L_{vv} scale $\sigma = 3$.

```
im = Import["hands.gif"][[1, 1]]; Lvv = gauge2DN[im, 2, 0, 3];
DisplayTogetherArray[ListDensityPlot /@ {im, Lvv}, ImageSize → 450];
```

Figure 6.15 The invariant feature L_{vv} is a ridge detector. Here applied on an X-ray of two hands at $\sigma = 3$ pixels. Image resolution: 361 x 239 pixels.

▲ Task 6.4 Study the ridges L_{vv} of the fingers at different scales, and note the scale-dependent interpretation.

Noise has structure too. Here are the ridges of uniform white noise:

```
im = Table[Random[], {128}, {256}];
ListDensityPlot[gauge2DN[im, 2, 0, 4]];
```

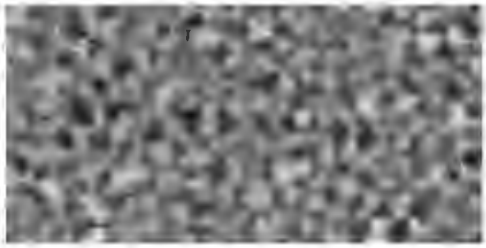


Figure 6.16 The invariant feature L_{vv} detects the ridges in white noise here, $\sigma = 4$ pixels, image resolution: 256 x 128 pixels.

▲ Task 6.5 Study in the same way the gradient of white noise at a range of scales. Do you see the similarity with a brain surface at larger scales?

We will encounter the second order gauge derivative L_{vv} in chapter 19 in the 'fundamental' equation of Alvarez et al. [Alvarez1992a, Alvarez1993], a *nonlinear* (geometry driven) diffusion equation: $\frac{\partial L}{\partial t} = L_{vv}$.

This equation is used to evolve the image in a way that locally adapts the amount of blurring to differential invariant structure in the image in order to do e.g. edge-preserving smoothing. We discuss this in detail in chapter 21.

Detection of ridges is an active topic in multi-scale feature detection [Koenderink1993a, Maintz1996a, Eberly1993, Eberly1994, Eberly1994a, Eberly1994b, Damon1999, Lindeberg1998b, Lopéz1999], as it focuses on the *dual* of boundaries.

6.6.2 Isophote and flowline curvature in gauge coordinates

The derivation of the formula for isophote curvature is particularly easy when we express the problem in gauge coordinates. Isophote curvature κ is defined as the change $w'' = \frac{\partial^2 w}{\partial v^2}$ of the tangent vector $w' = \frac{\partial w}{\partial v} = v$ in the gradient-gauge coordinate system. The definition of an isophote is: $L(v, w) =$ Constant, and $w = w(v)$. So, in *Mathematica* we implicitly differentiate the equality (==) to v:

```
L[v, w[v]] == Constant;
v =.; w =.; D[L[v, w[v]] == Constant, v]
```

$$w'[v]\, L^{(0,1)}[v, w[v]] + L^{(1,0)}[v, w[v]] == 0$$

We know that $L_v \equiv 0$ by definition of the gauge coordinates, so $w' = 0$, and the curvature $\kappa = w''$ is found by differentiating the isophote equation again and solving for w'':

```
κ = w''[v] /. Solve[D[L[v, w[v]] == Constant, {v, 2}] /. w'[v] -> 0, w''[v]]
```

$$\left\{-\frac{L^{(2,0)}[v, w[v]]}{L^{(0,1)}[v, w[v]]}\right\}$$

So $\kappa = -\frac{L_{vv}}{L_w}$. In Cartesian coordinates we recognize the well-known formula:

```
im =.; κ = -gauge2D[L[x, y], 2, 0] / gauge2D[L[x, y], 0, 1] // shortnotation
```

$$-\frac{-2\, L_x\, L_{xy}\, L_y + L_{xx}\, L_y^2 + L_x^2\, L_{yy}}{(L_x^2 + L_y^2)^{3/2}}$$

Here is an example of the isophote curvature at a range of scales for a sagittal MR image:

```
im = Import["mr256.gif"][[1, 1]];
κplot =
  ListDensityPlot[-gauge2DN[im, 2, 0, #] / gauge2DN[im, 0, 1, #], PlotRange -> {-5, 5}] &;
```

```
DisplayTogetherArray[
  {ListDensityPlot[im], κplot[1], κplot[2], κplot[3]}, ImageSize -> 470];
```

Figure 6.17 The isophote curvature κ is a rotationally and translationally invariant feature. It takes high values at extrema. Image resolution: 256^2 pixels.

The reason we see extreme low and high values is due to the singularities that occur at intensity extrema, where the gradient $L_w = 0$.

▲ Task 6.6 Why was not in a single pixel infinite isophote curvature encountered? There are many maxima and minima in the image.

Lopéz et al. [Lopéz2000b] defined a robust multi-scale version of a local curvature measure, which they termed level set extrinsic curvature, based on the divergence of the gradient field, integrated over a path (with a certain are: the scale) around the point of interest.
The *perception of curvature* is influenced by its context, as is clear from the Tolansky's curvature illusion (see figure 6.18).

```
Show[
  Graphics[{Thickness[.01], Circle[{0, 0}, 10, {0, π}], Circle[{0, -4},
     10, {π/4, 3π/4}], Circle[{0, -8}, 10, {3π/8, 5π/8}]}],
  AspectRatio → Automatic, ImageSize -> 260];
```

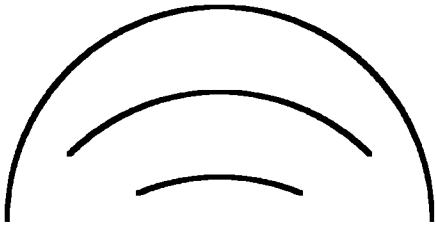


Figure 6.18 Tolansky's curvature illusion. The three circle segments have the same curvature 1/10.

We remember the *flowlines* as the integral curves of the gradient. In figure 6.6 they were depicted together with their *duals*, the isophotes. In that particular case, for such circular objects flowlines are straight lines with curvature zero. In figure 6.6 the isophote curvature at the top of the blob goes to infinity and is left out for that reason.

▲ Task 6.7 Prove, with the methodology sketched above, that the flowline curvature expressed in first order gauge coordinates is: $\mu = -\frac{L_{vw}}{L_w}$.

The third (and last) member of the set of second order derivatives in gauge coordinates is L_{ww}. This is the derivative of the gradient in the gradient direction. So when we want to find the maximum of the gradient, we can inspect zeros of L_{ww}.
Historically, much attention is paid to the zerocrossings of the Laplacian due to the groundbreaking work of Marr and Hildreth. As a rotational isotropic filter, and its close analogy to the retinal receptive fields, its zerocrossings were often interpreted as the maxima of a rotational invariant edge detector. The zerocrossings are however displaced on curved edges.

Note that with the compact expression for isophote curvature $\kappa = -\frac{L_{vv}}{L_w}$ we can establish a relation between the Laplacian and the second order derivative in the gradient direction we want to investigate for zerocrossings: L_{ww}. From the expression of the Laplacian in gauge coordinates $\Delta L = L_{ww} + L_{vv} = L_{ww} - \kappa L_w$ we see immediately that there is a deviation term κL_w which is directly proportional to the isophote curvature κ. Only on a straight edge with local isophote curvature zero the Laplacian is numerically equal to L_{ww}. Without gauge coordinates, this is much harder to prove. It took Clark two full pages in PAMI to show this [Clark1989]!

```
im = Import["thorax02.gif"][[1, 1]];
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[im];
  p2 = ListContourPlot[gauge2DN[im, 0, 2, 4], Contours -> {0}];
  p3 = ListContourPlot[gD[im, 2, 0, 4] + gD[im, 0, 2, 4], Contours -> {0}]];

DisplayTogetherArray[{Show[{p1, p2}], Show[{p1, p3}]}, ImageSize -> 380];
```

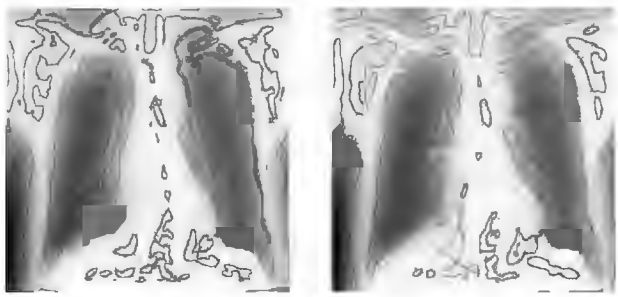


Figure 6.19 Contours of $L_{vv} = 0$ (left) and $\Delta L = 0$ (right) superimposed on the X-thorax image for $\sigma = 4$ pixels.

The term $v = -\frac{L_{ww}}{L_w}$ is not a curvature, but can be interpreted as a *density* of isophotes.

Notice that the isophote curvature $\kappa = -\frac{L_{vv}}{L_w}$ and flowline curvature $\mu = -\frac{L_{vw}}{L_w}$ have equal dimensionality for the intensity in both nominator and denominator. This leads to the desirable property that these curvatures do not change when we e.g. manipulate the contrast or brightness of an image. In general, these curvatures are said to be *invariant under monotonic intensity transformations*. In section 6.7 we elaborate on this special case of invariance.

6.6.3 Affine invariant corner detection

Corners are defined as locations with high isophote curvature and high intensity gradient. An elegant reasoning for an affine invariant corner detector was proposed by Blom [Blom1991a], then a PhD student of Koenderink. We reproduce it here using *Mathematica*. Blom proposed to take the *product* of isophote curvature $-\frac{L_{vv}}{L_w}$ and the gradient L_w raised to some (to be determined) power n:

$$\Theta^{[n]} = -\frac{L_{vv}}{L_w} L_w{}^n = \kappa L_w{}^n = -L_{vv} L_w{}^{n-1}.$$

An obvious advantage is invariance under a transformation that changes the opening angle of the corner. Such a transformation is the *affine* transformation. An affine transformation is a *linear* transformation of the coordinate axes:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{1}{a\,d-b\,c} \begin{pmatrix} a & b \\ c & d \end{pmatrix} (x \quad y) + (e\,f).$$

We omit the translation term $(e\,f)$ and study the affine transformation proper. The term $\frac{1}{a\,d-b\,c}$ is the determinant of the transformation matrix, and is called the *Jacobian*. Its purpose is to adjust the amplitude when the area changes.

A good example of the effect of an affine transformation is to study the projection of a square from a large distance. Rotation over a vertical axis shortens the x-axis. Changing both axes introduces a *shear*, where the angles between the sides change. The following example illustrates this by an affine transformation of a square:

```
square = {{0, 0}, {1, 0}, {1, 1}, {0, 1}, {0, 0}};
affine = {{5, 2}, {0, .5}}; afsquare = affine.# & /@ square;
DisplayTogetherArray[Graphics[Line[#], AspectRatio -> 1] & /@
   {square, afsquare}, ImageSize -> 200];
```

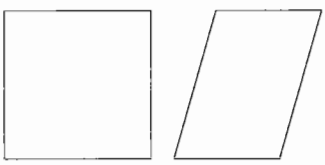


Figure 6.20 Affine transformation of a square, with transformation matrix $\begin{pmatrix} 5 & 2 \\ 0 & .5 \end{pmatrix}$ mapped on each point.

The derivatives transform as $\begin{pmatrix} \partial_{x'} \\ \partial_{y'} \end{pmatrix} = \frac{1}{a\,d - b\,c} \begin{pmatrix} a & b \\ c & d \end{pmatrix} (\partial_x \;\; \partial_y)$. We put the affine transformation $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ into the definition of affinely transformed gauge coordinates:

```
Clear[a, b, c, d];
gauge2Daffine[f_, nv_, nw_] := Module[{Lx, Ly, v, w, A = ( a  b
                                                           c  d )},
  w = {Lx', Ly'} / Sqrt[Lx'^2 + Ly'^2]; v = ( 0  1
                                             -1  0 ).w;
  Simplify[Nest[v.(1/Det[A] A.{∂x #, ∂y #}) &,
      Nest[w.(1/Det[A] A.{∂x #, ∂y #}) &, f, nw], nv] /.
    {Lx' → (a Lx + b Ly)/Det[A], Ly' → (c Lx + d Ly)/Det[A]} /. {Lx → ∂x f, Ly → ∂y f}]];
```

The equation for the affinely distorted coordinates $-L_{v_a v_a} L_{w_a}{}^{n-1}$ now becomes:

```
-gauge2Daffine[L[x, y], 2, 0] gauge2Daffine[L[x, y], 0, 1]^(n-1) //
  Simplify // shortnotation
```

$$\frac{\left(\frac{(a^2+c^2)\,L_x^2 + 2\,(a\,b+c\,d)\,L_x\,L_y + (b^2+d^2)\,L_y^2}{(b\,c-a\,d)^2}\right)^{\frac{1}{2}(-3+n)} (2\,L_x\,L_{xy}\,L_y - L_{xx}\,L_y^2 - L_x^2\,L_{yy})}{(b\,c-a\,d)^2}$$

Very interesting: when $n = 3$ and for an affine transformation with unity Jacobean ($a\,d - b\,c = 1$, a so-called *special* transformation) we are independent of the parameters a, b, c and d! This is the affine invariance condition.

So the expression $\Theta = \frac{L_{vv}}{L_w} L_w{}^3 = L_{vv} L_w{}^2 = 2\,L_x L_{xy} L_y - L_{xx} L_y{}^2 - L_x{}^2 L_{yy}$ is an *affine invariant corner detector*. This feature has the nice property that it is not singular at locations where the gradient vanishes, and through its affine invariance it detects corners at all 'opening angles'.

We show corner detection at two scales on the 'Utrecht' image:

```
im = SubMatrix[Import["Utrecht256.gif"][[1, 1]], {1, 128}, {128, 128}];
corner1 = gauge2DN[im, 2, 0, 1] gauge2DN[im, 0, 1, 1]^2;
corner3 = gauge2DN[im, 2, 0, 3] gauge2DN[im, 0, 1, 2]^2;
```

```
DisplayTogetherArray[
  ListDensityPlot /@ {im, corner1, corner3}, ImageSize → 500];
```

Figure 6.21 Corner detection with the $L_{vv} L_{w}{}^{2}$ operator. Left: original image, dimensions 128^2. Middle: corner detection at σ = 1 pixel; right: corner detection at σ = 3 pixels. Isophote curvature is signed, so note the positive (convex, light) and negative (concave, dark) corners.

▲ Task 6.8 Show why the compound spike response, where an rotationally invariant operator is applied on a spike image (discrete delta function), leads to a rotationally symmetric response. An example is given below:

```
spike = Table[0, {128}, {128}]; spike[[64, 64]] = 100;
gradient = gauge2DN[spike, 0, 1, 15];
cornerness = -gauge2DN[spike, 2, 0, 15] gauge2DN[spike, 0, 1, 15]^2;
DisplayTogetherArray[
  ListDensityPlot /@ {spike, gradient, cornerness}, ImageSize -> 400];
```

Figure 6.22 Convolution of a spike (Delta function) image with a kernel gives the kernel itself as result. Left: spike image, middle: response to the gradient kernel assembly, right: response to the cornerness kernel assembly. Scale σ = 15 pixels, resolution image 128^2.

6.7 A curvature illusion

A particular visual illusion shows the influence of the *multi-scale* perception of a local property, like curvature. In figure 6.23 the lines appear curved, though they are really straight.

```
star = Graphics[Table[
    Line[{{Cos[ϕ], Sin[ϕ]}, {-Cos[ϕ], -Sin[ϕ]}}], {ϕ, 0, π, π / 20}]];
lines = Graphics[{Thickness[.015], DarkViolet,
    Line[{{-1, .1}, {1, .1}}], Line[{{-1, -.1}, {1, -.1}}]}];
Show[{star, lines}, PlotRange → {{-.4, .4}, {-.2, .2}},
  AspectRatio → Automatic, ImageSize -> 300];
```

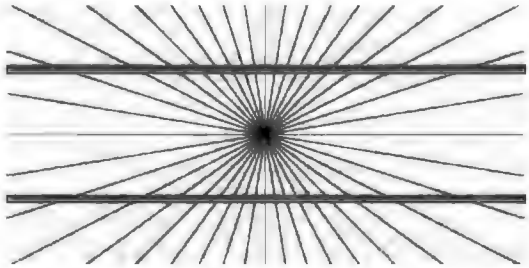


Figure 6.23 The straight lines appear curved due to the surrounding pattern.

When we calculate the isophote curvature $\kappa = \frac{-L_{vv}}{L_w}$ for this figure at a coarse scale, we see that the curvature is not constant along the horizontal lines, but changes when moving from the center. Figure 6.24 shows the curvature and the profile along the center of the horizontal line.

```
curvill = Show[{star, lines}, PlotRange → {{-.4, .4}, {-.2, .2}},
  AspectRatio → Automatic, ImageSize -> 432, DisplayFunction -> Identity];
Export["curvillusion-star.jpg", curvill];
iml = Import["curvillusion-star.jpg"][[1, 1]] /. {a_, b_, c_} -> (a + b + c) / 3;
DeleteFile["curvillusion-star.jpg"];

DisplayTogetherArray[
  {ListDensityPlot[κ1 = -gauge2DN[iml, 2, 0, 20] / gauge2DN[iml, 0, 1, 20], PlotRange -> {-.1, .1},
    Epilog -> {Red, Line[{{110, 161}, {320, 161}}]}],
ListPlot[Take[κ1, {161, 161}, {110, 320}] // Flatten,
    AspectRatio -> .4, AxesLabel -> {"", "κ1"}]}, ImageSize -> 450];
```

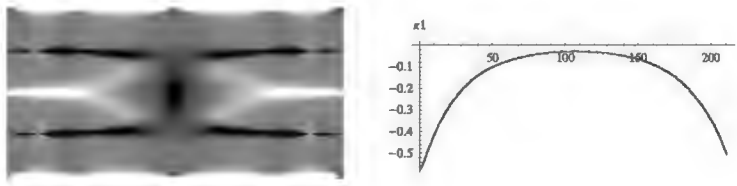


Figure 6.24 Left: Isophote curvature κ at a scale of $\sigma = 20$ pixels for the pattern in figure 6.23, dimensions image 216 x 432 pixels. Right: profile of curvature along the central portion of the top horizontal line (to avoid boundary effects only the central portion is shown, indicated by the red line in the left figure).

6.8 Second order structure

The second order structure of the intensity landscape is rich. To describe and to represent it, we will develop a precise mathematical formulation in order to do a proper analysis.

Let us first develop some intuitive notions by visual inspection. Figure 6.25 shows a blurred version of an X-thorax image is depicted as a height plot. We see hills and dales, saddle points, ridges, maxima and minima. Clearly curvature plays an important role.

The second order structure of the intensity landscape $L(x, y; \sigma)$ in a point $L(x_0, y_0; \sigma)$ is described by the second order term in the local Taylor expansion around the point (x_0, y_0). Without any loss of generalization we take $(x_0, y_0) = (0, 0)$:

```
s = Series[L[x, y], {x, 0, 2}, {y, 0, 2}] // Normal // shortnotation
```

$$L[0,0] + x\,L_x + \frac{x^2\,L_{xx}}{2} + y\left(\frac{x^2\,L_{xxy}}{2} + x\,L_{xy} + L_y\right) + \frac{1}{4}\,y^2\,(x^2\,L_{xxyy} + 2\,(x\,L_{xyy} + L_{yy}))$$

The second order term is $\frac{1}{2} L_{xx}\, x^2 + L_{xy}\, x\, y + \frac{1}{2} L_{yy}\, y^2$. The second order derivatives are the coefficients in the quadratic polynomial that describes the second order landscape.

```
im = Import["thorax02.gif"][[1, 1]];
DisplayTogetherArray[ListDensityPlot[im],
  ListPlot3D[-gD[im, 0, 0, 2], Mesh → False], ImageSize → 320];
```

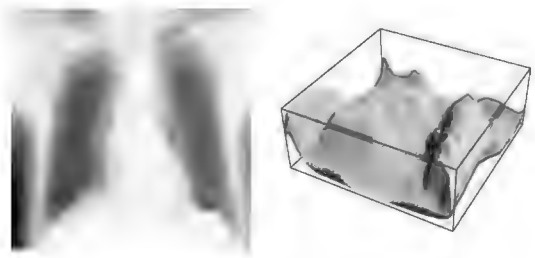


Figure 6.25 Left: An X-thorax image (resolution 256^2) and its 'intensity landscape' at $\sigma = 2$ pixels (right).

We investigate the role of the coefficients in this second order polynomial. In the graph below we vary all three coefficients. In the three groups of 9 plots the value of the mixed coefficient L_{xy} has been varied (value -1, 0 and 1). In each group the 'pure' order terms L_{xx} and L_{yy} are varied (values -1, 0 and +1). In the middle group we see concave, convex, cylindrical and saddle shapes.

```
Show[GraphicsArray[
  Table[GraphicsArray[Table[Plot3D[Lxx/2 x^2 + Lxy x y + Lyy/2 y^2, {x, -3, 3},
    {y, -3, 3}, PlotRange → {-18, 18}, AspectRatio → 1,
    DisplayFunction → Identity, Boxed → True, Mesh → False],
  {Lxx, -1, 1}, {Lyy, -1, 1}], Frame → True], {Lxy, -1, 1}]], ImageSize → 480];
```

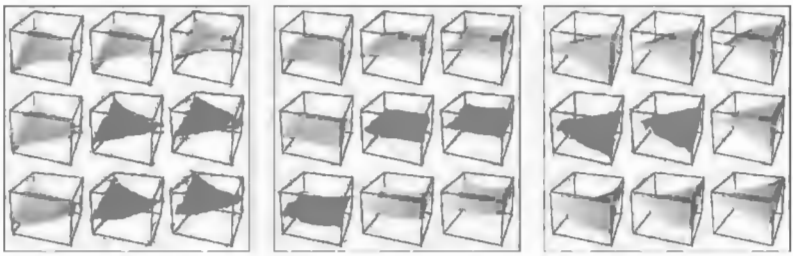


Figure 6.26 Plots of $\frac{L_{xx}}{2} x^2 + L_{xy}\, x\, y + \frac{L_{yy}}{2} y^2$. Left: $L_{xy} = -1$. Middle: $L_{xy} = 0$. Right: $L_{xy} = 1$. In each frame: upper row: $L_{xx} = 1$, middle row: $L_{xx} = 0$, lower row: $L_{xx} = -1$, left column: $L_{yy} = -1$, middle column: $L_{yy} = 0$, right row: $L_{yy} = 1$.

When three variables are at steak, and a visual impression may give valuable insight, one can exploit the trichromacy of our vision. We employ the *invariant* second order derivatives, L_{vv}, L_{vw} and L_{ww}. This shows the triple $\{L_{vv}, L_{vw}, L_{ww}\}$ as RGBColor$[L_{vv}, L_{vw}, L_{ww}]$ color directive settings in each pixel. The color coefficients for this function need to be scaled between 0 and 255.

```
im = Import["thorax02.gif"]; σ = 5; impix = im[[1, 1]]; imcolor = im;
min = Min[color = Transpose[{gauge2DN[impix, 2, 0, σ],
      gauge2DN[impix, 1, 1, σ], gauge2DN[impix, 0, 2, σ]}, {3, 1, 2}]];
max = Max[color - min]; imcolor[[1, 1]] = N[(color - min)/max 255];
imcolor[[1, 4]] = ColorFunction → RGBColor;
DisplayTogetherArray[Show /@ {im, imcolor}, ImageSize → 400];
```

Figure 6.27 Left: An X-thorax image (resolution 256^2) and a mapping of the triple of invariant second order derivatives $\{L_{vv}, L_{vw}, L_{ww}\}$ on the RGB coordinates in each pixel.

6.8.1 The Hessian matrix and principal curvatures

At any point on the surface we can step into an infinite number of directions away from the point, and in each direction we can define a curvature. So in each point an infinite number of curvatures can be defined. It runs out that the curvatures in opposite directions are always the same. Secondly, when we smoothly change direction, there are two (opposite) directions where the curvature is maximal, and there are two (opposite) directions where the curvature is minimal. These directions are perpendicular to each other, and the extremal curvatures are called the *principal curvatures*.

The Hessian matrix is the gradient of the gradient vectorfield. The coefficients form the *second order structure matrix*, or the *Hessian matrix*, also known as the *shape operator* [Gray1993]. The Hessian matrix is a square, symmetric matrix:

```
hessian2D = ( ∂x,x L[x, y]   ∂x,y L[x, y] )
            ( ∂x,y L[x, y]   ∂y,y L[x, y] );
```

The Hessian matrix is square and symmetric, so we can bring it in diagonal form by calculating the Eigenvalues of the matrix and put these on the diagonal elements:

```
DiagonalMatrix[Eigenvalues[hessian2D]] // shortnotation
```

$$\left\{\left\{\frac{1}{2}\left(L_{xx}+L_{yy}-\sqrt{L_{xx}^2+4\,L_{xy}^2-2\,L_{xx}\,L_{yy}+L_{yy}^2}\right),\ 0\right\},\right.$$
$$\left.\left\{0,\ \frac{1}{2}\left(L_{xx}+L_{yy}+\sqrt{L_{xx}^2+4\,L_{xy}^2-2\,L_{xx}\,L_{yy}+L_{yy}^2}\right)\right\}\right\}$$

These special values are the *principal curvatures* of that point of the surface. In the diagonal form the Hessian matrix is rotated in such a way, that the curvatures are maximal and minimal. The principal curvature *directions* are given by the Eigenvectors of the Hessian matrix, found by solving the *characteristic equation* $|H-\kappa I|=0$ for κ, where $|\ldots|$ denotes the determinant, and I is the identity matrix (all diagonal elements are 1, rest zeros).

```
κ =.; Solve[Det[hessian2D - κ IdentityMatrix[2]] == 0, κ] // shortnotation
```

$$\left\{\left\{\kappa\to\frac{1}{2}\left(L_{xx}+L_{yy}-\sqrt{L_{xx}^2+4\,L_{xy}^2-2\,L_{xx}\,L_{yy}+L_{yy}^2}\right)\right\},\right.$$
$$\left.\left\{\kappa\to\frac{1}{2}\left(L_{xx}+L_{yy}+\sqrt{L_{xx}^2+4\,L_{xy}^2-2\,L_{xx}\,L_{yy}+L_{yy}^2}\right)\right\}\right\}$$

The command to calculate Eigenvalues is built into *Mathematica*:

```
{κ1, κ2} = Eigenvalues[hessian2D] // FullSimplify;
{κ1, κ2} // shortnotation
```

$$\left\{\frac{1}{2}\left(L_{xx}-\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}+L_{yy}\right),\ \frac{1}{2}\left(L_{xx}+\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}+L_{yy}\right)\right\}$$

The two principal curvatures are equal when $4\,{L_{xy}}^2+(L_{yy}-L_{xx})^2$ is zero. This happens in so-called *umbilical points*. In umbilical points the principal directions are undefined. The surface is locally *spherical*. The term $4\,{L_{xy}}^2+(L_{yy}-L_{xx})^2$ can be interpreted as 'deviation from sphericalness'.

6.8.2 The shape index

When the principal curvatures κ_1 and κ_2 are considered coordinates in a 2D 'shape graph', we see that all different second order shapes are represented. Each shape is a point on this graph. The following list gives some possibilities:

When both curvatures are zero we have the *flat* shape.
When both curvatures are positive, we have *concave* shapes.
When both curvatures are negative, we have *convex* shapes.
When both curvatures the same sign and magnitude: *spherical* shapes.
When the curvatures have opposite sign: *saddle* shapes.
When one curvature is zero: *cylindrical* shapes.

Koenderink proposed to call the angle, of where the shape vector points to, the *shape index*. It is defined as:

$$\text{shapeindex} \equiv \frac{2}{\pi} \arctan \frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2}, \kappa_1 \geq \kappa_2 .$$

The expression for $\frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2}$ can be markedly cleaned up:

```
Simplify[(κ1 + κ2)/(κ1 - κ2)] // shortnotation
```

$$\frac{-L_{xx} - L_{yy}}{\sqrt{4\,L_{xy}^2 + (L_{xx} - L_{yy})^2}}$$

so we get for the shape index:

$$\text{shapeindex} \equiv \frac{2}{\pi} \arctan\left(\frac{-L_{xx} - L_{yy}}{\sqrt{L_{xx}^2 + 4 L_{xy}^2 - 2 L_{xx} L_{yy} + L_{yy}^2}} \right).$$

The shape index runs from -1 (*cup*) via the shapes *trough*, *rut*, and *saddle rut* to zero, the saddle (here the shape index is undefined), and the goes via *saddle ridge*, *ridge*, and *dome* to the value of +1, the *cap*.

The length of the vector defines how curved a shape is, which gives Koenderink's definition of *curvedness*:

$$\text{curvedness} \equiv \frac{1}{2} \sqrt{\kappa_1^2 + \kappa_2^2} .$$

```
1/2 Sqrt[κ1^2 + κ2^2] // Simplify // shortnotation
```

$$\frac{1}{2} \sqrt{L_{xx}^2 + 2\,L_{xy}^2 + L_{yy}^2}$$

```
shapes =
  Table[GraphicsArray[Table[Plot3D[κ1 x^2 + κ2 y^2, {x, -3, 3}, {y, -3, 3},
      PlotRange → {-18, 18}, PlotLabel ->
        "κ1=" <> ToString[κ1] <> ", κ2=" <> ToString[κ2], AspectRatio → 1,
      DisplayFunction → Identity, Boxed → True, Mesh → False],
    {κ2, 1, -1, -1}, {κ1, -1, 1}]]];
```

```
Show[
  GraphicsArray[{Graphics[{Arrow[{0, 0}, {.7, .5}], Red, PointSize[.02],
      Point[{.7, .5}]}, PlotRange → {{-1, 1}, {-1, 1}},
    Frame → True, Axes → True, AxesLabel → {"κ1", "κ2"},
    AspectRatio → 1], shapes}], ImageSize → 450];
```

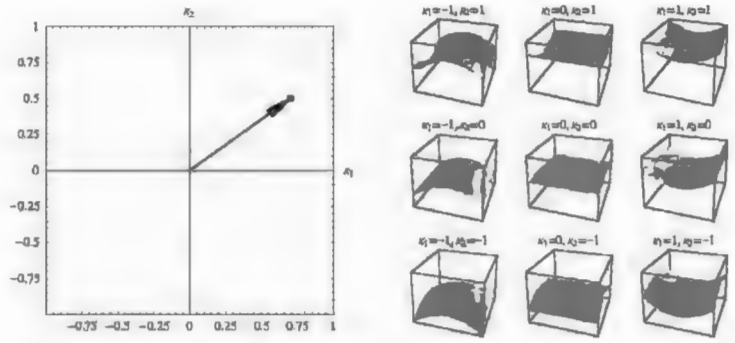


Figure 6.28 Left: Coordinate space of the shape index. Horizontal axis: maximal principal curvature κ_1, vertical axis: minimal principal curvature κ_2. The angle of the position vector determines the shape, the length the curvedness. Right: same as middle set of figure 6.22.

Here is the shape index calculated and plotted for every pixel on our familiar MR image at a scale of σ=3 pixels:

```
im = Import["mr128.gif"][[1, 1]];
shapeindex[im_, σ_] := 2/π ArcTan[(-gD[im, 2, 0, σ] - gD[im, 0, 2, σ]) /
    (√(gD[im, 2, 0, σ]^2 + 4 gD[im, 1, 1, σ]^2 -
        2 gD[im, 2, 0, σ] gD[im, 0, 2, σ] + gD[im, 0, 2, σ]^2))];
DisplayTogetherArray[ListDensityPlot[shapeindex[im, #]] & /@ Range[5],
  ImageSize → 400];
```

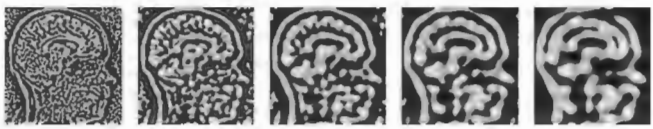


Figure 6.29 Shape index of the sagittal MR image at σ = 1, 2, 3, 4 and 5 pixels.

```
curvedness[im_, σ_] :=
  √(gD[im, 2, 0, σ]^2 + 2 gD[im, 1, 1, σ]^2 + gD[im, 0, 2, σ]^2);
DisplayTogetherArray[ListDensityPlot[curvedness[im, #]] & /@ Range[4],
  ImageSize → 400];
```

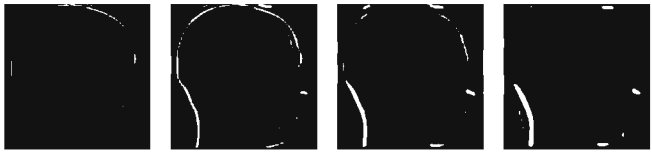


Figure 6.30 Curvedness of the sagittal MR image at σ = 1, 2, 3 and 4 pixels.

6.8.3 Principal directions

The principal curvature directions are given by the Eigenvectors of the Hessian matrix:

```
{vκ1, vκ2} = Eigenvectors[hessian2D]; {vκ1, vκ2} // shortnotation
```

$$\left\{\left\{-\frac{-L_{xx}+L_{yy}+\sqrt{L_{xx}^2+4L_{xy}^2-2L_{xx}L_{yy}+L_{yy}^2}}{2L_{xy}},\ 1\right\},\right.$$
$$\left.\left\{\frac{L_{xx}-L_{yy}+\sqrt{L_{xx}^2+4L_{xy}^2-2L_{xx}L_{yy}+L_{yy}^2}}{2L_{xy}},\ 1\right\}\right\}$$

The Eigenvectors are perpendicular to each other, there inner product is zero:

```
vκ1.vκ2 // Simplify
```

```
0
```

The local principal direction vectors form locally a *frame*. We inspect how the orientations of such frames are distributed in an image. We orient the frame in such a way that the largest Eigenvalue (maximal principal curvature) is one direction, the minimal principal curvature direction is $\pi/2$ rotated clockwise.

```
plotprincipalcurvatureframes[im_, σ_] :=
 Module[{hessian, frame, frames},
  hessian = ( gD[im, 2, 0, σ]  gD[im, 1, 1, σ]
              gD[im, 1, 1, σ]  gD[im, 0, 2, σ] );
frame = {Green, Arrow[#2 - .5, #2 - .5 + First[#1]],
     Red, Arrow[#2 - .5, #2 - .5 + Last[#1]]} &;
frames = MapIndexed[frame, .5 Map[Eigenvectors,
      Transpose[hessian, {4, 3, 2, 1}], {2}], {2}];
plot = ListDensityPlot[gD[im, 0, 0, σ], Epilog → frames]]
im = Import["mr32.gif"][[1, 1]];
```

```
plotprincipalcurvatureframes[im, 1];
```

Figure 6.31 Frames of the normalized principal curvature directions at a scale of 1 pixel. Image resolution 32^2 pixels. Green: maximal principal curvature direction; red: minimal principal curvature direction.

The principal curvatures have been employed by Niessen, ter Haar Romeny and Lopéz in studies to the 2D and 3D structure of trabecular bone [TerHaarRomeny1996f, Niessen1997b, Lopéz200a]. The local structure was defined as *flat* when the two principal curvatures of the iso-intensity surface in 3D were are both small, as *rod-like* if one of the curvatures was small and the other high, giving a local cylindrical shape, and *sphere-like* if two principal curvatures were both high. See also Task 19.8.

6.8.4 Gaussian and mean curvature

The Gaussian curvature $\mathcal{K}$ is defined as the product of the two principal curvatures: $\mathcal{K} = \kappa_1 \kappa_2$.

```
𝒦 = κ1 κ2 // Simplify // shortnotation
```

$$-L_{xy}^2 + L_{xx} L_{yy}$$

The Gaussian curvature is equal to the determinant of the Hessian matrix:

```
Det[hessian2D] // shortnotation
```

$$-L_{xy}^2 + L_{xx} L_{yy}$$

The sign of the Gaussian curvature determines if we are in a concave / convex area (positive Gaussian curvature) or in a saddle-like area (negative Gaussian curvature). This shows saddle-like areas as dark patches:

```
im = Import["mr256.gif"][[1, 1]];
σ = 5; K = -gD[im, 1, 1, σ]^2 + gD[im, 2, 0, σ] gD[im, 0, 2, σ];
DisplayTogetherArray[Append[ListDensityPlot /@ {K, Sign[K]},
   ListContourPlot[K, Contours → {0}]], ImageSize → 390];
```

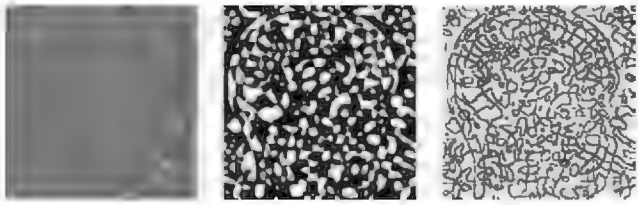


Figure 6.32 Left: Gaussian curvature $\mathcal{K}$ for a 256^2 sagittal MR image at a scale of 5 pixels. Middle: sign of $\mathcal{K}$. Right: zerocrossings of $\mathcal{K}$.

The locations where the Gaussian curvature is zero, are characterized by the fact that at least one of the principal curvatures is zero. The collection of locations where the Gaussian curvature is zero is known as the *parabolic lines*. It was shown by Koenderink that these lines play an important role in reflection and shape-from-shading.

The mean curvature is defined as the arithmetic mean of the principal curvatures: $\mathcal{H} = \frac{\kappa_1 + \kappa_2}{2}$.

The mean curvature is related to the trace of the Hessian matrix:

```
H = (κ1 + κ2)/2 // Simplify // shortnotation
```

$$\frac{1}{2}\,(L_{xx} + L_{yy})$$

```
Tr[hessian2D] // shortnotation
```

$$L_{xx} + L_{yy}$$

The relation between the mean curvature $\mathcal{H}$ and the Gaussian curvature $\mathcal{K}$ is given by $\kappa^2 - 2\mathcal{H}\kappa + \mathcal{K} = 0$, which has solutions:

```
H =.; K =.; Solve[κ^2 - 2 H κ + K == 0, κ]
```

$$\left\{\left\{\kappa \to \mathcal{H} - \sqrt{\mathcal{H}^2 - \mathcal{K}}\right\},\ \left\{\kappa \to \mathcal{H} + \sqrt{\mathcal{H}^2 - \mathcal{K}}\right\}\right\}$$

The mean curvature $\mathcal{H}$ and the Gaussian curvature $\mathcal{K}$ are well defined in umbilical points.

The directional derivative of the principal curvature in the direction of the principal direction is called the *extremality* [Monga1995].

Because there are two principal curvatures, there are two extremalities, $\overrightarrow{\nabla\kappa_1}.\vec{\nabla}\kappa_1$ and $\overrightarrow{\nabla\kappa_2}.\vec{\nabla}\kappa_2$:

```
<< Calculus`VectorAnalysis`;
```

```
e1 = vκ1.Take[Grad[{κ1, 0, 0}, Cartesian[x, y, z]], 2] // FullSimplify;
e1 // shortnotation
```

$$\left\{-\frac{1}{4\,L_{xy}}\left(\left(L_{xxx}+L_{xyy}+\frac{-4\,L_{xxy}\,L_{xy}-(L_{xxx}-L_{xyy})\,(L_{xx}-L_{yy})}{\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}}\right)\left(-L_{xx}+\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}+L_{yy}\right)\right),\right.$$

$$\left.-\frac{1}{4\,L_{xy}}\left(\left(-L_{xx}+\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}+L_{yy}\right)\left(L_{xxy}+\frac{-4\,L_{xy}\,L_{xyy}-(L_{xx}-L_{yy})\,(L_{xxy}-L_{yyy})}{\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}}+L_{yyy}\right)\right),\ 0\right\}$$

```
e2 = vκ2.Take[Grad[{κ2, 0, 0}, Cartesian[x, y, z]], 2] // FullSimplify;
e2 // shortnotation
```

$$\left\{-\frac{1}{4\,L_{xy}}\left(\left(L_{xxx}+L_{xyy}+\frac{4\,L_{xxy}\,L_{xy}+(L_{xxx}-L_{xyy})\,(L_{xx}-L_{yy})}{\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}}\right)\left(-L_{xx}-\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}+L_{yy}\right)\right),\right.$$

$$\left.-\frac{1}{4\,L_{xy}}\left(\left(-L_{xx}-\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}+L_{yy}\right)\left(L_{xxy}+\frac{4\,L_{xy}\,L_{xyy}+(L_{xx}-L_{yy})\,(L_{xxy}-L_{yyy})}{\sqrt{4\,L_{xy}^2+(L_{xx}-L_{yy})^2}}+L_{yyy}\right)\right),\ 0\right\}$$

The lines defined by the zerocrossings of each of these two extremalities are called the *extremal lines* [Thirion1995a, Thirion1996]. There are 4 types of these lines:
- lines of maximum largest principal curvature (these are called *crest lines*);
- lines of minimum largest principal curvature;
- lines of maximum smallest principal curvature;
- lines of minimum smallest principal curvature.

The product of the two extremalities is called the *Gaussian extremality* $\mathcal{G} = e_1.e_2$, a true local invariant [Thirion1996]. The boundaries of the regions where the Gaussian extremality changes sign, are the extremal lines.

```
e1.e2 // Simplify // shortnotation
```

$$-(L_{xy}^2\,(L_{xxx}^2+2\,L_{xxx}\,L_{xyy}-3\,(L_{xxy}^2+L_{xyy}^2))+L_{xxx}\,L_{xyy}\,(L_{xx}-L_{yy})^2+2\,L_{xxx}\,L_{xxy}\,L_{xy}\,(-L_{xx}+L_{yy})+(L_{xx}^2\,L_{xxy}-2\,L_{xy}\,L_{xyy}\,L_{yy}+2\,L_{xx}\,(L_{xy}\,L_{xyy}-L_{xxy}\,L_{yy})+L_{xxy}\,(2\,L_{xy}^2+L_{yy}^2))\,L_{yyy}+L_{xy}^2\,L_{yyy}^2)\,/\,(L_{xx}^2+4\,L_{xy}^2-2\,L_{xx}\,L_{yy}+L_{yy}^2)$$

```
DisplayTogetherArray[
  ListDensityPlot[Sign[e1.e2 /. Derivative[nx_, ny_][L][x_, y_] →
        gD[im0, nx, ny, #] /. im0 → im]] & /@ {2, 6, 10}, ImageSize → 400];
```

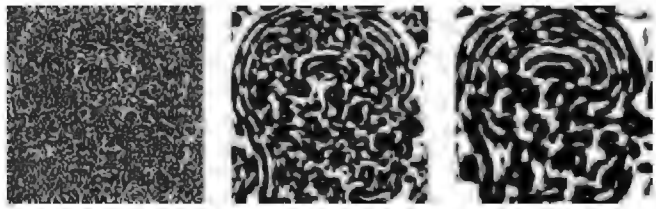


Figure 6.33 Left: Gaussian extremality $\mathcal{G} = e_1 e_2$ for a 256^2 sagittal MR image at a scale of 2 pixels (left), 6 pixels (middle) and 10 pixels (right).

The mesh that these lines form on an iso-intensity surface in 3D is called the *extremal mesh*. It has been applied for 3D image registration, by extracting the lines with a dedicated 'marching lines' algorithm [Thirion1996].

```
Show[Import["extremal mesh - Thirion.jpg"], ImageSize → 250];
```

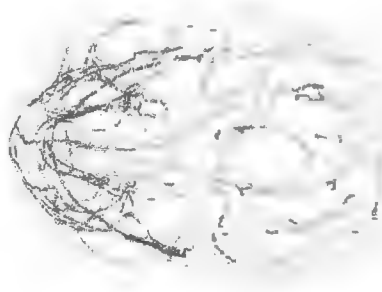


Figure 6.34 Extremal mesh on a 3D skull from a 3D CT dataset. The extremal lines are found with the marching lines algorithm. From [Thirion1993].

6.8.5 Minimal surfaces and zero Gaussian curvature surfaces

Surfaces that have everywhere mean curvature zero, are called *minimal surfaces*. There are many beautiful examples of such surfaces (see e.g. the Scientific Grahics Project, http://www.msri.org/publications/sgp/SGP/indexc.html. Soap bubbles are famous and much studied examples of minimal surfaces.

From the wonderful interactive book by Alfred Gray [Gray1993] (written in *Mathematica*) we plot two members of the family of zero Gaussian curvature manifolds that can be constructed by a moving straight line through 3D space:

```
heltocat[t_][u_, v_] := Cos[t] {Sinh[v] Sin[u], -Sinh[v] Cos[u], u} +
   Sin[t] {Cosh[v] Cos[u], Cosh[v] Sin[u], v};
moebiusstrip[u_, v_] := {Cos[u] + v Cos[u / 2] Cos[u],
   Sin[u] + v Cos[u / 2] Sin[u], v Sin[u / 2]};
DisplayTogetherArray[{ParametricPlot3D[
    Evaluate[heltocat[0][u, v]], {u, -π, π}, {v, -π, π},
    PlotPoints -> 30, Axes -> None, BoxRatios -> {1, 1, 1},
    PlotRange -> {{-13, 13}, {-13, 13}, {-π, π}}],
ParametricPlot3D[moebiusstrip[u, v] // Evaluate,
    {u, 0, 2 Pi}, {v, -.3, .3}, PlotPoints -> {30, 4},
Axes -> None]}, ImageSize → 390];
```

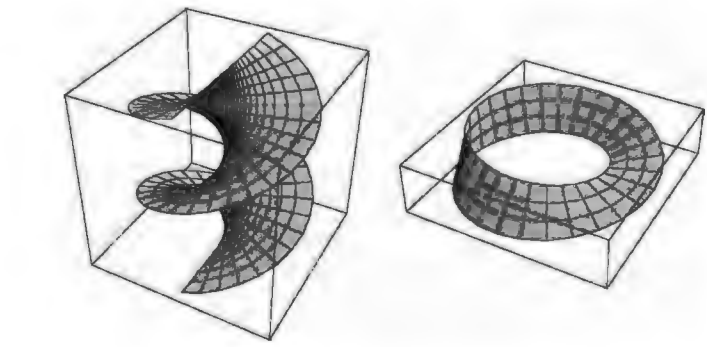


Figure 6.35 Surfaces with zero Gaussian curvature. Left the helicoid, a member of the heltocat family. Right the Moebius strip. Both surfaces can be constructed by a moving straight line. From [Gray1993].

▲ Task 6.9 Which surfaces have constant mean curvature? And which surfaces have constant Gaussian curvature?

▲ Task 6.10 If I walk with my principal coordinate frame over an egg, something goes wrong when I walk through an umbilical point. What?

6.9 Third order image structure: T-junction detection

An example of third order *geometric reasoning* in images is the detection of T-junctions [TerHaarRomeny1991a, TerHaarRomeny1993b]. T-junctions in the intensity landscape of natural images occur typically at occlusion points. Occlusion points are those points where a contour ends or emerges because there is another object in front of the contour. See for an artistic example the famous painting 'the blank cheque' by Magritte.

```
Show[Import["blank cheque.jpg"], ImageSize -> 210];
```

Figure 6.36 The painting 'the blank cheque' by the famous Belgian surrealist painter René Magritte (1898 - 1967). Source: Paleta (www.paletaworld.org).

In this section we develop a third order detector for "T-junction-likeniness". In the figure below the circles indicate a few particular T-junctions:

```
blocks = Import["blocks.gif"][[1, 1]];
ListDensityPlot[blocks,
  Epilog -> (circles = {Circle[{221, 178}, 13], Circle[{157, 169}, 13],
      Circle[{90, 155}, 13], Circle[{148, 56}, 13],
      Circle[{194, 77}, 13], Circle[{253, 84}, 13]}), ImageSize -> 300];
```

Figure 6.37 T-junctions often emerge at occlusion boundaries. The foreground edge is most likely to be the straight edge of the "T", with the occluded edge at some angle to it. The circles indicate some T-junctions in the image.

When we zoom in on the T-junction of an observed image and inspect locally the isophote structure at a T-junction, we see that at a T-junction the derivative of the isophote curvature κ

in the direction perpendicular to the isophotes is high. In the figure below the isophote landscape of a blurred T-junction illustrates the direction of maximum change of κ:

```
im = Table[If[y < 64, 0, 1] + If[y < x && y > 63, 2, 1], {y, 128}, {x, 128}];
DisplayTogetherArray[ListDensityPlot[im],
  ListContourPlot[gD[im, 0, 0, 7], Contours → 15,
   PlotRange → {-0.3, 2.8}], ImageSize -> 280];
```

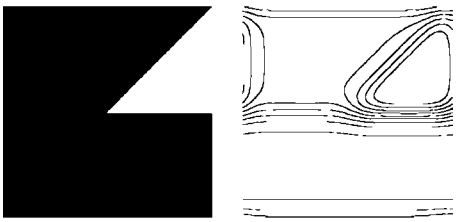


Figure 6.38 The isophote structure (right) of a simple idealized and observed (blurred) T-junction (left) shows that isophotes strongly bend at T-junctions when we walk through the intensity landscape.

When we study the curvature of the isophotes in the middle of the image, at the location of the T-junction, we see the isophote 'sweep' from highly curved to almost straight for decreasing intensity. So the geometric reasoning is the "the isophote curvature changes a lot when we traverse the image in the w direction". It seems to make sense to study $\frac{\partial \kappa}{\partial w}$:

We recall that the isophote curvature κ is defined as $\kappa = -\frac{L_{vv}}{L_w}$:

```
κ = gauge2D[L[x, y], 2, 0] / gauge2D[L[x, y], 0, 1]; κ // Simplify // shortnotation
```

$$\frac{-2\,L_x\,L_{xy}\,L_y + L_{xx}\,L_y^2 + L_x^2\,L_{yy}}{(L_x^2 + L_y^2)^{3/2}}$$

The derivative of the isophote curvature in the direction of the gradient, $\frac{\partial \kappa}{\partial w}$ is quite a complex third order expression. The formula is derived by calculating the *directional derivative* of the curvature in the direction of the normalized gradient. We define the gradient (or *nabla*: ∇) operator with a pure function:

```
grad = {∂x #, ∂y #} &;
dκdw = grad[L[x, y]] / Sqrt[grad[L[x, y]].grad[L[x, y]]].grad[κ];
dκdw // Simplify // shortnotation
```

$$\frac{1}{(L_x^2 + L_y^2)^3}$$
$$(L_{xxy}\,L_y^5 + L_x^4\,(-2\,L_{xy}^2 + L_x\,L_{xyy} - L_{xx}\,L_{yy}) - L_y^4\,(2\,L_{xy}^2 - L_x\,(L_{xxx} - 2\,L_{xyy}) + L_{xx}\,L_{yy}) +$$
$$L_x^2\,L_y^2\,(-3\,L_{xx}^2 + 8\,L_{xy}^2 + L_x\,(L_{xxx} - L_{xyy}) + 4\,L_{xx}\,L_{yy} - 3\,L_{yy}^2) +$$
$$L_x^3\,L_y\,(6\,L_{xy}\,(L_{xx} - L_{yy}) + L_x\,(-2\,L_{xxy} + L_{yyy})) +$$
$$L_x\,L_y^3\,(6\,L_{xy}\,(-L_{xx} + L_{yy}) + L_x\,(-L_{xxy} + L_{yyy})))$$

To avoid singularities at vanishing gradients through the division by $(L_x{}^2 + L_y{}^2)^3 = L_w{}^6$ we use as our T-junction detector $\tau = \frac{\partial \kappa}{\partial w} L_w{}^6$:

```
tjunction = dκdw (grad[L[x, y]].grad[L[x, y]])^3;
tjunction // shortnotation
```

$$
\begin{aligned}
&L_x^5\, L_{xyy} + L_y^4\, (-2\, L_{xy}^2 + L_{xxy}\, L_y - L_{xx}\, L_{yy}) + \\
&L_x^3\, L_y\, (6\, L_{xx}\, L_{xy} + L_{xxx}\, L_y - L_{xyy}\, L_y - 6\, L_{xy}\, L_{yy}) + \\
&L_x\, L_y^3\, (-6\, L_{xx}\, L_{xy} + L_{xxx}\, L_y - 2\, L_{xyy}\, L_y + 6\, L_{xy}\, L_{yy}) - \\
&L_x^4\, (2\, L_{xy}^2 + 2\, L_{xxy}\, L_y + L_{xx}\, L_{yy} - L_y\, L_{yyy}) + \\
&L_x^2\, L_y^2\, (-3\, L_{xx}^2 + 8\, L_{xy}^2 - L_{xxy}\, L_y + 4\, L_{xx}\, L_{yy} - 3\, L_{yy}^2 + L_y\, L_{yyy})
\end{aligned}
$$

Finally, we apply the T-junction detector on our blocks at a rather fine scale of $\sigma = 2$ (we plot `-tjunction` to invert the contrast):

```
σ = 2; ListDensityPlot[
 tjunction /. Derivative[nx_, ny_][L][x, y] -> gD[im0, nx, ny, σ] /.
  im0 -> blocks, Epilog -> circles, ImageSize -> 230];
```

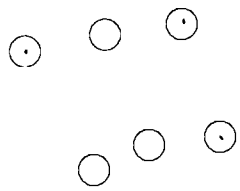


Figure 6.39 Detection of T-junctions in the image of the blocks with the detector $\tau = \frac{\partial \kappa}{\partial w} L_w{}^6$. The same circles have been drawn as in figure 6.32.

Compare the detected points with the circles in the input image. Note that in medical *tomographic* images (CT, MR, PET, SPECT, US) there is no occlusion present. One can however use third order properties in any *geometric reasoning* scheme, as the 'change of a second order property'.

▲ Task 6.11 Investigate if the expression for the T-junction $\tau = \frac{\partial \kappa}{\partial w} L_w{}^6$ is affine invariant.

▲ Task 6.12 Another definition for a T-junction detector might be the magnitude o the gradient of the curvature: $\tau = \sqrt{(\frac{\partial \kappa}{\partial w})^2 + (\frac{\partial \kappa}{\partial w})^2}\, L_w{}^6$, or the derivative of the curvature in the v direction: $\frac{\partial \kappa}{\partial v} L_w{}^6$. Study and explain the differences.

6.10 Fourth order image structure: junction detection

As a final fourth order example, we give an example for a detection problem in images at high order of differentiation from algebraic theory. Even at orders of differentiation as high as 4, invariant features can be constructed and calculated for discrete images through the biologically inspired scaled derivative operators. Our example is to find in a checkerboard the crossings where 4 edges meet. We take an algebraic approach, which is taken from Salden et al. [Salden1999a].

When we study the fourth order local image structure, we consider the fourth order polynomial terms from the local Taylor expansion:

```
pol4 = Lxxxx x^4 + 4 Lxxxy x^3 y + 6 Lxxyy x^2 y^2 + 4 Lxyyy x y^3 + Lyyyy y^4;
```

The main theorem of algebra states that a polynomial is fully described by its roots: e.g. $ax^2 + bx + c = (x - x_1)(x - x_2)$. It was shown more than a century ago by Hilbert [Hilbert1890] that the 'coincidencesness' of the roots, or how well all roots coincide, is a particular invariant condition. From algebraic theory it is known that this 'coincidenceness' is given by the *discrininant*, defined below (see also [Salden1999a]):

```
Discriminant[p_, x_] :=
 With[{m = Exponent[p, x]}, Cancel[(-1)^(1/2 m (m-1)) Resultant[p, ∂x p, x] / Coefficient[p, x, m]]]
```

The resultant of two polynomials a and b, both with leading coefficient one, is the product of all the differences $a_i - b_j$ between roots of the polynomials. The resultant is always a number or a polynomial. The discriminant of a polynomial is the product of the squares of all the differences of the roots taken in pairs. We can express our function in two variables $\{x, y\}$ as a function in a single variable $\frac{x}{y}$ by the substitution $y \to 1$. Some examples:

```
Discriminant[Lxx x^2 + 2 Lxy x y + Lyy y^2, x] /. {y → 1}

-4 (-Lxy^2 + Lxx Lyy)
```

The discriminant of second order image structure is just the determinant of the Hessian matrix, i.e. the Gaussian curvature. Here is our fourth order discriminant:

```
Discriminant[pol4, x] /. {y → 1} // Simplify

256 (-27 Lxxxy^4 Lyyyy^2 + Lxxxy^3 (-64 Lxyyy^3 + 108 Lxxyy Lxyyy Lyyyy) -
   12 Lxxxx Lxxxy Lxyyy (-9 Lxxyy Lxyyy^2 + 15 Lxxyy^2 Lyyyy + Lxxxx Lyyyy^2) -
   6 Lxxxy^2 (-6 Lxxyy^2 Lxyyy^2 + 9 Lxxyy^3 Lyyyy +
      Lxxxx Lxyyy^2 Lyyyy - 9 Lxxxx Lxxyy Lyyyy^2) +
   Lxxxx (-54 Lxxyy^3 Lxyyy^2 + 81 Lxxyy^4 Lyyyy + 54 Lxxxx Lxxyy Lxyyy^2 Lyyyy -
      18 Lxxxx Lxxyy^2 Lyyyy^2 + Lxxxx (-27 Lxyyy^4 + Lxxxx Lyyyy^3)))
```

It looks like an impossibly complicated polynomial in fourth order derivative images, and it is. Through the use of Gaussian derivative kernels each separate term can easily be

calculated. We replace (with the operator /.) all the partial derivatives into scaled Gaussian derivatives:

```
discr4[im_, σ_] := Discriminant[pol4, x] /.
  {y → 1, Lxxxx → gD[im, 4, 0, σ], Lxxxy → gD[im, 3, 1, σ],
   Lxxyy → gD[im, 2, 2, σ], Lxyyy → gD[im, 1, 3, σ], Lyyyy → gD[im, 0, 4, σ]}
```

Let us apply this high order function on an image of a checkerboard, and we add noise with twice the maximum image intensity to show its robustness, despite the high order derivatives (see figure 6.40).

Note that we have a highly symmetric situation here: the four edges that come together at the checkerboard vertex cut the angle in four. The symmetry can be seen in the complex expression for discr4: only pure partial derivatives of fourth order occur. For a less symmetric situation we need a detector which incorporates in its expression also the lower order partial derivatives. For details see [Salden1999a].

```
t1 = Table[If[(x > 50 && y > 50) || (x ≤ 50 && y ≤ 50), 0, 100] + 200 * Random[],
    {x, 100}, {y, 100}];
t2 = Table[If[(x + y - 100 > 0 && y - x < 0) || (x + y - 100 ≤ 0 && y - x ≥ 0),
     0, 100] + 200 * Random[], {x, 100}, {y, 100}];

noisycheck = Transpose[Join[t1, t2]];
DisplayTogetherArray[ListDensityPlot /@
   {noisycheck, discr4[noisycheck, 5]}, ImageSize -> 400];
```

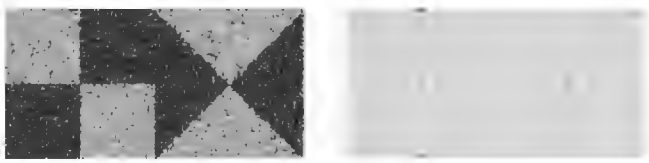


Figure 6.40 Left: A noisy checkerboard detail at two orientations. Right: the output of the 4th order discriminant. The detection clearly is rotation invariant, robust to noise, and there is no detection at corners (e.g. center of the image).

6.11 Scale invariance and natural coordinates

The intensity of images and invariant features at larger scale decreases fast. This is due to the non-scale-invariant use of the differential operators. For, if we consider the transformation $\frac{x}{\sigma} \rightarrow \hat{x}$, then $\hat{x}$ is dimensionless. At every scale now distances are measured with a distance yardstick with is scaled relative to the scale itself. This is the scale-invariance.

The dimensionless coordinate is termed the *natural* coordinate. This implies that the derivative operator in natural coordinates has a scaling factor: $\frac{\partial^n}{\partial \hat{x}^n} \rightarrow \sigma^n \frac{\partial^n}{\partial x^n}$.

Here we generate a scale-space of the intensity gradient. To study the absolute intensities, we plot every image with the same intensity plotrange of {0,40}:

```
im = Import["mr128.gif"][[1, 1]]; Block[{$DisplayFunction = Identity},
 pl = Table[grad = √ (gD[im, 1, 0, σ]^2 + gD[im, 0, 1, σ]^2);
   ListDensityPlot[#, PlotRange -> {0, 40}] & /@ {grad, σ grad}
   , {σ, 1, 5}]]; Show[GraphicsArray[Transpose[pl]], ImageSize → 450];
```

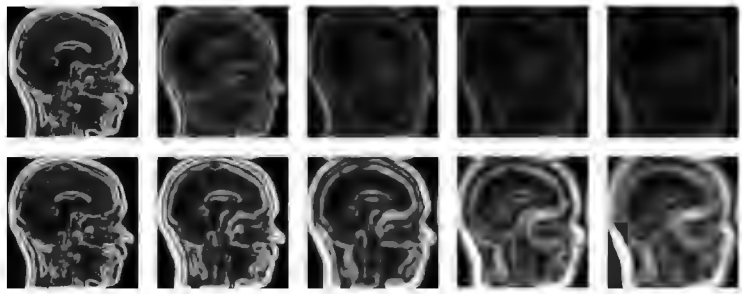


Figure 6.41 The gradient of a 128^2 image plotted at 5 scales, for σ = 1, 2, 3, 4 and 5 pixels respectively. All images (in both rows) are plotted at a fixed intensity range {0,40}. Top row shows the regular gradient, clearly showing the decrease in intensity for larger blurring. Bottom row: the gradient in natural coordinates (multiplied by σ). The intensity dynamic range is now kept more or less constant.

Clearly the gradient magnitude expressed in the natural coordinates keeps its average output range. For a Laplacian scale-space stack in natural coordinates we need to multiply the Laplacian with σ^2: $\frac{\partial^2}{\partial \hat{x}^2} + \frac{\partial^2}{\partial \hat{y}^2} = \sigma^2\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$, and so on for higher order derivative operators in natural coordinates.

```
Block[{$DisplayFunction = Identity},
 pl = Table[lapl = gD[im, 2, 0, σ] + gD[im, 0, 2, σ];
   ListDensityPlot[#, PlotRange -> {-90, 60}] & /@ {lapl, σ^2 lapl}
   , {σ, 1, 5}]]; Show[GraphicsArray[Transpose[pl]], ImageSize → 450];
```

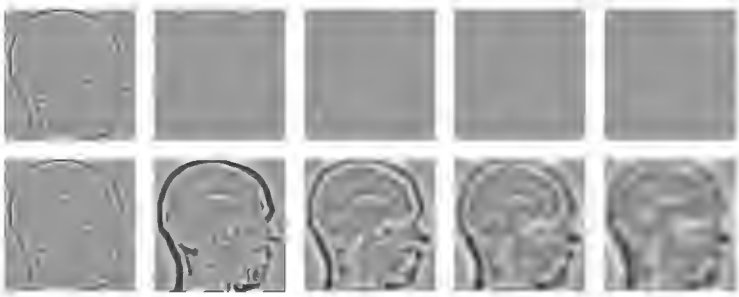


Figure 6.42 The Laplacian of a 128^2 image plotted at 5 scales, for σ = 1, 2, 3, 4 and 5 pixels respectively. Top row: Laplacian in regular coordinates. Bottom row: Laplacian in natural coordinates. Top and bottom rows at fixed intensity range of {-90,60}.

6.12 Irreducible invariants

Invariant differential features are independent of changes in specific groups of coordinate transformations. Note that the transformations of the *coordinates* are involved as the basic physical notion, as the particular choice of coordinates is just a mean to describe the world, the real situation should be independent of this choice. This is often misunderstood, e.g. when rotation invariance is interpreted as that all results are the same when the image itself is rotated. Rotation invariance is a *local* property, and as such a coordinate rotation and an image rotation are only the same when we consider a single point in the image.

For medical imaging the most important groups are the orthogonal transformations, such as translations, rotations, mirroring and scaling, and the affine transformations, such as shear. There are numerous other groups of transformations, but it is beyond the scope of this book to treat this. The differential invariants are the natural building blocks to express local differential structure.

It has been shown by Hilbert [Hilbert1893] that *any invariant of finite order can be expressed as a polynomial function of a set of irreducible invariants*. This is an important result. For e.g. scalar images these invariants form the *fundamental set of image primitives* in which all local intrinsic properties can be described. In other words: any invariant can be expressed in a polynomial combination of the irreducible invariants.

Typically, and fortunately, there are only a small number of irreducible invariants for low order. E.g. for 2D images up to second order there are only 5 of such irreducibles. We have already encountered one mechanism to find the irreducible set: gauge coordinates. We found the following set:

Zeroth order	L
First order	L_w
Second order	L_{vv}, L_{vw}, L_{ww}
Third order	$L_{vvv}, L_{vvw}, L_{vww}, L_{www}$
etc.	

Each of these irreducible invariants cannot be expressed in the others. Any invariant property to some finite order can be expressed as a combination of these irreducibles. E.g. isophote curvature, a second order local invariant feature, is expressed as: $\kappa = -L_{vv}/L_w$.
Note that the first derivative to v is missing. But $L_v \equiv 0$ is just the gauge condition! There is always that one degree of freedom to rotate the coordinate system in such a way that the tangential derivative vanishes. This gives a way to estimate the number of irreducible invariants for a given order: It is equal to the number of partial derivative coefficients in the local Taylor expansion, minus 1 for the gauge condition. E.g. for the 4^{th} order we have the partial derivatives L_{vvvv}, L_{vvvw}, L_{vvww}, L_{vwww}, and L_{wwww}, so in total we have $1+1+3+4+5 = 14$ irreducible invariants for the 4^{th} order.
These irreducibles form a *basis* for the differential invariant structure. The set of 5 irreducible grayvalue invariants in 2D images has been exploited to classify local image structure by Schmidt et al. [Schmidt1996a, Schmidt1996b] for statistical object recognition.

This assigns the three RGB channels of a color image to the irreducible invariants $\{L,\ L_w$ and $L_{vv} + L_{ww}\}$ of a scalar grayvalue image for $\sigma = 2$ pixels:

```
im = Import["mr256.gif"]; px = im[[1, 1]]; σ = 2;
r = gD[px, 0, 0, σ]; g = √(gD[px, 1, 0, σ]^2 + gD[px, 0, 1, σ]^2);
b = gD[px, 2, 0, σ]^2 + gD[px, 0, 2, σ]^2;
g = g 255/Max[g]; b = b 255/Max[b]; imtr = Transpose[{r, g, b}, {3, 1, 2}];
im[[1, 1]] = imtr; im[[1, 4]] = ColorFunction → RGBColor; Show[im, ImageSize -> 150];
```

Figure 6.43 RGB color coding with the triplet of differential invariants $\{L,\ L_i\ L_i,\ L_{ii}\}$.

Intermezzo: Tensor notation

There are many ways to set up an irreducible basis, but it is beyond the scope of this introductory book to go in detail here. We just give one example of another often used scheme to generate irreducible invariants: tensor notation (see for details e.g. [Florack1993a]). Here tensor *indices* denote partial derivatives and run over the dimensions, e.g. L_i denotes the vector $\{L_x, L_y\}$, L_{ij} denotes the second order matrix (the Hessian) $\begin{pmatrix} L_{xx} & L_{xy} \\ L_{yx} & L_{yy} \end{pmatrix}$, etc.

When indices come in pairs, summation over the dimensions is implied (the so-called Einstein summation convention, or *contraction*): $L_{ii} = \sum_{i=x}^{D} L_{ii} = L_{xx} + L_{yy}$, etc. So we get:

Zeroth order	L	
First order	$L_i\ L_i$	$(= L_x\ L_x + L_y\ L_y$, the gradient)
Second order	L_{ii}	$(= L_{xx} + L_{yy}$, the Laplacian)
	$L_{ij}\ L_{ji}$	$(= {L_{xx}}^2 + 2\ {L_{xy}} + {L_{yy}}^2$, the 'deviation from flatness'),
	$L_i\ L_{ij}\ L_j$	$(= {L_x}^2\ L_{xx} + 2\ L_x\ L_y\ L_{xy} + {L_y}^2\ L_{yy}$, 'curvature')

etc.

Some statements by famous physicists:
- "Gauge invariance is a classic case of a good idea which was discovered before its time." (K. Moriyasu, An Elementary Primer for Gauge Theory, World Scientific, 1984).
- "The name 'gauge' comes from the ordinary English word meaning 'measure'. The history of the use of this name for a class of field theories is very roundabout, and has little to do

with their physical significance as we now understand it." (S. Weinberg, "The Forces of Nature", Am. Scientist, 65, 1977).
- "As far as I see, all a priori statements in physics have their origin in symmetry." (H. Weyl, Symmetry, 1952).

6.13 Summary of this chapter

Invariant differential feature detectors are special (mostly) polynomial combinations of image derivatives, which exhibit invariance under some chosen group of transformations. We only discussed invariance under translations and rotations, the most common groups, especially for medical images. The derivatives are easily calculated from the image through the multi-scale Gaussian derivative kernels.

The notion of invariance is crucial for geometric relevance. Non-invariant properties have no value in general feature detection tasks. A convenient paradigm to calculate features invariant under Euclidean coordinate transformations is the notion of *gauge coordinates*. For first order in 2D they are defined as a local frame with one unit vector $\vec{w}$ pointing in the direction of the gradient, the other perpendicular unit vector $\vec{v}$ pointing in the direction tangential to the isophote. Any combination of derivatives with respect to v and w is invariant under Euclidean transformations. We discussed the second order examples of isophote and flowline curvature, cornerness and the third order example of T-junction detection in this framework.

Mathematica offers a particularly attractive framework, in that it combines the analytical calculation of features under the Euclidean invariance condition with a final replacement of the analytical derivatives with numerical Gaussian derivatives. In this way even high order (up to order 4) examples could be discussed and calculated.

7. Natural limits on observations

He who asks a question is a fool for five minutes;
he who does not ask a question remains a fool forever.
Chinese Proverb

7.1 Limits on differentiation: scale, accuracy and order

For a given order of differentiation we find that there is a limiting scale-size below which the results are no longer exact. E.g. when we study the derivative of a ramp with slope 1, we expect the outcome to be correct. Let us look at the *observed* derivative at the center of the image for a range of scales ($0.4 < \sigma < 1.2$ in steps of 0.1):

```
<< FrontEndVision`FEV`; im = Table[x, {y, 64}, {x, 64}];
b = Table[{σ, gDf[im, 1, 0, σ][[32, 32]]}, {σ, .4, 1.2, .1}];
ListPlot[b, PlotJoined → True,
  PlotRange → All, AxesLabel → {"σ", "∂xL"},
  PlotStyle -> Thickness[.01], ImageSize -> 250];
```

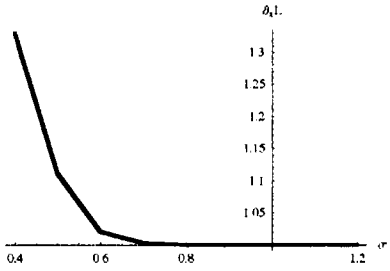


Figure 7.1 The measured derivative value of the function $y = x$ is no longer correct for decreasing scale. For scales $\sigma < 0.6$ pixels a marked deviation occurs.

The value of the derivative starts to deviate for scales smaller then say $\sigma = 0.6$. Intuitively, we understand that something must go wrong, when we decrease the size of the kernel in the spatial domain: it becomes increasingly difficult to fit the Gaussian derivative function with its zerocrossings. We recall from chapter 4 that the number of zerocrossings of a Gaussian derivative kernel is equal to the order of differentiation.
There is a fundamental relation between the order of differentiation, scale of the operator and the accuracy required [TerHaarRomeny1994b]. We will derive now this relation.
The Fourier transform of a Gaussian kernel is again a Gaussian:

```
Unprotect[gauss]; gauss[x_, σ_] := 1/Sqrt[2 π σ^2] Exp[-x^2/(2 σ^2)];
fftgauss[ω_, σ_] = FourierTransform[gauss[x, σ], x, ω]
```

$$\frac{e^{-\frac{1}{2}\sigma^2\omega^2}}{\sqrt{2\pi}}$$

The Fourier transform of the derivative of a function is $-i\omega$ times the Fourier transform of the function:

```
FourierTransform[∂x gauss[x, σ], x, ω] / FourierTransform[gauss[x, σ], x, ω]
```

$$-i\omega$$

The Fourier transform of the n-th derivative of a function is $(-i\omega)^n$ times the Fourier transform of the function. Note that there are several definitions for the signs (see the *Mathematica* Help browser for **Fourier**).
A smaller kernel in the spatial domain gives rise to a wider kernel in the Fourier domain, as shown below for a range of widths of first order derivative Gaussian kernels (in 1D):

```
DisplayTogetherArray[
  {Plot3D[fftgauss[ω, σ], {ω, -π, π}, {σ, .4, 2}, PlotPoints → 30,
    AxesLabel → {"ω", "σ", "fft"}, Axes → True, Boxed → True],
   Plot3D[gauss[x, σ], {x, -π, π}, {σ, .4, 2}, PlotPoints → 30, AxesLabel →
     {"x", "σ", "gauss"}, Axes → True, Boxed → True]}, ImageSize -> 490];
```

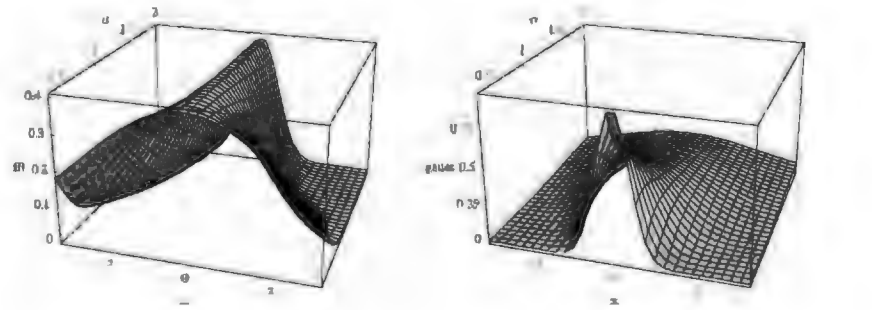


Figure 7.2 Left: The Fourier transform of the Gaussian kernel is defined for $\pi < \omega < \pi$. The function repeats forever along the frequency axis over this domain. For decreasing scale σ in the spatial domain the Fourier transform get wider in the spatial frequency domain. At some value of σ a significant leakage (aliasing) occurs. Right: The spatial Gaussian kernel as a function of scale.

We plot the Fourier spectrum of a kernel, and see that the function has signal energy outside its proper domain $[-\pi, \pi]$ for which the spectrum is defined:

```
FilledPlot[{If[-π < ω < π, fftgauss[ω, .5], 0], fftgauss[ω, .5]},
  {ω, -2 π, 2 Pi}, Fills → {{{1, Axis}, GrayLevel[.5]}},
  Ticks → {{-π, π}, Automatic},
  AxesLabel → {"ω", "G(ω,σ=.5)"}, ImageSize -> 350];
```

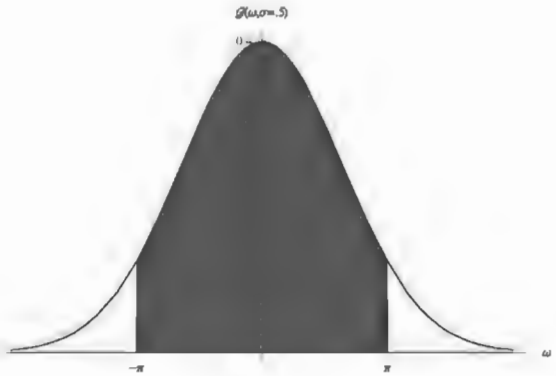


Figure 7.3 The definition of the leakage is the (unshaded) area under the curve outside the definition domain, relative to the total area under the curve. Here the definition is given for the 1D Gaussian kernel. Due to the separability of Gaussian kernels this definition is easily extended to higher dimensions.

The error is defined as the amount of the *energy* (the square) of the kernel that is 'leaking' relative to the total area under the curve (note the integration ranges):

$$\text{error}[n\_,\ \sigma\_] = 100\ \frac{\int_{\pi}^{\infty} (I\,\omega)^{2n}\ \text{fftgauss}[\omega,\ \sigma]^2\ d\omega}{\int_{0}^{\infty} (I\,\omega)^{2n}\ \text{fftgauss}[\omega,\ \sigma]^2\ d\omega}$$

$$\frac{100\ ((1+2n)\ \text{Gamma}[\frac{1}{2}+n] - 2\ \text{Gamma}[\frac{3}{2}+n] + (1+2n)^2\ \text{Gamma}[\frac{1}{2}+n,\ \pi^2\ \sigma^2])}{(1+2n)^2\ \text{Gamma}[\frac{1}{2}+n]}$$

We plot this Gamma function for scales between $\sigma = 0.2 - 2$ and order of differentiation from 0 to 10, and we insert the 5% error line in it (we have to lower the plot somewhat (-6%) to make the line visible):

```
Block[{$DisplayFunction = Identity},
  pl = Plot3D[error[n, σ] - 6, {σ, .2, 2}, {n, 0, 10}, PlotRange → All,
    AxesLabel → {"σ", "n", "error %"}, Boxed → True, Axes → True];
  p2 = ContourPlot[error[n, σ], {σ, .2, 2}, {n, 0, 10},
    ContourShading → False, Contours → {5}];
  c3d = Graphics3D[Graphics[p2][[1]] /.
     Line[pts_] :> ({Thickness[.01], val = Apply[error, First[pts]];
        Line[Map[Append[#, val] &, pts]]})][];
```

```
Show[p1, c3d, ImageSize -> 310];
```

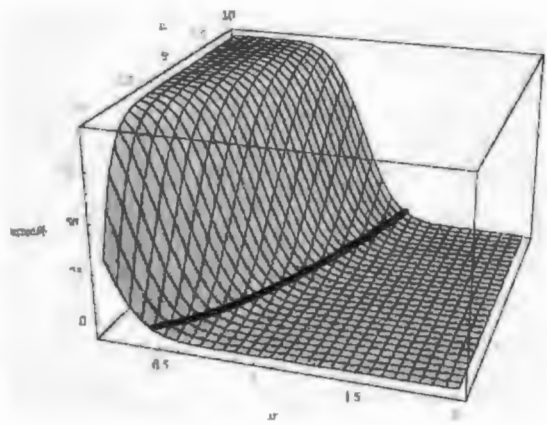


Figure 7.4 Relation between scale σ, order of differentiation n, and accepted error (in %) for a convolution with a Gaussian derivative function, implemented through the Fourier domain.

```
ContourPlot[error[n, σ], {n, 0, 10}, {σ, .1, 2},
  ContourShading → False, Contours → {1, 5, 10}, FrameLabel ->
   {"Order of differentiation", "Scale in pixels"}, ImageSize -> 275];
```

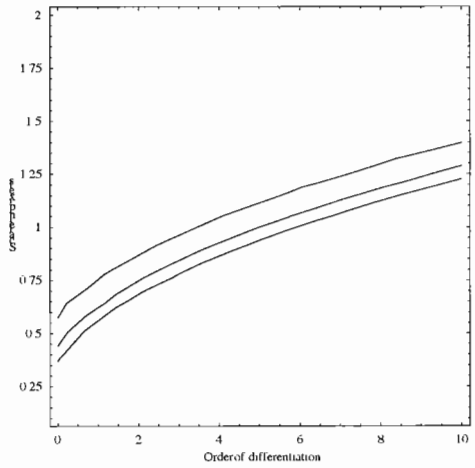


Figure 7.5 Relation between scale σ and the order of differentiation n for three fixed accuracies for a convolution with a Gaussian derivative function, implemented through the Fourier domain: upper graph: 1%, middle graph: 5%, lower graph: 10% accepted error.

The lesson from this section is that we should never make the scale of the operator, the Gaussian kernel, too small. The lower limit is indicated in the graph above. A similar reasoning can be set up for the outer scale, when the aliasing occurs in the spatial domain.

We summarize with a table of the minimum σ, given accuracies of 1, 5, resp 10%, and differentiation up to fifth order:

```
TableForm[Table[
  Prepend[{σ /. FindRoot[error[n, σ] == #, {σ, .6}]} & /@ {1, 5, 10}, n],
  {n, 1, 5}], TableHeadings ->
  {None, {"Order", "σ @ 1%", "σ @ 5%", "σ @ 10%"}}]
```

Order	σ @ 1%	σ @ 5%	σ @ 10%
1	0.758115	0.629205	0.56276
2	0.874231	0.748891	0.684046
3	0.967455	0.844186	0.78025
4	1.04767	0.925811	0.862486
5	1.11919	0.998376	0.935501

▲ Task 7.1 This chapter discusses the fundamental limit which occurs by too much 'broadening' of the Gaussian kernel in the Fourier domain for small scales (the 'inner scale limit'). Such a broadening also occurs in the spatial domain, when we make the scale too large. A similar fundamental limit can be established for the 'outer scale limit'.
Find the relation between scale, differential order and accuracy for the outer scale.

The reasoning in this chapter is based on the implementation of a convolution in the Fourier domain. The same reasoning holds however when other choices are made for the implementation. In each case, a decision about the periodicity or extension of the image values outside the domain (see the discussion in chapter 5), determines the fundamental limit discussed here.

7.2 Summary of this chapter

There is a limit to the order of differentiation for a given scale of operator and required accuracy. The limit is due to the no longer 'fitting' of the Gaussian derivative kernel in its Gaussian envelop, known as aliasing. We derived the analytic expression for this error.

As a rule of thumb, for derivatives up to 4^{th} order, the scale should be not less then one pixel.

8. Differentiation and regularization

8.1 Regularization

Regularization is the technique to make data behave well when an operator is applied to them. Such data could e.g. be functions, that are impossible or difficult to differentiate, or discrete data where a derivative seems to be not defined at all. In scale-space theory, we realize that we do physics. This implies that when we consider a system, a small variation of the input data should lead to small change in the output data.
Differentiation is a notorious function with 'bad behaviour'. Here are some examples of non-differentiable functions:

```
<< FrontEndVision`FEV`;
Block[{$DisplayFunction = Identity},
  p1 = Plot[Exp[-Abs[x]], {x, -2, 2}, PlotStyle -> Thickness[.01]];
  p2 = Plot[UnitStep[x - 1], {x, -2, 5}, PlotStyle -> Thickness[.01]];
  p3 = Plot[Floor[4 Sin[x]], {x, 0, 4 π}, PlotStyle -> Thickness[.01]];
  p4 = ListPlot[Table[Sin[4 π / √i], {i, 2, 40}], PlotStyle -> PointSize[.02]]];
Show[GraphicsArray[{{p1, p2}, {p3, p4}}], ImageSize -> 300];
```

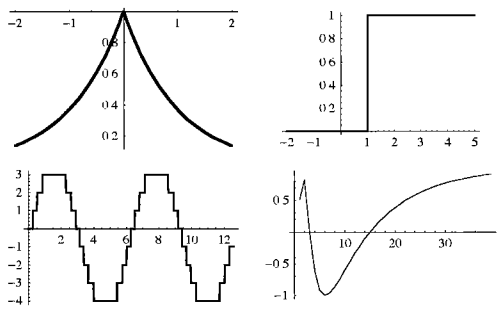


Figure 8.1 Some functions that can not be differentiated.

In mathematical terms it is said that the operation of differentiation is *ill-posed*, the opposite of *well-posed*. Jacques Hadamard (1865–1963) [Hadamard1902] stated the conditions for well-posedness:

- The solution must exist;
- The solution must be uniquely determined;
- The solution must depend continuously on the initial or boundary data.

The first two requirements state that there is one and only one solution. The third requirement assures that if the initial or boundary data change slightly, it should also have a limited impact on the solution. In other words, *the solution should be stable*.

Regularization is a hot topic. Many techniques are developed to regularize the data, each based on a constraint on how one wishes the data to behave without sacrificing too much. Well known and abundantly applied examples are:

- *smoothing* the data, convolution with some extended kernel, like a 'running average filter' (e.g. $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$) or the Gaussian;
- *interpolation*, by a polynomial (multidimensional) function;
- *energy minimization*, of a cost function under constraints [Mumford1985a, Mumford1989a, Mumford1994a];
- *fitting a function* to the data (the best known examples are splines, i.e. polynomials fitting a curve or a surface up to some order [DeBoor1978]. The *cubic splines* are named so because they fit to third order, x^3;
- *graduated convexity* [Blake1987];
- *deformable templates* ('snakes') [McInerney1996];
- *thin plates* or *thin plate splines* [Bookstein1989] (see also mathworld.wolfram.com/ThinPlateSpline.html);
- *Tikhonov regularization*, discussed in detail in the next section.

However, smoothing before the differentiation does *not* solve the ill-posedness problem. The crucial difference between the approaches above and scale-space theory is that the first methods *change the data*, your most valuable source of information, before the operation (e.g. differentiation) is applied. The derivative is taken of the regularized data.

When we recall the importance of doing a measurement uncommitted, we surely should *not* modify our data in any way. We need a regularization of the operator, not the operand. Actually, the only control we have when we do a measurement is in our measurement device. There we can change the size, location, orientation, sensitivity profiles etc. of our filtering kernels. That is something completely different from the methods described above. It is one of the cornerstones in scale-space theory that the *only control allowed* is in the filters. As such, scale-space theory can be considered the 'theory of apertures'.

8.2 Regular tempered distributions and test functions

The formal mathematical method to solve the problems of ill-posed differentiation was given by Laurent Schwartz [Schwartz1951] (see figure 8.2) as was noted by Florack [Florack1994a]. The following is adapted from [Niessen1997a]. A *regular tempered distribution* associated with an image is defined by the action of a *smooth test function* on the image. Smooth is here used in the mathematical definition, i.e. infinitely differentiable, or C^∞.

The class of smooth test functions ϕ (also called the *Schwartz space* $\mathcal{S}(\mathbb{R}^D)$) is large. It comprises all smooth functions that decrease sufficiently fast to zero at the boundaries. They are mathematically defined as the functions ϕ that are C^∞ and whose derivative to whatever order goes faster to zero to any polynomial. Mathematically stated:

$$\phi \in \mathcal{S}(\mathbb{R}^D) \iff \phi \in C^\infty(\mathbb{R}^D) \wedge \sup \| x^m \partial_{i_1 \ldots i_n} \phi(x) \| < \infty$$

for all m and n. As we consider any dimension here, m and n are multi-indices.

Let us give an example of such a function. The Gaussian kernel has the required properties, and belongs to the class of smooth test functions. E.g. its goes faster to zero then e.g. a 13th order polynomial:

```
Limit[x^13 Exp[-x^2], x → ∞]
```

```
0
```

Gaussian derivatives are also smooth test functions. Here is an example for the third order:

```
Limit[x^13 ∂_{x,x,x} Exp[-x^2], x → ∞]
```

```
0
```

The reason that the function e^{-x^2} suppresses any polynomial function, is that a series expansion leads to polynomial terms of any desired order:

```
Series[Exp[-x^2], {x, 0, 15}]
```

$$1 - x^2 + \frac{x^4}{2} - \frac{x^6}{6} + \frac{x^8}{24} - \frac{x^{10}}{120} + \frac{x^{12}}{720} - \frac{x^{14}}{5040} + O[x]^{16}$$

▲ Task 8.1 Find a number of functions that fulfill the criteria for being a member of the class of smooth test functions, i.e. a member of the Schwartz space.

A *regular tempered distribution* T_L associated with image $L(x)$ is defined as:

$$T_L = \int_{-\infty}^{\infty} L(x)\,\phi(x)\,dx$$

The testfunction 'samples' the image, and returns a scalar value. The derivative of a regular tempered distribution is defined as:

$$\partial_{i_1 \ldots i_n} T_L = (-1)^n \int_{-\infty}^{\infty} L(x)\,\partial_{i_1 \ldots i_n}\,\phi(x)\,dx$$

Thus the image is now 'sampled' with the derivative of the test function. This is the key result of Schartz' work. It is now possible to take a derivative of all the nasty functions we gave as examples above. We can now also take derivatives of our discrete images. But we still need to find the testfunction ϕ. Florack [Florack1994b] found the solution in demanding that the derivative should be a new observable, i.e. that the particular test function can be interpreted as a linear filter.
The choice for the filter is then determined by physical considerations, and we did so in chapter 2 where we derived the Gaussian kernel and all its partial derivatives as the causal non-committed kernels for an observation.

We saw before that the Gaussian kernel and its derivatives are part of the Schwartz space.

```
Show[Import["Laurent Schwartz.jpg"], ImageSize -> 150];
```

Figure 8.2 Laurent Schwartz (1915 -). Schwartz spent the year 1944-45 lecturing at the Faculty of Science at Grenoble before moving to Nancy where he became a professor at the Faculty of Science. It was during this period of his career that he produced his famous work on the theory of distributions. Harald Bohr presented a Fields Medal to Schwartz at the International Congress in Harvard on 30 August 1950 for his work on the theory of distributions. Schwartz has received a long list of prizes, medals and honours in addition to the Fields Medal. He received prizes from the Paris Academy of Sciences in 1955, 1964 and 1972. In 1972 he was elected a member of the Academy. He has been awarded honorary doctorates from many universities including Humboldt (1960), Brussels (1962), Lund (1981), Tel-Aviv (1981), Montreal (1985) and Athens (1993).

So we can now define a well-posed derivative of an image $L(x)$ in the proper 'Schwartz way':

$$\partial_{i_1 \dots i_n} L(x) = (-1)^n \int_{-\infty}^{\infty} L(y)\, \partial_{i_1 \dots i_n} \phi(y, x)\, d^d y$$

We have no preference for a particular point where we want this 'sampling' to be done, so we have linear shift invariance: $\phi(y; x) = \phi(y - x)$. We now get the result that taking the derivative of an image is equivalent with the *convolution* of the image with the *derivative of the test function*:

$$\partial_{i_1 \dots i_n} L(x) = \int_{-\infty}^{\infty} L(y)\, \partial_{i_1 \dots i_n} \phi(y - x)\, d^d y$$

The set of test functions is here the Gaussian kernel and all its partial derivatives. We also see now the relation with *receptive fields*: they are the Schwartz test functions for the visual system. They take care of making the differentiation regularized, well posed. Here is a comparison list:

Mathematics	⇔	Smooth test function
Computer vision	⇔	Kernel, filter
Biological vision	⇔	Receptive field

Of course, if we relax or modify the constraints of chapter 2, we might get other kernels (such as the Gabor kernels if we are confine our measurement to just a single spatial frequency). As long as they are part of the Schwartz space we get well posed derivatives.

The key point in the reasoning here is that there is no attempt to smooth the data and to take the derivative of the smoothed result, but that the differentiation is done *prior to* the smoothing. Differentiation is transferred to the filter. See for a full formal treatment on Schwartz theory for images the papers by Florack [Florack1992a, Florack1994a, Florack1996b, Florack1997a].

The theory of distribution is a considerable broadening of the differential and integral calculus. Heaviside and Dirac had generalized the calculus with specific applications in mind. These, and other similar methods of formal calculation, were not, however, built on an abstract and rigorous mathematical foundation. Schwartz's development of the theory of distributions put methods of this type onto a sound basis, and greatly extended their range of application, providing powerful tools for applications in numerous areas.

8.3 An example of regularization

The classical example of the regularization of differentiation by the Gaussian derivative is the signal with a high-frequency disturbance $\epsilon \cos(\omega x)$. Here ϵ is a small number, and ω a very high frequency.

We compare the mathematical derivative with convolution with the Gaussian derivative. First we calculate the mathematical derivative:

$$\partial_x \left(L[x] + \epsilon \cos[\omega x]\right)$$

$$-\epsilon\, \omega \sin[x\, \omega] + L'[x]$$

For large ω the disturbance becomes very large. The disturbance can be made arbitrarily small, provided that the derivative of the signal is computed at a sufficiently coarse scale σ in scale-space:

$$gx[x\_, \sigma\_] := \partial_x \left(\frac{1}{\sqrt{2\pi}\,\sigma} E^{-\frac{x^2}{2\sigma^2}} \right);$$

$$\text{Simplify}\left[\int_{-\infty}^{\infty} \epsilon \cos[\omega\,(x-\alpha)]\, gx[\alpha, \sigma]\, d\alpha\,,\ \{\omega > 0,\ \sigma > 0\}\right]$$

$$-e^{-\frac{1}{2}\sigma^2\omega^2}\, \epsilon\, \omega \sin[x\, \omega]$$

8.4 Relation regularization $\Longleftrightarrow$ Gaussian scale-space

When data are regularized by one of the methods above that 'smooth' the data, choices have to be made as how to fill in the 'space' in between the data that are not given by the original data. In particular, one has to make a choice for the order of the spline, the order of fitting polynomial function, the 'stiffness' of the physical model etc. This is in essence the same choice as the scale to apply in scale-space theory. In fact, it is becoming clear that there are striking analogies between scale-space regularization and other means of regularization.

An essential result in scale-space theory was shown by Mads Nielsen. He proved that the well known and much applied method of regularization as proposed by Tikhonov and Arsenin [Tikhonov and Arsenin 1977] (often called 'Tikhonov regularization') is essentially equivalent to convolution with a Gaussian kernel [Nielsen1996b, Nielsen1997a, Nielsen1997b]. Tikhonov and Arsenin tried to regularize functions with a formal mathematical approach from variational calculus called the method of Euler-Lagrange equations.

This method studies a function of functions, and tries to find the minimum of that function given a set of constraints. Their proposed formulation was the following: Make a function $E(g) = \int_{-\infty}^{\infty} (f-g)^2 \, dx$ and minimize this function for g. f and g are both functions of x. The function g must become the regularized version of f, and the problem is to find a function g such that it deviates as little as possible from f. The difference with f is taken with the so-called 2-norm, $(f-g)^2$, and we like to find that g for which this squared difference is minimal, *given a constraint.*

This constraint is the following: we also like the first derivative of g to x (g_x) to behave well, i.e. we require that when we integrate the square of g_x over its total domain we get a finite result. Mathematically you see such a requirement sometimes announced as that the functions are 'mapped into a Sobolev space', which is the space of square integrable functions.

The method of the Euler-Lagrange equations specifies the construction of an equation for the function to be minimized where the constraints are added with a set of constant factors λ_i, one for each constraint, the so-called Lagrange multipliers. In our case: $E(g) = \int_{-\infty}^{\infty} (f-g)^2 + \lambda_1 {g_x}^2 \, dx$. The *functional* $E(g)$ is called the Lagrangian and is to be minimized with respect to g, i.e. we require $\frac{dE}{dg} = 0$.

In the Fourier domain the calculations become a lot more compact. $\tilde{f}$ and $\tilde{g}$ are denoted as the Fourier transforms of f and g respectively. The famous theorem by Parceval states that the Fourier transform of the square of a function is equal to the square of the function itself. Secondly, we need the result that the Fourier transform of a derivative of a function is the Fourier transform of that function multiplied with the factor $-i\omega$. So $\mathcal{F}\left(\frac{\partial g(x)}{\partial x}\right) = -i\omega \mathcal{F}(g(x))$ where $\mathcal{F}$ denotes the Fourier transform.

For the square of such a derivative we get the factor ω^2, because the square of a complex function z is the function z multiplied with its *conjugate* ($i \rightarrow -i$), denoted as z^*, so

$z^2 = z\,z^*$ which gives the factor $(-i\,\omega)(-i\,\omega)^* = \omega^2$. So finally, because the Fourier transform is a linear operation, we get for the Lagrangian $\tilde{E}$:

$$\tilde{E}(\tilde{g}) = \mathcal{F}\{\int_{-\infty}^{\infty}(f-g)^2 + \lambda_1 g_x{}^2\, d\omega\} = \int_{-\infty}^{\infty}(\tilde{f}-\tilde{g})^2 + \lambda_1 \tilde{g}_x{}^2\, d\omega = \int_{-\infty}^{\infty}(\tilde{f}-\tilde{g})^2 + \lambda_1\,\omega^2\,\tilde{g}^2\, d\omega$$

$$\frac{d\tilde{E}(\tilde{g})}{d\tilde{g}} = 2(\tilde{f}-\tilde{g})^2 + 2\lambda_1\,\omega^2\,\tilde{g} = 0, \text{ so } (\tilde{f}-\tilde{g}) + \lambda_1\,\omega^2\,\tilde{g} = 0 \leftrightarrow \tilde{g} = \frac{1}{1+\lambda_1\,\omega^2}\,\tilde{f}.$$

The regularized function $\tilde{g}$ is (in the Fourier domain, only taking into account a constraint on the first derivative) seen to be the product of two functions, $\frac{1}{1+\lambda_1\,\omega^2}$ and $\tilde{f}$, which product is a convolution in the spatial domain.

The first result is that this first order regularization can be implemented with a spatial filtering operation. The filter $\frac{1}{1+\lambda_1\,\omega^2}$ in the spatial domain looks like this:

```
gl[x_] = InverseFourierTransform[1/(1 + λ1 ω^2), ω, x] // Simplify
```

$$e^{-\text{Abs}[x]}\sqrt{\frac{\pi}{2}}$$

```
λ1 = 1; Plot[gl[x], {x, -3, 3}, ImageSize -> 150];
```

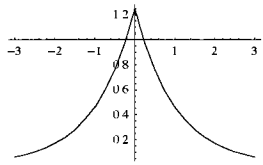


Figure 8.3 Filter function proposed by Castan et al. [Castan1990].

It is precisely the function proposed by Castan et al. [Castan1990]. The derivative of this filter is not well defined, as we can clearly see.

This is a first result for the inclusion of the constraint for the first order derivative. However, we like our function $\tilde{g}$ to be regularized with *all* derivatives behaving nicely, i.e. square integrable. When we add the constraint of the second derivative, we get two Lagrangian multipliers, λ_1 and λ_2:

$$\tilde{E}(\tilde{g}) = \int_{-\infty}^{\infty}(\tilde{f}-\tilde{g})^2 + \lambda_1\,\tilde{g}_x{}^2 + \lambda_2\,\tilde{g}_{xx}\, d\omega = \int_{-\infty}^{\infty}(\tilde{f}-\tilde{g})^2 + \lambda_1\,\omega^2\,\tilde{g}^2 + \lambda_2\,\omega^4\,\tilde{g}^2\, d\omega$$ and we find in a similar way for $\tilde{g}$:

$$\frac{d\tilde{E}(\tilde{g})}{d\tilde{g}} = 2(\tilde{f}-\tilde{g})^2 + 2\lambda_1\,\omega^2\,\tilde{g} + \lambda_2\,\omega^4\,\tilde{g} = 0 \leftrightarrow \tilde{g} = \frac{1}{1+\lambda_1\,\omega^2+\lambda_2\,\omega^4}\,\tilde{f}.$$

This is a regularization involving well behaved derivatives of the filtered $\tilde{f}$ to second order. This filter was proposed by Deriche [Deriche1987], who made a one-parameter family of this filter by setting a relation between the λ's: $\lambda_1 = 2\sqrt{\lambda_2}$. The dimensions of λ_1 and λ_2 are correctly treated by this choice. When we look at the Taylor series expansion of the

Gaussian kernel in the Fourier domain, we see that his choice is just the truncated Gaussian to second order:

```
Simplify[FourierTransform[1/Sqrt[2 π] σ E^(-x^2/(2 σ^2)), x, ω], σ > 0]
```

$$\frac{e^{-\frac{1}{2}\sigma^2\omega^2}}{\sqrt{2\pi}}$$

```
Series[E^(1/2 σ^2 ω^2), {ω, 0, 10}]
```

$$1+\frac{\sigma^2\omega^2}{2}+\frac{\sigma^4\omega^4}{8}+\frac{\sigma^6\omega^6}{48}+\frac{\sigma^8\omega^8}{384}+\frac{\sigma^{10}\omega^{10}}{3840}+O[\omega]^{11}$$

Here is how Deriche's filters look like (see figure 8.4):

```
λ1 = .; λ2 = .; g2[x_] =
 InverseFourierTransform[1/(1 + 2 Sqrt[λ2] ω^2 + λ2 ω^4), ω, x] // FullSimplify
```

$$\frac{1}{2\sqrt{\lambda 2}}\left(\sqrt{\frac{\pi}{2}}\left(\text{Cosh}\left[\frac{x}{\lambda 2^{1/4}}\right](\lambda 2^{1/4}+x\,\text{Sign}[x])-(x+\lambda 2^{1/4}\,\text{Sign}[x])\,\text{Sinh}\left[\frac{x}{\lambda 2^{1/4}}\right]\right)\right)$$

and the graph, for $\lambda_2 = 1$:

```
λ2 = 1; Plot[g2[x], {x, -4, 4}, ImageSize -> 150];
```

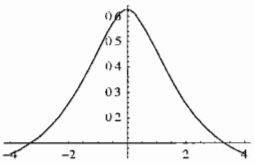


Figure 8.4 Filter function proposed by Deriche [Deriche1987].

From the series expansion of the Gaussian, and the induction from the lower order regularization, we may develop the suspicion that by adding the constraints for *all* derivatives to behave well, we get the infinite series

$$\tilde{g} = \frac{1}{1+\lambda_1\omega^2+\lambda_2\omega^4+\cdots+\lambda_n\omega^{2n}}\tilde{f} = \tilde{h}\tilde{f} \text{ for } n \to \infty.$$

Nielsen showed that the filter $\tilde{h}$ is the Gaussian kernel indeed. The reasoning goes as follows. We have an infinite number of unknowns here, the λ_n's, so we need to come up with an additional constraint that gives us just as many equations, so we can solve uniquely for this system of equations. We have just as many terms ω^{2n}, so we look for a constraint on them. It is found in the requirement that we want *scale invariance* for the filters, i.e. we want two filters $h(s)$ and $h(t)$ to *cascade*, i.e. $h(s \oplus t) = h(s) \otimes h(t)$ where $\otimes$ is the convolution operator. The parameters s and t are the scales of the filters. The operator $\oplus$ stands for the

summation at any norm, which is a compact writing for the definition $a \oplus b = (a+b)^p = a^p + b^p$. It turns out that for $p = 2$ we have the regular addition of the variances of the scales, as we have seen now several times before, due to the requirement of separability and Euclidean metric.
Implementing the cascading requirement for the first order:

$$\frac{1}{1+\lambda_1(s\oplus t)\,\omega^2} = \frac{1}{1+\lambda_1(s)\,\omega^2}\cdot\frac{1}{1+\lambda_1(t)\,\omega^2} \text{ giving}$$
$$1+\lambda_1(s\oplus t)\,\omega^2 = 1+\lambda_1(s)\,\omega^2+\lambda_1(t)\,\omega^2+\lambda_1(t)\,\lambda_1(s)\,\omega^4 \text{ and}$$
$$\lambda_1(s\oplus t) = \lambda_1(s)+\lambda_1(t)+\lambda_1(t)\,\lambda_1(s)\,\omega^2 .$$

We equal the coefficients of powers of ω both sides, so for ω^0 we find $\lambda_1(s\oplus t) = \lambda_1(s)+\lambda_1(t)$ which means that λ_1 must be a linear function of scale, $\lambda_1 = \alpha s$.
Now for the second order:

$$\frac{1}{1+\lambda_1(s\oplus t)\,\omega^2+\lambda_2(s\oplus t)\,\omega^4} = \frac{1}{1+\lambda_1(s)\,\omega^2+\lambda_2(s)\,\omega^4}\cdot\frac{1}{1+\lambda_1(t)\,\omega^2+\lambda_2(t)\,\omega^4}$$
giving
$$\lambda_1(s\oplus t)\,\omega^2+\lambda_2(s\oplus t)\,\omega^4 = \lambda_1(s)\,\omega^2+\lambda_2(s)\,\omega^4+\lambda_1(t)\,\omega^2+$$
$$\lambda_1(t)\,\lambda_1(s)\,\omega^4+\lambda_1(t)\,\lambda_2(s)\,\omega^6+\lambda_2(t)\,\omega^4+\lambda_2(t)\,\lambda_1(s)\,\omega^6+\lambda_2(t)\,\lambda_2(s)\,\omega^8$$

and equating the coefficients for ω^4 on both sides:

$\lambda_2(s\oplus t) = \lambda_1(t)\,\lambda_1(s)+\lambda_2(t)+\lambda_2(t)$ from which dimension we see that λ_2 must be quadratic in scale, and $\lambda_2 = \frac{\alpha^2 s^2}{2} = \frac{1}{2}\lambda_1{}^2$.

This reasoning can be extended to higher scale, and the result is that we get the following series:

$$g\lambda_1 = \alpha s,\ \lambda_2 = \frac{1}{2}\alpha^2 s^2,\ \lambda_3 = \frac{1}{2.4}\alpha^4\sigma^4,\ \lambda_4 = \frac{1}{2.4.6}\alpha^6\sigma^6 \text{ etc.}$$

We recall that the series expansion of the Gaussian function $E^{-\frac{1}{2}\sigma^2\omega^2}$ is, using

```
Series[E^(1/2 σ^2 ω^2), {ω, 0, 10}]
```

$$1+\frac{\sigma^2\omega^2}{2}+\frac{\sigma^4\omega^4}{8}+\frac{\sigma^6\omega^6}{48}+\frac{\sigma^8\omega^8}{384}+\frac{\sigma^{10}\omega^{10}}{3840}+O[\omega]^{11}$$

$$\frac{1}{1+\frac{\sigma^2\omega^2}{2}+\frac{\sigma^4\omega^4}{2\,4}+\frac{\sigma^6\omega^6}{2\,4\,6}+\frac{\sigma^8\omega^8}{2\,4\,6\,8}+\frac{\sigma^{10}\omega^{10}}{2\,4\,6\,8\,10}+O[\omega]^{11}}$$

```
σ = 1.; Plot[Evaluate[InverseFourierTransform[E^(-1/2 σ^2 ω^2), ω, x]],
 {x, -4, 4}, ImageSize -> 300];
```

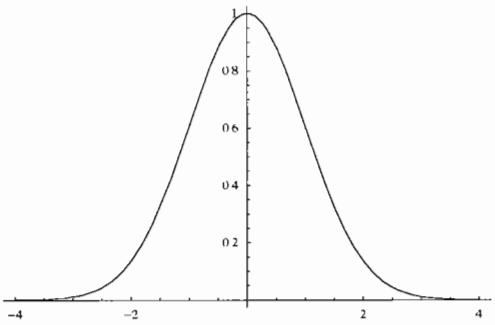


Figure 8.5 The Gaussian kernel ($\sigma = 1$).

When we take the arbitrary constant $\alpha = 1$, we get as the optimal regularization filter, where all derivatives are required to behave normal, precisely the Gaussian kernel! This important result is due to Nielsen [Nielsen1996b, Nielsen1997a, Nielsen1997b]. It has recently been proved by Radmoser, Scherzer and Weickert for a number of other regularization methods that they can be expressed as a Gaussian scale-space regularization [Radmoser1999a, Radmoser2000a].

▲ Task 8.2 Prove the equations for the coefficients λ_n in the section above by induction.

8.5 Summary of this chapter

Many functions can not be differentiated. A sampled image is such a function. The solution, due to Schwartz, is to *regularize* the data by convolving them with a smooth *test function*. Taking the derivative of this 'observed' function is then equivalent to convolving with the derivative of the test function. This is just what the receptive fields of the front-end visual system do: regularization and differentiation. It is one of the key results of scale-space theory.

A well know variational form of regularization is given by the so-called Tikhonov regularization: a functional is minimized in L^2 sense with the constraint of well behaving derivatives. It is shown in this chapter, with a reasoning due to Nielsen, that Tikhonov regularization with inclusion of the proper behaviour of *all* derivatives is essentially equivalent to Gaussian blurring.

9. The front-end visual system - the retina

We all share the same biology, regardless of ideology. (Sting)

9.1 Introduction

The visual system is our most important sense. It is estimated that about one quarter of all nerve cells we have in our central nervous system (CNS) is related to vision in some way. The task of the system is not to form an image of the outside world into the brain, but to help us to survive in this world. Therefore it is necessary to perform a substantial *analysis* of the 2D image as a projection of the 3D world. For this reason there is much more measured than just the spatial intensity distribution. We will see that the front-end visual system measures simultaneously at multiple resolutions, it measures directly (in the scale-space model) derivatives of the image in all directions at least up to fourth order, it measures temporal changes of intensity, the motion and disparity parameters, and the color differential structure. As a consequence, the layout of the receptors on the retina is strikingly different from the 2D pixel arrays in our conventional digital camera's.

There are some excellent books giving a good introduction to the basics of the human visual system. A few deserve special mentioning here:

1. "Eye, Brain and Vision" by David Hubel [Hubel1988a], a Scientific American Library book by this Nobel laureate about his pioneering work-of-life. This delightful book is from 1988, and many new findings have been added since then, but as an introduction it is very nice reading.
2. "The Visual System" by Semor Zeki [Zeki1993], with more historical descriptions of the discoveries. Zeki, as a physician, also incorporates many clinical findings in the quest.

3. "Principles of neural science" by Eric Kandel, James Schwartz and Thomas Jessell (4^{th} edition [Kandel2000]). An excellent and widely used textbook on neurons, perception and vision.
4. "The First Steps in Seeing" by Robert Rodieck [Rodieck1998] gives a detailed account of the neurophysiology of the visual front-end (particularly the retina, but also the lateral geniculate nucleus and primary visual cortex).
5. "Foundations of Vision", by Brian Wandell [Wandell1995] gives attention to both biological vision as models for processing.
6. "Neuro-Vision Systems" by Madan Gupta and George Knopf [Gupta1993a]. This is a convenient selection of reprints of the classical papers on vision and vision modeling. There are quite a few tutorial chapters added.

These books focus on the introduction into the visual system, and all cover the essential properties and details. They have excellent illustrations, and read easily for computer scientists and non-biologically educated people. They all contain pointers to the original scientific papers.

In the next few chapters we give a summary of the properties, subsystems and possible models for the front-end visual system as is relevant in the context of scale-space theory. We discussed in the previous chapters the multi-scale notion of spatial and spatiotemporal *image structure*, these chapters are about how a biological system may deal with *measurement* and *analysis* of this structure.

The most important properties in the context of this book are the following:

1. The human visual system is a *multi-scale sampling device* of the outer world. It exploits this strategy by the creation of so-called *receptive fields* (RF's) on the retina: groups of receptors assembled in such a way that they form a set of apertures of widely varying size. They together measure a *scale-space* of every image. The hierarchical structure of the input image is contained in this multi-scale stack of images measured at a range of scales. We call this the *deep structure*.

2. The human visual system does *ensemble* measurements: for every (perceivable) aspect of the stimulus it has a dedicated set of detector (receptive fields or receptive field pairs)

They span the full measurement range of the parameter, i.e. for every location, order of spatial and temporal differentiation of the stimulus, for every orientation, for every velocity in every direction, for every disparity, etc. Amazing, but there seems to be no lack of hardware ('wetware') in the visual system as we shall see.

3. The visual system is considered *layered*: its first stages measure the *geometrical* structure by multi-scale partial derivatives in space and time, and subsequent layers perform an analysis of the *contextual structure*, by perceptual grouping and hierarchical topological analysis, the highest stages do the cognitive, highly associative tasks. This rough division in processing layers is also known as front-end, intermediate or high level visual processing.

We will now look into more detail into the neuro- and electrophysiological (and psychophysical) findings that corroborate this model.

9.2 Studies of vision

The visual system is a signal processing system with spectacular performance. First of all, the mere quantity of information processed in the visual system is enormous. But most importantly, the system is capable of extracting *structural* information from the visual picture with astonishing capabilities and with real-time speed. The visual system is one of the best-studied systems in the brain.

We are however still a long way from understanding its intricate details. In this chapter we focus on the first stages of vision, where data from neuro-physiology, anatomy and computational models give rise to a reasonable insight in the processing that happens here.

After the measurement through the eye (the *looking*) our *seeing* is done in the visual cortex, a number of specialized cortical areas in the back of our brain. It has been estimated that 25% of all our estimated 10^{10} brain cells are in some way involved with the visual process.

A study of the visual system can be done from many different viewpoints:

Eye physics: the study of the physical limitations due to the optics and retinal structure.

Psychophysics: how well does the visual system perform? What are the limits of perception? Can it be tricked?

Neurophysiology and -anatomy: a study of the system's wetware organization, e.g. by measurement of the electrical activity of single or small groups of cells, and by mapping neural pathways and connections.

Functional and optical imaging: measure the functional activity of arrays or large clusters of cells (in chapter 8 these methods are described in Intermezzo 8.5).

Computational models: the field of computer vision mimicking the neural substrate to understand and predict its behavior, and to inspire artificial vision algorithms for computer vision tasks. Despite many efforts and high mathematical sophistication, today we still see a rather limited performance of computer vision algorithms in general.

Traditionally, vision is coarsely divided into three levels: front-end, intermediate and high level vision. Traditionally, the visual front-end has been defined as the measurement and first geometric analysis stage, where associative memory and recognition do not yet play a role. It is becoming clear that the visual front end receives many inputs from higher levels, making the front-end a *front-end in a context*. The outputs of the front-end go to all further stages, the intermediate and high levels.
In the front-end the first processing is done for shape, motion, disparity and color analysis (in more or less separate parallel channels). The intermediate level is concerned with perceptual grouping, more complex shape, depth and motion analysis and first associations with stored information. The high level stages are concerned with cognition, recognition and conscious perception.

This chapter focuses on front-end vision, as this is by far the best understood, and its principles may guide progress in the research of higher levels. High-level vision, where the cognitive processes take place, is a huge research area, and the most difficult one. It is the domain of many scientific disciplines, the cognitive sciences.

The visual system turns out to be extremely well organized. The retinal grid of receptors maps perfectly to the next layers in the brain in a retinotopic fashion: neighborhood relations are preserved.

Two cells, next to each other in the retina, map to cells next to each other in higher stages, i.e. the lateral geniculate nucleus (this structure in the brain on which most of the optic nerve

projects is explained in detail in chapter 7) and the primary visual cortex. We recognize a cascade of steps, many (or all?) of the stages equipped with extensive feedback to earlier stages. The mapping is not one-to-one: there is neighborhood convergence (many nearby cells to one) and neighborhood divergence (one cell to many nearby).

Many models can be proposed for the assumed working of the visual front-end in its visual processing, and extensive literature can be found. A successful recent realization is that the front-end can be regarded as a *multi-scale geometry engine* [Koenderink1984a, Koenderink1990c, Koenderink1992d]. What does this mean? Multiscan, or multi-resolution, is the property of the visual system to measure and process the information at many simultaneous levels of resolution. It is a direct consequence of the physics paradigm that the world consists of objects and structure at many sizes, and they should be measured with apertures at these sizes, i.e. at these different resolutions.

Our retina is not sending down the measurements of single rods and cones, but of *groups* of rods or cones. Such a (typically circular) group is called a *receptive field.* They come in a large range of sizes (minutes of arc to many degrees), and measure the world consequently from sharp to very blurred. For very fine details we employ the smallest receptive fields, for the larger structures we employ the larger receptive fields. Moreover: it is an extra dimension of measurement: we sample not only the spatial and temporal axes, but also along the scale axis.

The notion of geometry engine is reflected in the important contemporary model that the front-end visual system extracts *derivatives* in space and time to high order from the visual input pattern on the retina.

It turns out that in the visual front-end we have separate parallel channels for shape, motion, color and disparity processing. In our visual system we generate an extremely redundant set of measurements: for every possible value of a parameter we seem to have a specialized receptive field.

E.g. for every velocity, for every direction, for all sizes, necessary differential order and all orientations. We will study this framework in greater detail in the next sections, while following the *visual pathway*, i.e. the neuronal path of the information from the retina into the brain.

9.3 The eye

The eyes are the two moveable stereo cameras to measure the light distribution reflected and emitted from the objects in the world around us. The image is formed upside down on the retina, which is the layer of light sensitive receptors in the back of our eye. The breaking power of lens optics is defined as $1/f$ where f is the focal distance of the lens, and is expressed in *diopters* (meter^{-1}).

The breaking power of the eye optics cornea system is 60 diopters. This is due to both the cornea (43 diopter) and the lens (17 diopter), and can be varied over about 8 diopters (accommodation of the lens).

```
<< FrontEndVision`FEV`;
Show[Import["binocular projection.jpg"], ImageSize -> 300];
```

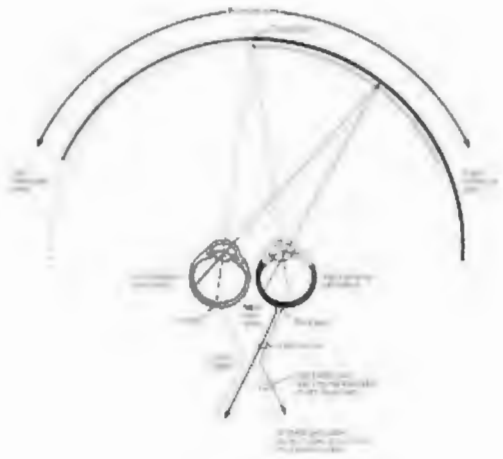


Figure 9.1 Where the visual fields of both eyes overlap we are equipped for stereo vision. The left and right visual field are each treated in a different hemisphere. The optic nerves split underway in the *chiasma* (from [Kandel 2000]).

The eye has a diameter of about 17 mm. The processing of information starts already in the retina, as this really is extended brain tissue: similar neurotransmitters are found in the retina as in the cortical brain tissues, and we recognize the same strictly layered structure in the retina as we will meet later in the visual cortex.

9.4 The retina

The retina consists coarsely of three layers of cells (figure 9.3). The light sensitive receptors, i.e. the rods and the cones, are located in the back of the eye behind the other layers. The reason for this is the close neighborhood to the nursing vessel bed in the back of the eye, the pigmented cells or chromatoid.

The rhodopsin molecules in the receptors that are bleached by the light can in this way be easily replenished. The middle layer contains horizontal, bipolar and amacrine cells (figure 9.2). The front layer contains the about one million ganglion cells, whose axons form together the optic nerve, the output of the retina.

The rods and cones are packed tightly together in a more or less hexagonal array. We have much more rods then cones. It is estimated that we have about 110,000,000 to 125,000,000 rods in our retina, and about 6,400,000 cones [Østerberg1935]. The diameter of a rod in the periphery is about 2.5 μm, or about 0.5 minute of visual angle. In the fovea, the central area of the retina, the receptors become smaller, about half this size (figure 9.2).

```
Show[Import["rods and cones mosaic.jpg"], ImageSize -> 300];
```

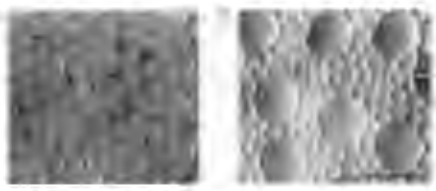


Figure 9.2 Hexagonal packing of the cones in the fovea (left) and rods/cones in the periphery (right) in the human retina. Scale bar = 10 μm. From: [Curcio et al. 1990].

Rods are only used at very dim light levels. This is rod vision, or *scotopic vision*. In normal daylight lighting conditions, they are in a completely saturated state, and have no role in perception. Rods are single color, so in the dim light we see no colors.

```
Show[Import["retina layers.jpg"], ImageSize -> 240];
```

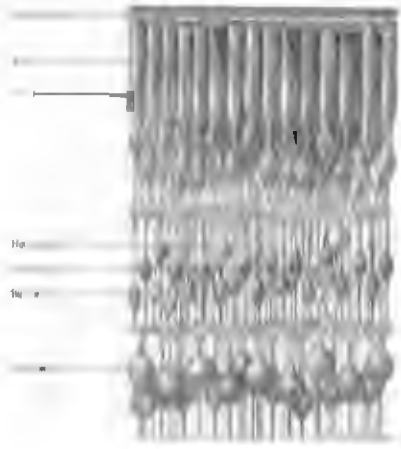


Figure 9.3 The cell layers of the retina. Light is coming from below (indeed, it must pass all the transparent cell layers!). The receptors at the top of the image touch the pigmented cells in the highly vascularized chromatoid layer, from which the bleached rhodopsin is replenished. The bipolar cells connect the receptors with the ganglion cells. The horizontal cells enable lateral interaction. The function of the many types of amacrine cells is unclear. They may have a role in motion processing (e.g. as a time-delay cell between time-coupled receptive fields). The ganglion cells at the bottom form the output and are the only cells in the retina that generate action potentials. The collective axons of all ganglion cells (about one million) form the match-thick optic nerve. From [Hubel 1988a].

The optimal sensitivity is in the green-yellow. Cones are used at normal light levels, i.e. *photopic vision*. Cones come in three types, for long (red), medium (green) and short (blue) wavelength sensitivity. Therefore these types are called L, M and S cones.

Figure 9.5 shows an electron-microscopic cross-section of a single rod. On top the hundreds of disks can be seen, which contain the light sensitive rhodopsin molecules in their membranes. Even a single photon is capable to evoke a measurable chemical reaction. We need about 3-4 photons to see them consciously. The sensitivity range of the retina is impressive: the ratio between the dimmest and the brightest light we can see (without damage) is 10^{14} !The caught photon changes the structure of the rhodopsin molecule, after which a whole chain of events generates a very small voltage change at the base of the cell, where it is transmitted to the next layer of cells, to the horizontal and bipolar cells. A rod or cone does *not* generate an action potential, just a small so-called receptor-potential of a few millivolts. Much research has been done on this process; it is beyond the scope of this chapter to go into more detail here. A good tutorial overview can be found in [Kandel et al. 2000, 4th edition].

```
Show[Import["rod disks.jpg"], ImageSize -> 400];
```

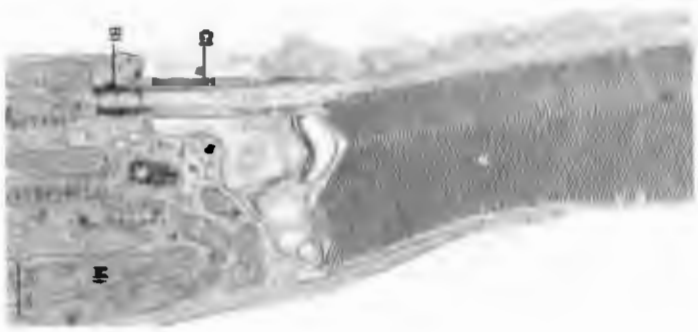


Figure 9.4 Electron-microscopic cross-section of a rod. At the right in the image are the disks with the membrane vesicles (V), which contain the visual pigment, i.e. the rhodopsin molecules. Ci = cilium, the small connecting tube between the top (outer segment) and bottom part (inner segment) of the cell. Mi = mitochondrion. Bl = basal body. Scale bar: 400 μ m. From the beautifully illustrated book [Kessel & Kardon 1979].

The receptors are not evenly distributed over the retina: in the fovea (about 1.5 mm or 5.2 degrees in diameter; one degree of visual angle is equal to 288 μm on the retina), we find an area free of rods of a roughly 250 - 750 μm.

The number of cones in the fovea is approximately 200,000 at a density of 17,500 cones/degree^2. The rod free area is about 1^o, thus there are about 17,500 cones in the central rod-free fovea. The density distribution of rods and cones as a function of eccentricity is given in figure 9.5.

```
Show[
  Import["retinal receptor distribution according to Osterberg.jpg"],
  ImageSize -> 220];
```

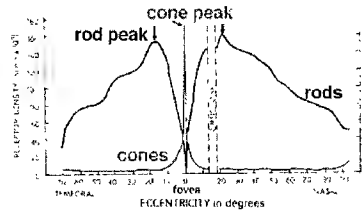


Figure 9.5 Density of retinal rods and cones as a function of eccentricity. The central area of the fovea is rod-free. From [Østerberg 1935].

9.5 Retinal receptive fields

The story of the receptive fields begins in the early fifties, when Torsten Wiesel and David Hubel (see figure 9.6) in the lab of Stephen Kuffler began to map the responses of individual ganglion cells in the retina.

```
ims = Import /@ {"David Hubel.jpg",
    "Torsten Wiesel.jpg", "Stamp Nobelpris1981.jpg"};
Show[GraphicsArray[ims], ImageSize -> 380];
```

Figure 9.6 David Hubel (left) and Torsten Wiesel (middle). They received the Nobel Prize in Medicine for their pioneering work in front-end vision neurophysiology in 1981 (right, stamp Sweden 1984).

When a very small electrode (an extruded hollow glass tube filled with electrolyte, or a tiny tip of a platinum-iridium electrode, tipsize 0.5 - 1 μm), one can record the action potentials of the ganglion cells. In the dark these cells typically 'fire' with frequencies in the 1-2 KHz range. Surprisingly, when the retina was illuminated as a whole (with a 'Ganzfeld'), the firing rates of ganglion cells did not change, no matter the level of illumination.

It was not until Hubel and Wiesel discovered that the sensitivity came in *receptive field sensitivity patterns*. Only a tiny spot of illumination could increase or decrease the firing rate of the ganglion cell (see figure 9.7).

This important discovery started a first understanding of the retinal function.

```
Show[Import["retinal RF response.jpg"], ImageSize -> 400];
```

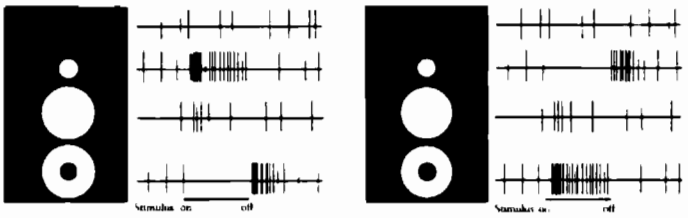


Figure 9.7 Retinal receptive field behavior on light stimulation. First row: In the dark the ganglion cell fires at the spontaneous firing rate. Third row: With homogeneous light stimulation the average firing rate is the same as in the dark. Second row: central spot stimulation. Bottom row: peripheral annular stimulation of the surround. Left: on-center/off-surround receptive field, right: off-center/on-surround receptive field. From [Hubel1988].

The area that needed to be illuminated to change the firing rate of a ganglion cell turned out to be roughly circular, contained about 30-50 receptors and had a sensitivity pattern which was excitatory (frequency-increasing) in the center and inhibitory (frequency-decreasing) in the surround in 50% of the cells. This sensitivity pattern was called *on-center center-surround*. The group of receptors and the single ganglion cell they project on, as well as the intermediate cells in between, is called a *receptive field*. The retinal receptive fields are all of the center-surround type.

For the other 50% of the ganglion cells the situation was reversed: *off-center center-surround* receptive fields. Receptive fields were found at many sizes, from a few minutes of arc to many degrees in diameter. One degree of visual angle corresponds to 200 micron distance on the retina.

It turns out that the receptive field structure is a general feature in the human senses when a spatial distribution of a signal is to be measured. The organ of Corti on the basilar membrane in the inner ear displays receptive fields, and we find them on the skin, where tactile receptive fields are formed of Pacini pressure-sensitive receptors (see figure 9.8).

It is interesting to note that the output of the central nervous system (CNS) exploits a similar strategy of 'blurring' the output of force development in a muscle. Typically, a motor neuron drives several hundreds of muscle fibres by sprouting of its axon terminals. The set of roughly circular and widely overlapping set of muscle fibres belonging and its driving motor neuron is called a 'motor unit'.

```
Show[Import["Tactile RFs.jpg"], ImageSize -> 380];
```

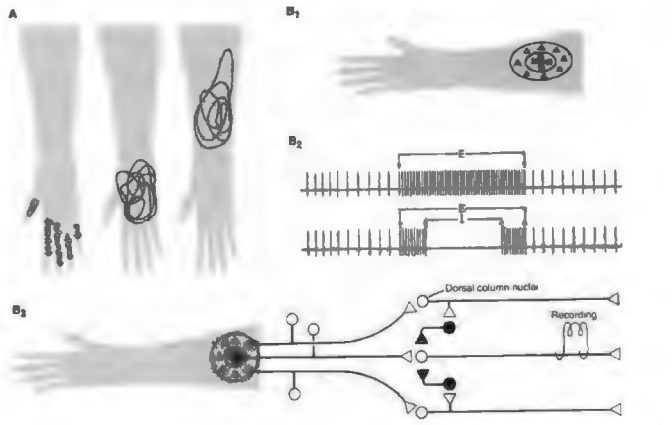


Figure 9.8 Tactile receptive fields of Pacini pressure-sensitive receptors found in the skin have a center surround structure, strongly overlap and come at a wide range of sizes. From [Kandel et al. 2000].

Hubel and Wiesel expanded these experiments, and started to measure the receptive fields of many other cells in the visual pathway, such as in the Lateral Geniculate Nucleus (LGN: a nucleus in the midbrain, discussed in detail in the next chapter), the primary visual cortex (V1), and higher.

9.6 Sensitivity profile measurement of a receptive field

DeAngelis, Ozahwa and Freeman at UC Berkeley (among others) have been able to carefully measure the receptive field sensitivity profiles of visual pathway cells in cats and monkeys (figure 9.7, see also the demonstrations on the Freeman's Lab website: http://neurovision.berkeley.edu/). They recorded the firing of the cell by inserting a very thin needle into the cell's soma (Greek: soma = body), and recorded the increase or decrease in firing rate dependent on the stimulation of its receptive field with a tiny light stimulus. They first found the general coarse location of the receptive field by manually searching the retina with a small light stimulus, and then mapped quantitatively the area belonging to it. Small light and dark flashes were randomly presented on a gray background in fast sequence (see figure 9.9).

```
Show[GraphicsArray[
   p1 = Table[Graphics[{GrayLevel[.5], Rectangle[{0, 0}, {100, 100}],
       GrayLevel[Random[Integer, {0, 1}]], Rectangle[
        rnd = {Random[Integer, {1, 92}], Random[Integer, {1, 92}]},
        rnd + {7, 7}]}, AspectRatio -> 1], {4}, {8}]], ImageSize -> 300];
```

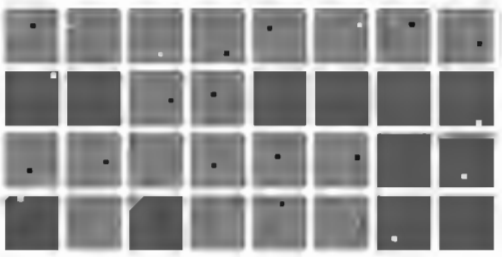


Figure 9.9 Reverse correlation stimulus. Small dot stimuli with positive or negative contrast are presented sequentially. The images shown are row by row the individual frames of the presented sequence.

For every recorded action potential it was checked what has been the location of the stimulus before, and the bin of that location was incremented. By dividing the time after the stimulus in discrete time slots, for every slot a mapping could be made, thus giving a spatio-temporal mapping, i.e. a series of maps of the RF over time (see also chapter 20). This mapping technique is known as 'reverse correlation' [de Boer1968, MacLean1994, Ringach1997] (see figure 9.10).

The advantages of the reverse correlation are plentiful. The stimulus is a white noise stimulus at low intensity, thus operating in the linear domain of the receptive field. The correlation between the stimulus and the response is calculated fast. Many stimuli can be given in a short period of time. The number of stimuli is larger then the number of responses, thus the reversing of the time axis in the analysis pays off.

An alternative way is to stimulate with white noise with a so-called maximum-length sequence. This has the advantage that numerous locations are stimulated at once, and one has a better chance of mapping also the weaker outer borders of the receptive field profile. For a more detailed discussion, see [DeValois2000], [DeAngelis1995a].

A web tutorial from Ralph Freeman's lab is available at
neurovision.berkeley.edu/Demonstrations/VSOC/teaching/AA_RFtutorial.html.
Recently, registrations from 2 neighbouring neurons could be recorded [DeAngelis1999a].

▲ Task 9.1 With the smallest foveal cone having a diameter of 2.5 μm, what is the diameter of the smallest dot you can see from a distance of 10 meters?

▲ Task 9.2 Find literature with actual measurements of the sensitivity profiles of the receptive fields in the retina, and cortical cells.

```
Show[Import["reverse correlation.gif"], ImageSize -> 250];
```

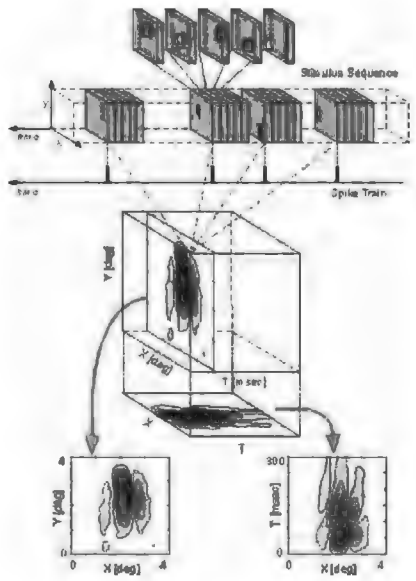


Figure 9.10 Diagram of the reverse correlation stimulus method. Top: Sequence of stimulus images as a function of time, time running to the right. During the stimulus presentation action potentials (the spike train) are recorded with a needle electrode from a single cell in LGN or cortex. For every spike recorded the causing stimulus is found (reversed in time). The time after the stimulus is divided into time bins (slots). If a spike occurs e.g. 25 ms after a certain stimulus, it is recorded at the appropriate position in the 2D post stimulus-time histogram (PSTH) at that particular time instance after the stimulus (25 ms in this case). This 25ms-PSTH is depicted vertically in the space-time cube in the middle, and in the lower left of the figure. It shows the spatial sensitivity profile of the receptive field of the cell. The set of PSTH's for the different time slots form the frames of the temporal behaviour of the cell's receptive field sensitivity profile. They can be presented as a movie, showing the dynamic change of the sensitivity profile of the receptive field. From [DeAngelis1995a].

In the retina only center-surround RF profiles are found, which are static. They are found at a wide range of sizes (from a few minutes of arc to tens of degrees). See an example profile in figure 9.11. This wide range of sizes is the retinal mechanism to sample at a range o resolutions simultaneously at the very first stage of the measurement. This is the multi-scale sampling strategy.

```
Show[Import["RF center surround cell.jpg"], ImageSize -> 150];
```

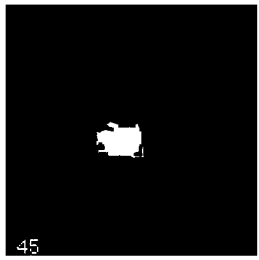


Figure 9.11 Center-surround sensitivity profile on the retina of a receptive field of a cell recorded in the lateral geniculate nucleus (LGN) in the thalamus (a midbrain structure). Cells in the LGN have the same center-surround structure as retinal ganglion cells. Tiny light measurement probe stimulation, measurement is a spatial post-stimulus-time histogram at 31 x 31 locations. This is an on-center RF: the cell increased firing when stimulated in the central area (green), and decreased firing when stimulated in the surround area (red). Field of view: 3 degrees of arc. From [DeAngelis et al. 1995].

9.7 Summary of this chapter

The eye optics project the visual world upside down on the retina, which is a receptor layer with about 150 million receptors. The retina is brain tissue, and already performs a substantial processing in its 4 main types of layered cells (receptors, horizontal, amacrine and ganglion cells). The receptors come in two types, about 120 million rods for scotopic (dim light) black and white vision, and about 30 million cones for photopic (bright light) color vision. There are three colors cones: types sensitive for light of large wavelengths (red), medium wavelengths (green) and short wavelengths (blue). The horizontal cells collect the receptor output over a roughly circular area, the bipolar cells transport the signals to the collecting large ganglion cells.

The axons of the about 1 million ganglion cells form together the match-thick optic nerve, the output of the retina. The optic nerve directly projects to the thalamus in the midbrain, ganglion cells looking to the right visual field to the left brain, and vice versa. The amacrine cells, which come in a rich variety, are assumed to play a role in motion detection.

The contribution of a set of receptors to the firing of a single ganglion cell is spatially organized in a *receptive field.* A positive sensitivity in the receptive field means that a small illumination here increases the firing frequency of the ganglion cell, and vice versa. Receptive fields in the retina are 50% on-center/surround and 50% off-center/surround: a circular sensitivity profile at a wide range of sizes, reflecting the scale-space multi-scale sampling at the retinal level. Receptive fields can be accurately recorded with single cell electrophysiological methods. A classical technique is the method of RF mapping by reverse correlation.

10. A scale-space model for the retinal sampling

10.1 The size and spatial distribution of receptive fields

Why do we have all these different sizes? Smaller receptive fields are useful for a sharp high-resolution measurement, while the larger receptive fields measure a blurred picture of the world. We denote the size of the receptive field its scale. We seem to sample the incoming image with our retina *at many scales simultaneously*.

In regular man-made camera's we don't encounter this situation: we only measure at the highest possible resolution, the basic pixel of the grid. If we want a lower resolution, e.g. for computational efficiency, we blur the sharp image with a computer afterwards. There must be an important reason to have this multi-scale capability at the retina. The different resolutions are each sampled as an independent new measurement, and they somehow need to be available simultaneously. The retina measures a whole stack of images: a *scale-space* (x-y-σ, recall figure 2.12). We will study this in detail in the chapter on the *deep structure* of images.

The measurement at the different scales gives an interesting model for the retinal receptive field distribution. The debate on why we have such an inhomogeneous receptor and receptive field distribution is old and still not set definitively (see e.g. [Williams 1991]). In this chapter we present a model from a scale-space perspective.

We already have seen that the density of receptors is not homogeneous. We may assume here the principle of *scale-invariance*, expressed in words as: 'all scales should be dealt with in identical fashion, there is no preference whatsoever for a particular scale'.

This means, that also the processing capability for each scale is equal, and that we may expect the same processing capacity for each scale, i.e. the same amount of wetware. This boils down to an equal amount of receptive fields for each scale. The densest stacking in 2D of equal circular areas is a hexagonal array. Because the eyes can move, it is most logical to put the hexagonal arrays of receptive fields all in the center of the retina, superimposed on each other so we get a lot of overlap.

The hexagonal tiling of the smallest scale receptive fields forms the fovea. The slightly larger scale receptive field array has just as many receptive fields, but naturally these cover a slightly larger area. And so on, till we encounter the largest scale, of which the receptive fields just cover the whole available retinal area.

Figure 10.1 shows the stack model for receptive fields in the retina, as originally proposed by Lindeberg and Florack [Lindeberg1992]. It is based on the earlier ground-breaking 'sunflower' model by Koenderink [Koenderink1984d, Koenderink1988d].

```
<< FrontEndVision`FEV`;
Block[{$DisplayFunction = Identity, r = 2 Cos[π / 6]}, circles =
  Flatten[(# Append[Table[{Cos[t] + r Cos[i], Sin[t] + r Sin[i], 1},
        {i, 0, 2 π - .1, π / 3}], {Cos[t], Sin[t], 1}]) & /@
    (Exp[Range[.1, 2, .3]] - .5), 1];
par = ParametricPlot3D[Evaluate[circles], {t, 0, 2 π}];
 shadowplot = Shadow[par, XShadow → False,
   YShadow → False, BoxRatios → {1, 1, 1}, Boxed → False,
   ViewPoint -> {0.646, -3.424, 1.057}, ZShadowPosition → -.5];
 oe = Exp[1.9] - .5; ob = Exp[.1] - .5;
 lines = Graphics3D[
   {Line[{{0, 0, oe}, {0, 0, 0}}], Line[{{0, 0, 0}, {oe (r + 1), 0, oe}}],
    Line[{{0, 0, 0}, {-oe (r + 1), 0, oe}}]}];
 $TextStyle = {FontFamily → "Helvetica", FontSize → 11};
 text = Graphics3D[Text["fovea", {6, 0, ob}]];
]; Show[{shadowplot, lines, text}, ImageSize -> 230];
```

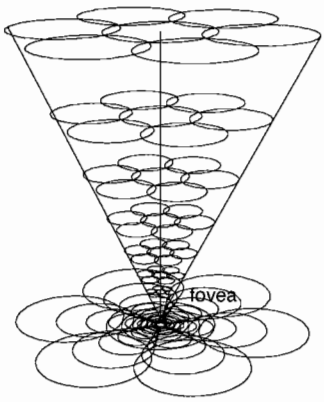


Figure 10.1 Stack model for the retinal receptive fields. Every scale is treated equally, so we assume equal hardware, i.e. numbers of receptive fields in a hexagonal sampling grid for that scale. The receptive field arrays are all centered in the middle of the retina, and are superimposed. One receptor contributes to all receptive fields that overlap with its position. The smallest array (the lowest set in the stack) forms the fovea; the largest array just covers the whole retina. From [Lindeberg and Florack 1992].

The model states that we do a *simultaneous* sampling of the image at all scales. We have scale-invariance, i.e. there is no preference for a particular scale. So what we measure, and send to the brain, is not a single image, but a *stack* of images, a *scale-space*. The very first stage of the visual front-end is a multi-scale sampling device. Figure 10.1 shows an example of such a stack of images.

This model is in good accordance with recent measurements of receptive field sizes. One measure for this size is the extent of the dendritic tree of the retinal ganglion cells. Dendrites

are the tree branch-like structures on the ganglion nerve cell body, collecting information from neighbouring cells that have synaptic connections on them. These dendrites can be measured accurately with sophisticated single cell dying techniques, after which the dendritic tree can be measured under the microscope (figure 10.3). There are different types of ganglion cells in the retina. The two most prominent families are the *midget* and the *parasol* ganglion cells. The midget cells are smaller, and project to the two layers with small cell bodies (the parvo-cellular layers; Latin: parvus = small) in the Lateral Geniculate Nucleus (LGN). The parasol cells are larger, and project to the magno-cellular layer in the LGN (Latin: magnus = large).

```
im = Import["lena64.gif"][[1, 1]];
{yres, xres} = Dimensions[im]; max = Max[im];
Block[{$DisplayFunction = Identity},
  bottom = Graphics3D[ListPlot3D[Table[0, {yres}, {xres}],
     Map[GrayLevel, im / max, {2}], Mesh → False, Boxed → False]];
stack = Table[blur = gD[im, 0, 0, i]; Graphics3D[ListPlot3D[
      Table[i 10, {yres}, {xres}], Map[GrayLevel, blur / max, {2}],
      Mesh → False, Boxed → False]], {i, 1, 6}]];
scalespace = Show[{bottom, stack}, BoxRatios → {1, 1, 1.3},
   ViewPoint -> {1.081, -3.236, 0.930}, Boxed -> True,
   DisplayFunction -> $DisplayFunction, ImageSize -> 250];
```

Figure 10.2 The retina does not measure a single image, but a series of images at different scales, here depicted as a *scale-space stack* of the famous 'Lena' image. The scale is the vertical dimension. The structural relations over scale in this stack are referred to as the 'deep structure' of an image.

As we shall see, the midget ganglion cells are involved in the measurement of *shape* and the parasol ganglion cells are involved in the measurement of *motion*.

```
Show[
  Import["midget and parasol dendritic trees.jpg"], ImageSize -> 250];
```

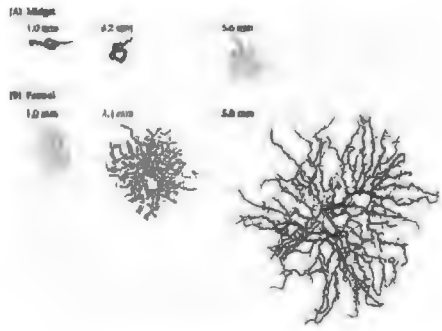


Figure 10.3 Dendritic tree size of retinal midget and parasol ganglion cells. The dendritic tree size is believed to be a representative measure for the receptive field size. Note the substantial range of sizes. Midget cells project to parvo-cellular LGN cells, parasol cells project to magno-cellular LGN cells. From [Rodieck 1998].

The model predicts that the RF diameter of both parasol and midget cell dendritic tree should increase linearly with eccentricity, which (in the macaque, a short-tailed asian monkey often used for research) is indeed found. When the measured area is plotted on the retina where they have been measured, we clearly see the increase of size with eccentricity (see figure 10.4).

```
Show[Import["RF eccentricity graph.jpg"], ImageSize -> 300];
```

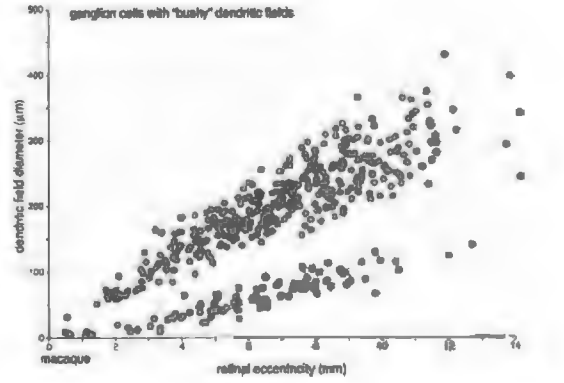


Figure 10.4 Largest diameter of the dendritic tree area in macaque retina as a function of eccentricity for midget and parasol ganglion cells. The size increases linearly with larger eccentricity, in accordance with the model. The upper cloud depicts the larger parasol ganglion cells, the lower cloud the smaller midget cells. From [Rodieck1998].

Note that there is a lack of small receptive fields at large eccentricities, as expected. We lack visual acuity at higher eccentricity due to a lack of small receptive fields. Acuity is so low,

that e.g. at 60 degrees eccentricity it is impossible to judge the number of fingers another person is sticking up at 1m distance!

Another observation is the lack of large receptive fields in the center, which the model predicts. This may be due to the fact that in the fovea it is increasingly difficult to inject the ganglion cell bodies with dye, due to the small size. The receptor density in the fovea is so high that in that area the ganglion cell bodies are displaced to slightly more outward areas, introducing another bias.

```
DisplayTogetherArray[
  Show /@ Import /@ {"midget cell rfs.jpg", "parasol cell rfs.jpg"},
  ImageSize -> 500];
```

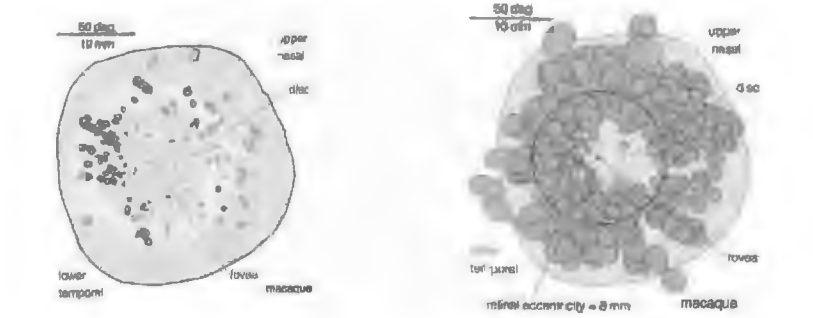


Figure 10.5 Retinal locations of different sizes of dendritic trees of midget (left) and parasol (right) retinal ganglion cells in the macaque. From [Rodieck 1998].

Very few large receptive fields populate the retina, making it statistically unlikely to hit them in the dye injection process.

Note also that the dendritic areas substantially overlap, another prediction by the model. Koenderink [Koenderink1984d] was the first to propose this retinal stack model, or 'sunflower model'.

Clearly, the distribution of dendritic tree sizes as a function of eccentricity (figure 10.4) shows a linear relation with eccentricity. From psychophysics it is well known that many important parameters change linearly with eccentricity. Among these parameters are decreasing visual acuity, decreasing velocity discrimination thresholds, and decreasing stereo acuity thresholds.

The scatter plot (figure 10.4) shows a linear upper and a lower bound to the distribution of cells. This can be understood by considering the receptive field sizes at a particular eccentricity: the smallest receptive field size is of those cells whose total area is just reaching the location at hand. Smaller receptive fields can only be found at smaller eccentricity. All the other receptive fields of this size, and all smaller ones, are at a smaller eccentricity. Only larger receptive fields are found when we go to greater eccentricity. The largest receptive field size at our specific location are bound by the fixed number of receptive fields in our tiling grid per scale that fits on the outer bounds of the retina.

```
Show[Import["on-off ganglion locations.gif"], ImageSize -> 330];
```

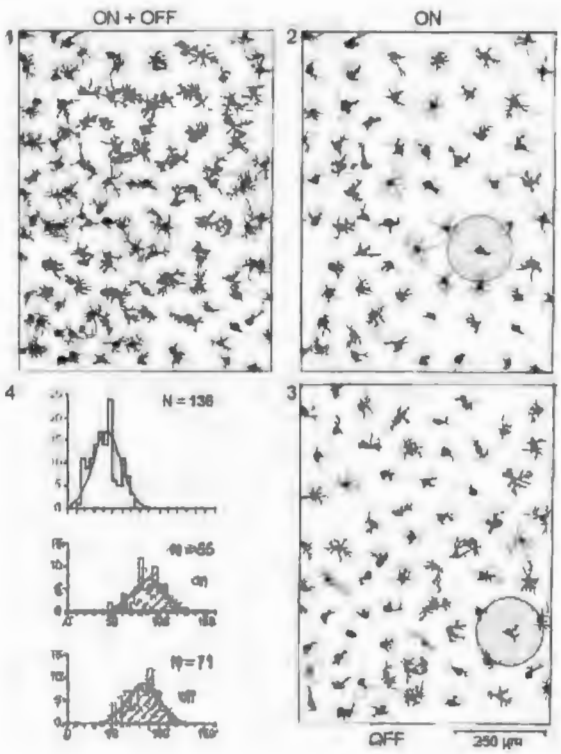


Figure 10.6 The distance between on- and off center receptive fields is significantly smaller than the distance between on-center or off-center cells proper. This implies that at a single retinal position the visual system measures both signs simultaneously. From [Dowling1987]. See also [Wässle1991].

10.2 A scale-space model for the retinal receptive fields

A good model for the sensitivity profile of a center-surround RF turns out to be Laplacian of the Gaussian kernel (figure 10.7). This is the well-known 'Mexican Hat' function, given by $\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$ where G is the Gaussian kernel given by $G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$.

Actually, it is a striking finding that we don't look with our rods and cones proper, but with receptive fields composed of (up to hundreds of) rods and cones in a center-surround structure. In scale-space terminology we 'observe the Laplacian of the world'. Figure 7.8 shows what is actually transmitted from the retina into the brain.

```
lapl=.; lapl[x_, y_, σ_] :=
 D[gauss[x, σ] gauss[y, σ], {x, 2}] + D[gauss[x, σ] gauss[y, σ], {y, 2}];
Block[{$DisplayFunction = Identity}, a = 11;
 SetOptions[Plot3D, PlotPoints -> 50, Shading -> True, Mesh -> False];
 {p1, p2} = Plot3D[Evaluate[-lapl[x, y, #]], {x, -a, a}, {y, -a, a}] & /@ {1, 3};
 {p3, p4} = Plot3D[Evaluate[lapl[x, y, #]], {x, -a, a}, {y, -a, a}] & /@ {1, 3};
 {p5, p6} =
  DensityPlot[Evaluate[-lapl[x, y, #]], {x, -a, a}, {y, -a, a}] & /@ {1, 3};
 {p7, p8} = DensityPlot[Evaluate[lapl[x, y, #]], {x, -a, a}, {y, -a, a}] & /@
   {1, 3}];
Show[GraphicsArray[{{p1, p2, p3, p4}, {p5, p6, p7, p8}}], ImageSize -> 480];
```

Figure 10.7 The Laplacian of Gaussian function as a mathematical model for the retinal center-surround receptive field sensitivity profile. Top row: small scale and large scale 'on'-center-surround (left) and 'off'-center-surround receptive field. Bottom row: The sensitivity profiles plotted as density functions. This is the model for the measurement depicted in the previous chapter in figure 9.12.

▲ Task 10.1 If we do not seem to measure the zeroth and first order spatial derivatives on the retina, hoes do we then perceive a linear gradient in intensity?

We know already for a decades that the center-surround receptive fields are causing specific 'illusions'. Two examples are given below. When we observe an image with a stepwise intensity ramp, we notice brighter regions at the side of the step closest to the darker region, and darker regions at the side of the step closest to the brighter region. When we take the Laplacian of the image, it becomes clear that the up- and downswing of the intensity can be nicely explained by taking the second order derivative (see figure 10.9).

```
im=Import["Utrecht256.gif"][[1,1]];
Block[{$DisplayFunction=Identity},
{p1,p2}=Table[ShadowPlot3D[# laplaceG[x,y,σ], {y, -15, 15}, {x, -15,
15},
PlotLabel→"σ = "<>ToString[σ],ShadowPosition->-#],{σ,2,6}]& /@ {1,-1};
{p3,p4}=Table[ListDensityPlot[# (gD[im,2,0,σ]+gD[im,0,2,σ])],{σ
,2,6}]& /@ {1,-1}];
```

```
Show[GraphicsArray[{p1,p3,p2,p4}],ImageSize->360];
```

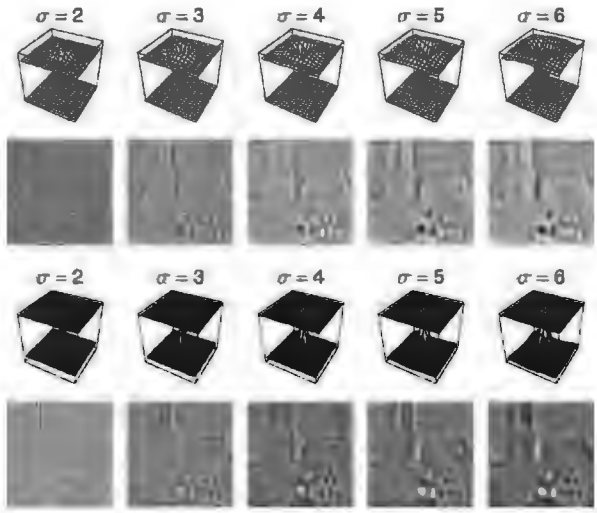


Figure 10.8 What the off-center surround (top row) and off-center surround (third row) receptive fields in the retina and LGN 'see' at a range of respective scales of the RF's. Scales: σ = 2 to 6 pixels.

```
steps = Table[Ceiling[x / 30], {y, 100}, {x, 300}];
lapl = -(gD[steps, 2, 0, 3] + gD[steps, 0, 2, 3]);
DisplayTogetherArray[
  {ListDensityPlot[#, AspectRatio → Automatic, Epilog ->
       {Hue[1], Line[{{0, 50}, {300, 50}}]}] & /@ {steps, lapl},
   ListPlot[#[[15]], AspectRatio -> .2, PlotJoined -> True,
      Axes → False] & /@ {steps, lapl}}, ImageSize → 400,
  Frame -> True, Epilog → {Text["brighter\tdarker", {0.25, 0.25}],
    Arrow[{.18, .28}, {.21, .35}], Arrow[{.32, .28}, {.28, .35}]}];
```

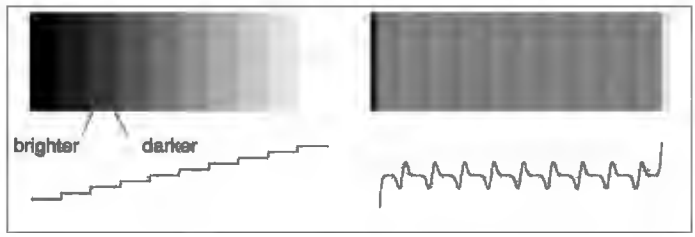


Figure 10.9 Left: linear intensity step function, resolution 300x100. Human observers see brighter intensities near an edge opposite to the darker region (a phenomenon known as the Craik-O'Brien-Cornsweet illusion, see www.sloan.salk.edu/~thomas/coce.html for the colored version of this illusion). Right: stepfunction convolved with a Laplacian center-surround receptive field.

The 2D case of this phenomenon is the illusion of the grey squares in the Hermann grid (see figure 10.10):

```
hermanngrid = Table[If[(Mod[x, 45] ≤ 32 ) && (Mod[y, 45] ≤ 32), -1, 1],
   {y, 300}, {x, 300}]; {lap11, lap12} =
  (gD[hermanngrid, 2, 0, #] + gD[hermanngrid, 0, 2, #]) & /@ {4, .5};
DisplayTogetherArray[ListDensityPlot /@ {hermanngrid, -lap11, lap12},
 ImageSize -> 350];
```

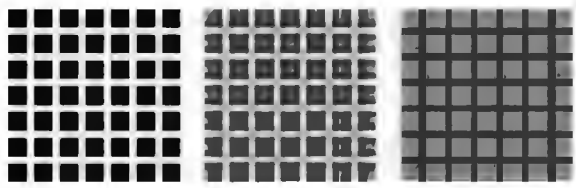


Figure 10.10 Left: the famous Hermann grid. Resolution 300^2. When we fixate on a crossing of the white lines, we notice grey spots at all other crossings. They do not appear at the crossing where we fixate. Middle: the grid convolved ('observed') with a Laplacian off-center receptive field of scale $\sigma = 4$. The grey spots are a result of the Laplacian filtering at a coarse scale. Right: idem for a Laplacian center-surround filter of small scale, $\sigma = 0.5$ pixels. At the fovea (the fixation point) we have small scales of the receptive fields, and do not observe the grey spots. For many more illusions, see Al Seckel's illusions webpage http://www.illusionworks.com.

It is unknown why we have this center-surround receptive field structure at the retinal level. The traditional textbook explanation stresses the notion of *lateral inhibition*, the enhancement of structure relative to its neighboring structures.

Another often used model is the representation of the receptive field sensitivity function of retinal ganglion cells as a *difference of Gaussians* function. It is unclear however (other than heuristic) how the two widths of the Gaussians should be taken.

```
Plot[gauss[x, 2] - gauss[x, 25], {x, -40, 40}, ImageSize -> 150];
```

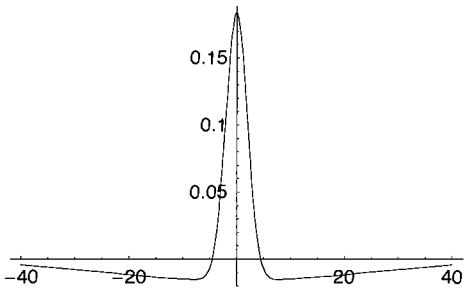


Figure 10.11 Model of the on-center receptive field function as a difference of Gaussian function.

The diffusion equation $\frac{\partial L}{\partial t} = \Delta L$ may be interpreted (in a sloppy sense) as $\delta L = \delta t \, \Delta L$, so integration of both sides of the equation over all scales gives a *robust* measurement in the sense that the exclusion of some scales due to damage to some ganglion cells or axons in the optic nerve may be less noticeable for the subsequent layers.

Another interpretation of the diffusion equation may be the conjecture that on the retina we actually sample $\frac{\partial L}{\partial t}$, the change in luminance when we change locally our aperture somewhat.

It is an intriguing observation that the multi-scale sampling of the outside world by the visual system takes place at the retinal level. All scales are separately and probably independently sampled from the incoming intensity distribution. In multi-scale *computer* vision applications the different scale representations are *generated* afterwards. The fundamental reason to sample at this very first retinal level is to observe the world at all scales *simultaneously*.

In chapters 13-15 we will discuss the resulting scale-space proper, i.e. the *deep structure* of the scale-space.

```
im = Import["T1.pgm"][[1, 1]]; lapl = (gD[im, 2, 0, #] + gD[im, 0, 2, #]) &;
stack = lapl /@ Table[E^τ, {τ, 0, 3, .15}];
DisplayTogetherArray[{ListDensityPlot /@ Take[stack, 7],
   ListDensityPlot /@ Take[stack, {8, 14}],
   ListDensityPlot /@ Take[stack, -7]}, ImageSize -> 490];
```

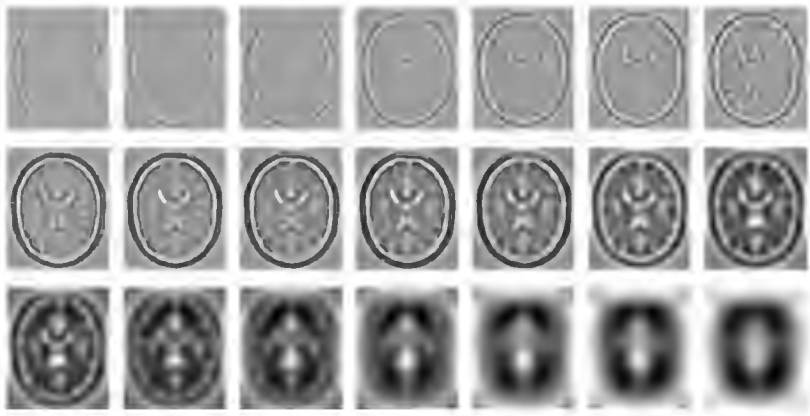


Figure 10.12 Stack of on-center receptive field function outputs as an exponential function of scales.

10.3 Summary of this chapter

The retina is a *multi-scale sampling device.* A scale-space inspired model for the retinal sampling at this very first level considers the retina as a stack of superimposed retinas each at a different scale. As a consequence of scale invariance, each scale is likely to be treated equally, and be equipped with the same processing capacity in the front-end. This leads to the model that each retina at a particular scale consists of the same number of receptive fields that tile the space, which may explain the linear decrease of acuity with eccentricity.

The reasons why we do a center-surround sampling on the retina are not yet clear. The sampling of the Laplacian at a range of scales may be necessary for efficient analysis in a proper multi-scale 'deep structure' setting, much of which strategy still needs to be discovered.

```
DisplayTogetherArray[
  ListDensityPlot /@ {im, Plus @@ stack, lapl[1.5]}, ImageSize -> 400];
```

Figure 10.13 Left: original image, resolution 217x181. Middle: Sum of the Laplacians of the original image at 16 scales (exponentially sampled between 1 and 20 pixels, see fig. 10.12). Right: The Laplacian of the original image at a scale of 1.5 pixels.

11. The front-end visual system - LGN and cortex

What we see depends mostly on what we look for. (Kevin Eikenberry, 1998)

11.1 The thalamus

From the retina, the optic nerve runs into the central brain area and makes a first monosynaptic connection in the Lateral Geniculate Nucleus, a specialized area of the thalamus (see figure 11.1 and 11.2).

```
<< FrontEndVision`FEV`;
Show[Import["optic pathway bottom view.jpg"], ImageSize -> 360];
```

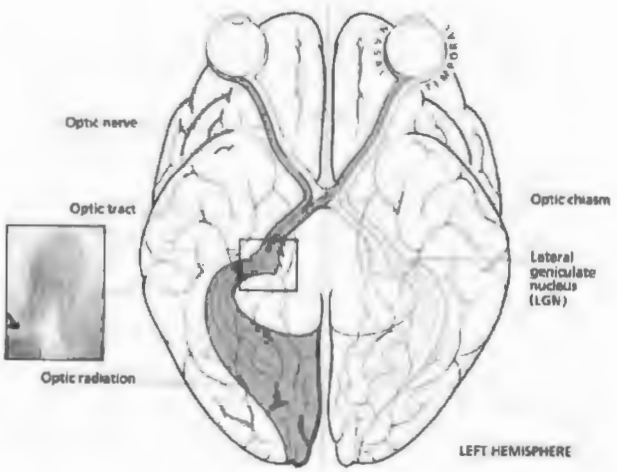


Figure 11.1 The visual pathway. The left visual field is processed in the right half of the brain, and vice versa. The optic nerve splits half its fibers in the optic chiasm on its way to the LGN, from where an extensive bundle projects to the primary visual cortex V1. From [Zeki 1993].

The thalamus is an essential structure of the midbrain. Here, among others, all incoming perceptual information comes together, not only visual, but also tactile, auditory and balance. It is one of the very few brain structures that cannot be removed surgically without lethal consequences.

The thalamus structure shows a precise somatotopic (Greek: soma = $\sigma\omega\mu\alpha$ = body; topos = $\tau\acute{o}\pi o\varsigma$ = location) mapping: it is divided in volume parts, each part representing a specific part of the body (see [Sherman and Kock 1990]).

```
Show[GraphicsArray[{Import["thalamus coronal cut.jpg"],
    Import["thalamus location.jpg"]}],
  GraphicsSpacing → -.15, ImageSize -> 500];
```

Figure 11.2 Location of the left and right thalamus with the LGN in the brain. Left: coronal slice through the middle of the brain. The thalamus is a very early structure in terms of brain development. Right: spatial relation of the thalamus and the cerebellum. From [Kandel et al. 2000].

```
Show[Import["thalamus with LGN.jpg"], ImageSize -> 450];
```

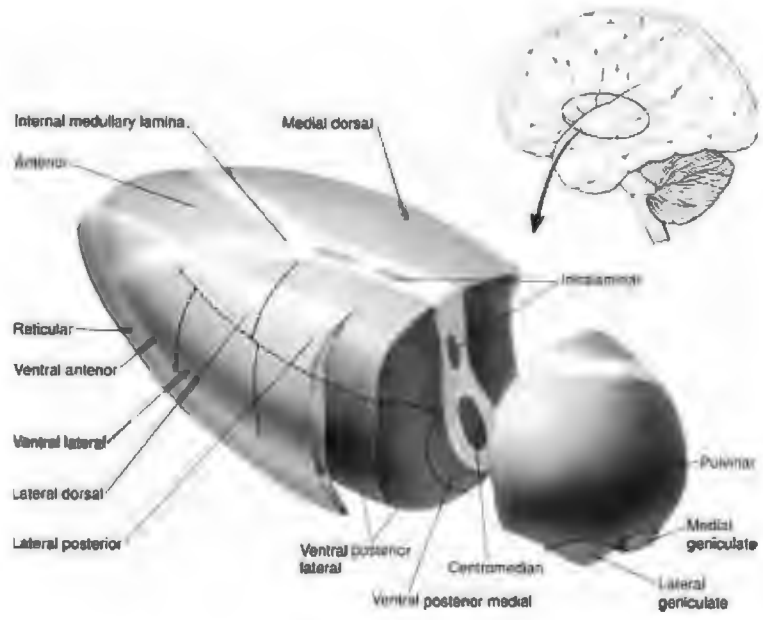


Figure 11.3 Subdivisions of the thalamus, each serving a different part of the body. Note the relatively small size of the LGN at the lower right bottom. From [Kandel et al. 2000]. See also biology.about.com/science/biology/library/organs/brain/blthalamusimages.htm.

11.2 The lateral geniculate nucleus (LGN)

On the lower dorsal part of each thalamus (figure 11.3) we find a small nucleus, the Lateral Geniculate Nucleus (LGN). The left and right LGN have the size of a small peanut, and consist of 6 layers (figure 11.4).
The top four layers have small cell bodies: the parvo-cellular layers (Latin: parvus = small). The bottom two layers have larger cell bodies: the magno-cellular layers (Latin: magnus = big). Each layer is monocular, i.e. it receives ganglion axons from a single eye by monosynaptic contact.

```
Show[Import["LGN layers.jpg"], ImageSize -> 210];
```

Figure 11.4 The 6 layers of the left Lateral Geniculate Nucleus (LGN). The top four parvo-cellular layers have relatively small cell bodies; the bottom two magno-cellular layers have relatively large cell bodies. The central line connects the cells receiving projections from the fovea. Cells more to the right receive projections from the nasal visual field, to the left from the temporal visual field. Note that the first and third layers are somewhat shorter on the right side: this is due to the blocking of the nasal visual field by the nose. Size of the image: 3 x 4 mm. From [Hubel 1988a].

The order interchanges from top to bottom for the parvo-cellular layers: left, right, left, right, and then a change for the magno-cellular layers: right, left. The magno-cellular layers are involved in mediating motion information; the parvo-cellular layers convey shape and color information. It is so far not known *what* is separated in the 2 pairs of parvo-cellular layers or in the pair of magno-cellular layers. Some animals have a total of 4 or eight layers in the LGN, and some patients have been found with as many as 8 layers.

The mapping from retina to the LGN is very precise. Each layer is a retinotopic (Greek: topos = location) map of the retina. The axons of the cells of the LGN project the visual signal further to the ipsi-lateral (on the same side) primary visual cortex through a wide bundle of projections, the optic radiations.

What does an LGN cell see? In other words, what do receptive fields of the LGN cells see?

Electrophysiological recordings by DeAngelis, Ohzawa and Freeman [DeAngelis1995a] learned that the receptive field sensitivity profile of LGN cells are the same as those of retinal ganglion cells: circular center-surround receptive fields, with on-center and off-center in equal numbers, and at the same range of scales.

However, the *receptive fields are not constant in time*, or stationary. With the earlier mentioned technique of reverse correlation DeAngelis et al. were able to measure the receptive field sensitivity profile at different times after the stimulus. The polarity turns out to change over time: e.g. first the cell behaves as an on-center cell, several tens of milliseconds later it changes into an off-center cell.

A good model describing this spatio-temporal behavior is the product of a Laplacian of Gaussian for the spatial component, multiplied with a first order derivative of a Gaussian for the temporal domain. In formula: $\left(\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}\right)\frac{\partial G}{\partial t}$. So such a cell is an operator: it takes the first order temporal derivative.

```
Show[Import["Spatiotemporal LGN.jpg"], ImageSize -> 180];
```

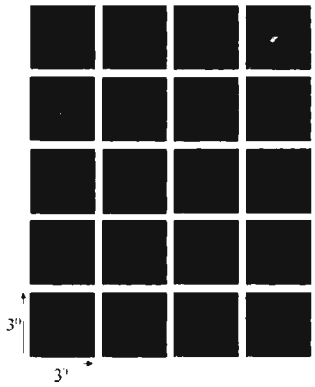


Figure 11.5 Spatio-temporal behavior of a receptive field sensitivity profile of an LGN cell. Delays after the stimulus shown 15-280 ms, in increments of 15 ms. Each figure is the sensitivity response profile at a later time (indicated in the lower left corner in ms) after the stimulus. Read row-wise, first image upper left. Green is positive sensitivity (cell increases firing when stimulated here), red is otherwise. From [DeAngelis et al. 1995a]. See also their informative web-page: http://totoro.berkeley.edu.

As in the retina, 50% of the center-surround cells is on-center, 50% is off-center. This may indicate that the foreground and the background are just as important (see figure 11.6).

```
DisplayTogetherArray[
  Show /@ Import /@ {"utensilsl.jpg", "utensils2.jpg"}, ImageSize -> 400];
```

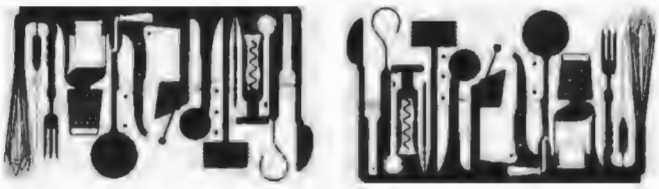


Figure 11.6 Foreground or background? The same image is rotated 180 degrees. From [Seckel2000].

11.3 Corticofugal connections to the LGN

It is well known that the main projection area after the LGN for the primary visual pathway is the primary visual cortex in Brodmann's area 17 (see figure 11.8). The fibers connecting the LGN with the cortex form a wide and conspicuous bundle, the *optic radiation* (see figure 11.10). The topological structure is kept in this projection, so again a map of the visual field is projected to the visual cortex. A striking recent finding is that 75% of the number of fibers in this bundle are corticofugal ('from the cortex away') and project from the cortex to the LGN! The arrow in figure 11.7 shows this.

```
Show[Import["visual pathway projections with arrow.gif"],
  ImageSize -> 260];
```

Figure 11.7 75% of the fibers in the optic radiation project in a retrograde (backwards) fashion, i.e. from cortex to LGN. This reciprocal feedback is often omitted in classical textbooks. Adapted from Churchland and Sejnowski [Churchland1992a].

This is an ideal mechanism for feedback control to the early stage of the thalamus. We discuss two possible mechanisms:

1. Geometry-driven diffusion;
2. Long-range interactions for perceptual grouping.

A striking finding is that most of the input of LGN cells comes from the primary cortex. This is strong feedback from the primary cortex to the LGN.

It turns out that by far the majority of the input to LGN cells (nearly 50%) is from higher cortical levels such as V1, and only about 15-20% is from retinal input (reviewed in [Guillery1969a, Guillery1969b, Guillery1971])!

It is not known what exact purpose these feedback loops have and how these retrograde (i.e. backwards running) corticofugal (i.e. fleeing from the cortex) projections are mapped. It is generally accepted that the LGN has a gating / relay function [Sherman1993a, Sherman1996a].

One possible model is the possibility to adapt the receptive field profile in the LGN with local geometric information from the cortex, leading e.g. to edge-preserving smoothing: when we want to apply small scale receptive fields at edges, to see them at high resolution, and to apply large scale receptive fields at homogeneous areas to exploit the noise reduction at coarser scales, the model states that the edginess measure extracted with the simple cells in the cortex may tune the receptive field size in the LGN. At edges we may reduce the LGN observation scale strongly in this way. See also [Mumford1991a, Mumford1992a, Wilson and Keil 1999]).

In physical terminology we may say that we have introduced local changes in the conductivity in the (intensity) diffusion. Compare our scale-space intensity diffusion framework with locally modulated heat diffusion: at edges we have placed heat isolators, so at those points we have reduced or blocked the diffusion process. In mathematical terms we may say that the diffusion is locally modulated by the first order derivative information.

Of course we may modulate with any order differential geometric information that we need in modeling this *geometry-driven, adaptive filtering* process. We also may modulate the size of the LGN receptive field, or its shape. Making a receptive field much more elongated along an edge than across an edge, we can smooth along the edge more then we smooth across the edge, thus effectively reducing the local noise without compromising the edge strength. In a similar fashion we make the receptive field e.g. banana-shaped by modulating its curvature so it follows even better the edge locally, etc.

This has opened a large field in mathematics, in the sense that we can make up new, nonlinear diffusion equations.

This direction in computer vision research is known as PDE (partial differential equation) -based computer vision. Many nonlinear diffusion schemes have been proposed so far, as well as many elegant mathematical solutions to solve these PDE's. We will study an elementary set of these PDE's in chapter 21 on nonlinear, geometry-driven diffusion.

An intriguing possibility is the exploitation of the filterbank of oriented filters we encounter in the visual cortex (see next chapter). The possibility of combining the output of differently oriented filters into a nonlinear perceptual grouping task is discussed in chapter 19.

```
Show[Import["A1-02-brodmann.gif"], ImageSize -> 400];
```

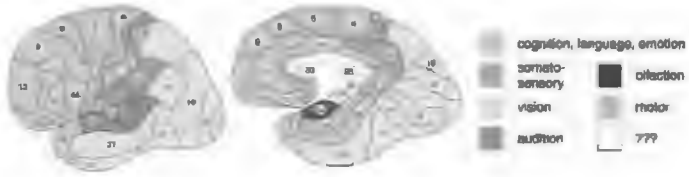


Figure 11.8 In 1909 Brodmann published a mapping of the brain in which he identified functional areas with numbers. The primary visual cortex is Brodmann's area 17. From these maps it is appreciated that the largest part of the back part of the brain is involved in vision. From [Garey1987]).

11.4 The primary visual cortex

The primary visual cortex is the next main station for the visual signal. It is a folded region in the back of our head, in the calcarine sulcus in area 17. In the visual cortex we find the first visual areas, denoted by V1, V2, V3, V4 etc.

The visual cortex (like any other part of the cortex) is extremely well organized: it consists of 7 layers, in a retinotopic, highly regular columnar structure. The layers are numbered 1 (superficial) to 7 (deep). The LGN output arrives in the middle layers 4a and 4b, while the output leaves primarily from the top and bottom layers.
The mapping from the retina to the cortical surface is a *log-polar* mapping (see for a geometrical first principles derivation of this transformation [Schwartz1994, Florack2000d]).

```
Show[Import["V1 cortical cross section.jpg"], ImageSize -> 320];
```

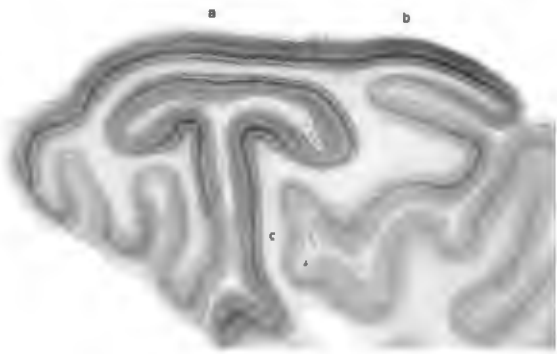


Figure 11.9 Horizontal slice through the visual cortex of a macaque monkey. Slice stained for cell bodies (gray matter). Note the layered structure and the quite distinct boundaries between the visual areas (right of b and left of c). (a) V1, center of the visual field. (b) V1, more peripheral viewing direction. (c) Axons between the cortical surfaces, making up the gross connection bundles, i.e. the white matter. From [Hubel1988a].

If the retinal polar coordinates are (ρ, θ), with are related to the Cartesian coordinates by $x = \rho\cos(\theta)$ and $y = \rho\sin(\theta)$, the polar coordinates on the cortex are described by $(\rho c, \theta c)$, with $\rho c = \ln\left(\frac{\rho}{\rho_0}\right)$ and $\theta c = q\,\theta$. Here ρ_0 is the size of the fovea, and $1/q$ is the minimal angular resolution of the log-polar layout. The fovea maps to an area on the cortex which is about the same size as the mapping from the peripheral fields.

▲ Task 11.1 Generate from an arbitrary input image the image as it is mapped from the retina onto the cortical surface according to the log-polar mapping. Hint: Use the *Mathematica* function `ListInterpolation[im]` and resample the input image according to the new transformed coordinates, i.e. fine in the center, coarse at the periphery.

```
Show[Import["Calcarine fissure.jpg"], ImageSize -> 350];
```

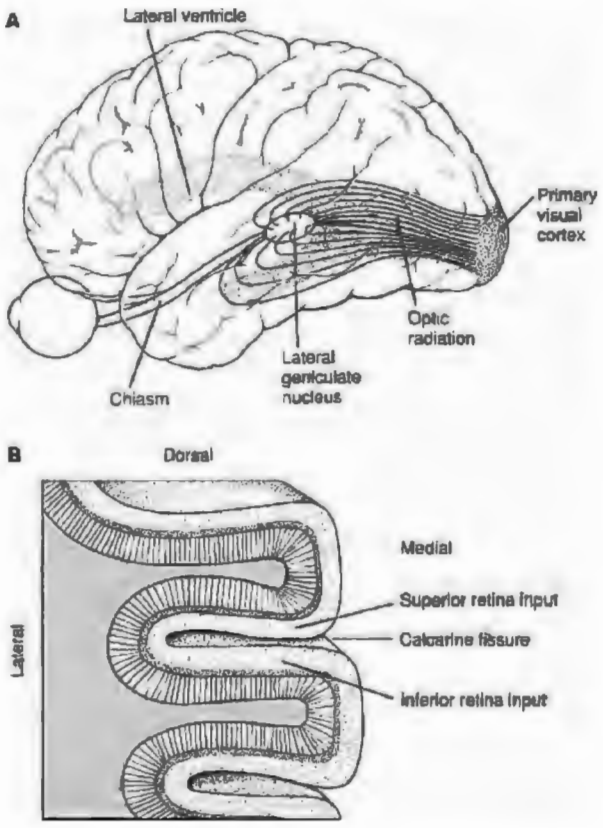


Figure 11.10 LGN pathway. From the LGN fibers project to the primary visual cortex in the calcarine sulcus in the back of the head. From [Kandel et al. 2000].

The cortical columns form a repetitive structure of little areas, about 1 x 1 mm, which can be considered the visual 'pixels'. Each column contains all processing filters for local geometrical analysis of that pixel. Hubel and Wiesel [Hubel1962] were the first to record the RF profiles of V1 cells. They found a wide variety of responses, and classified them broadly as *simple cells*, *complex cells* and *hypercomplex (end-stopped)* cells.

11.4.1 Simple cells

The receptive field sensitivity profiles of simple cells have a remarkable resemblance to Gaussian derivative kernels, as was first noted by Koenderink [Koenderink1984a]. He proposed the Gaussian derivative *family* as a taxonomy (structured name giving) for the simple cells.

```
point = Table[0, {128}, {128}]; point[[64, 64]] = 1000;
Block[{$DisplayFunction = Identity},
  pl = Table[ListContourPlot[gD[point, n, m, 15], ContourShading → True],
    {n, 1, 2}, {m, 0, 1}]];
Show[GraphicsArray[pl], GraphicsSpacing → 0, ImageSize -> 160];
```

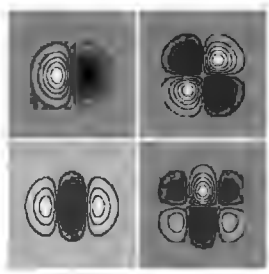


Figure 11.11 Gaussian derivative model for receptive profiles of cortical simple cells. Upper left: $\frac{\partial G}{\partial x}$, upper right: $\frac{\partial^2 G}{\partial x \partial y}$, lower left: $\frac{\partial^2 G}{\partial x^2}$, lower right: $\frac{\partial^3 G}{\partial^2 x \partial y}$.

Daugman proposed the use of Gabor filters in the modeling of the receptive fields of simple cells in the visual cortex of some mammals. In the early 1980's a number of researchers suggested Gaussian modulated sinusoids (Gabor filters) as models of the receptive fields of simple cells in visual cortex [Marcelja1980, Daugman1980] Watson1982a, Watson1987a, Pribram1991]. A good discussion on the use of certain models for fitting the measured receptive field profiles is given by [Wallis1994].

Recall figure 2.11 for some derivatives of the Gaussian kernel. DeAngelis, Ohzawa and Freeman have measured such profiles in single cortical cell recordings in cat and monkey [DeAngelis1993a, DeAngelis1995a, DeAngelis1999a].

Figure 11.12 shows a measured receptive field profile of a cell that can be modeled by a second order (first order with respect to x and first order with respect to t) derivative. Receptive fields of simple cells can now be measured with high accuracy (see also Jones and Palmer [Jones1987]).

```
Show[
  Import["RF simple cell XT separable series.jpg"], ImageSize -> 400];
```

Figure 11.12 Receptive field profile of a V1 simple cell in monkey as a function of time after the stimulus. Times in 25 ms increments as indicated in the lower left corner, 0-275 ms. Field of view 2 degrees. From [DeAngelis et al. 1995a].

```
Show[{Import["Simple cell V1 XT.gif"], Graphics[
    {Text["x→", {5., -5.}], Text["t↑", {-8., 8}]}]}, ImageSize -> 120];
```

Figure 11.13 Sensitivity profile of the central row of the sensitivity profile in the images of figure 11.12 as a function of time (vertical axis, from bottom to top). This cell can be modeled as the first (Gaussian) derivative with respect to space and the first (Gaussian) derivative with respect to time, at an almost horizontal spatial orientation. From [DeAngelis et al. 1995a].

As with the LGN receptive fields, all the cortical simple cells exhibited a dynamic behaviour. The receptive field sensitivity profile is not constant over time, but the profile is *modulated.*

In the case of the cell depicted in figure 11.12 this dynamic behaviour can be modeled by a first order Gaussian derivative with respect to time. Figure 11.13 shows the response as a function of both space and time.

11.4.2 Complex cells

The receptive field of a complex cell is not as clear as that of a simple cell. They show a marked temporal response, just as the simple cells, but they lack a clear spatial structure in the receptive field map. They appear to be a next level of abstraction in terms of image feature complexity.

```
Block[{$DisplayFunction = Identity}, pl =
  Table[Show[Import["complex " <> ToString[i] <> ".gif"]], {i, 1, 30}]];
Show[GraphicsArray[Partition[pl, 6]], ImageSize -> 260];
```

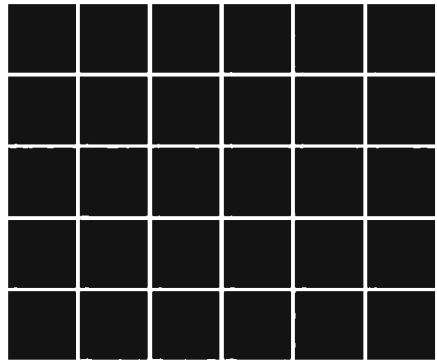


Figure 11.14 Complex cell receptive field sensitivity profile as a function of time. Only bright stimulus responses are shown. Responses to dark stimuli are nearly identical in spatial and temporal profiles. Bright bar response. XY domain size: 8 x 8 degs. Time domain: 0 - 150 msec in 5 msec steps. Orientation: 90 degrees. Bar size: 1.5 x 0.4 degrees. Cell bk326r21.02r. Data from Ohzawa 1995, available at neurovision.berkeley.edu/Demonstrations/VSOC/teaching/RF/Complex.html.

One speculative option is that they may be modeled as processing some (polynomial?) function of the neighboring derivative cells, and thus be involved in complex differential features (see also [Alonso1998a]).

As Ohzawa states: "Complex cell receptive fields are not that interesting when measured with just one stimulus, but they reveal very interesting internal structure when studied with two or more stimuli simultaneously" (http://neurovision.berkeley.edu/).

11.4.3 Directional selectivity

Many cells exhibit some form of strong directional sensitivity for motion. Small bars of stimulus light are moved across the receptive field area of a rabbit directionally selective cortical simple cell from different directions (see figure 11.15).

When the bar is moved in the direction of optimal directional response, a vigorous spiketrain discharge occurs. When the bar is moved in a more and more deviating direction, the response diminishes. When the bar is moved in a direction perpendicular to the optimal response direction, no response is measured. The response curve as a function of orientation is called the *orientation tuning curve* of the cell.

```
Show[Import["directional cell response.jpg"], ImageSize → 210];
```

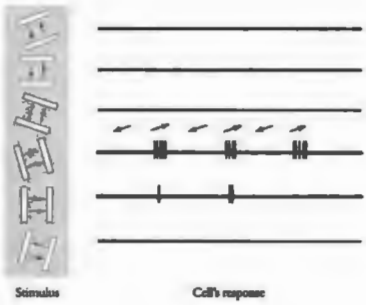


Figure 11.15 Directional response of a cortical simple cell. This behaviour may be explained by a receptive field that changes polarity over time, as in figure 11.12, or by a Reichart-type motion sensitive cell (see chapters 11 and 17). From [Hubel 1988].

The cortex is extremely well organized. Hubel and Wiesel pioneered the field of the discovery of the organizational structure of the visual system.

```
Show[Import["hypercolumn model.jpg"], ImageSize -> 340];
```

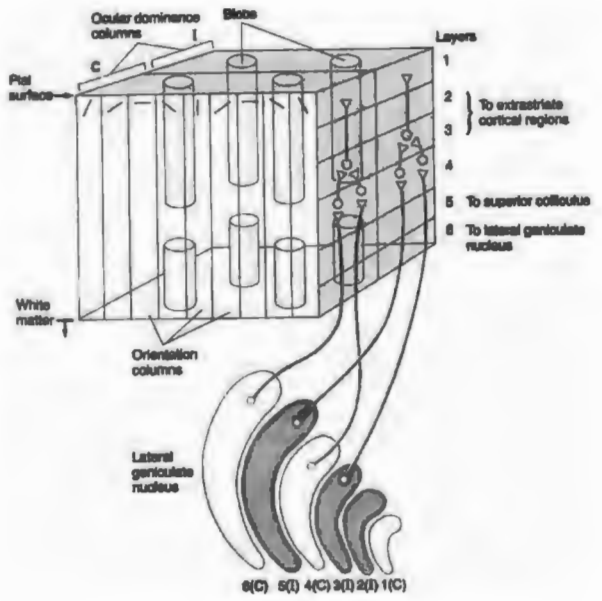


Figure 11.16 Left: projection of the LGN fibers to the cortical hypercolumns in the primary visual cortex V1. A hypercolumn seems to contain the full functionality hardware for a small visual space angle in the visual field, i.e. the RF's of the cells in a hypercolumn are represented for both eyes, at any orientation, at any scale, at any velocity, at any direction, and any disparity. In between the simple and complex cells there are small 'blobs', which contain cells for color processing. From [Kandel et al. 2000].

One of their first discoveries was that the left and right eye mapping originates in a kind of competitive fashion when, in the embryonal phase, the fibers from the eye and LGN start projecting to the cortical area. They injected a monkeys eye with a radioactive tracer, and waited a sufficiently long time that the tracer was markedly present, by backwards diffusion through the visual axons, in the cortical cells. They then sliced the cortex and mapped the presence of the radioactive tracer with an autoradiogram (exposure of the slice to a photographic film). Putting together the slices to a cortical map, they found the pattern of figure 11.17.

```
Show[Import["ocular dominance columns.gif"], ImageSize -> 200];
```

Figure 11.17 Ocular dominance bands of monkey striate cortex, measured by voltage sensitive dyes. One eye was closed, the other was visually stimulated. The white bands show the activity of the stimulated eye, the dark bands indicate inactivity. The bands are on average 0.3 mm wide. Electrode recordings (dots) along a track tangential to the cortical surface in layer 4 revealed that the single neuron response was consistent with the optical recording. From [Blasdel1986].

11.5 Intermezzo: Measurement of neural activity in the brain

Single cell recordings have for decades been the method of choice to record the activity of neural activity in the brain. The knowledge of the behaviour of a single cell however does not give information about *structures* of activity, encompassing hundreds and thousands of interconnected cells. We have now a number of methods capable of recording from many cells simultaneously, where the mapping of the activity is at a fair spatial and temporal resolution (for an overview see [Papanicolaou2000a]). The concise overview below is necessarily short, and primarily meant as a pointer.

Electro-Encephalography (EEG)

Electro-encephalography is the recording of the electrical activity of the brain by an array of superficial (on the scalp) or invasive (on the cortical surface) electrodes. Noninvasive EEG recordings are unfortunately heavily influenced by the inhomogeneities of the brain and scalp.

For localization this technique is less suitable due to the poor spatial resolution. Invasive EEG studies are the current gold-standard for localizations, but they come at high cost, and the results are often non-conclusive.

Magneto-Encephalography (MEG)

The joint activity of many conducting fibers leads to the generation of tiny magnetic fields (in the order of *femtoTesla* = 10^{-15}T). The high frequency maintained over a so-called Josephson junction, a superconducting semiconductor junction, is influenced by minute magnetic fields, forming a very sensitive magnetic field detector. Systems have now been built with dozens of such junctions close to the skull of a subject to measure a local map of the magnetic fields (see figure 11.18).

```
Show[Import["KNAW MEG system.jpg"], ImageSize -> 420];
```

Figure 11.18 The 150-channel MEG system at the KNAW-MEG Institute in Amsterdam, the Netherlands (www.azvu.nl/meg/).

```
DisplayTogetherArray[
  Show /@ Import /@ {"MEG-1.gif", "MEG-2.gif"}, ImageSize → 400];
```

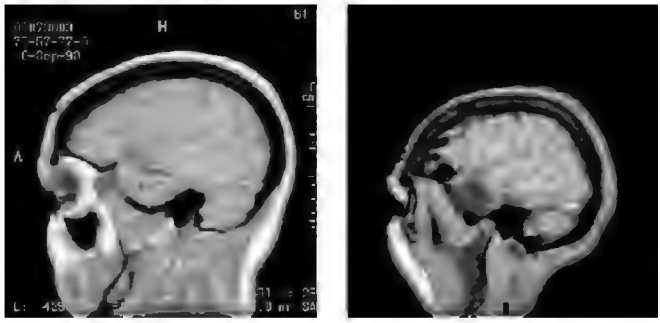


Figure 11.19 Epileptic foci (in red) calculated from magneto-encephalographic measurements superimposed on an MRI image. From: www.4dneuroimaging.com.

From these measurements the inducing electrical currents can be estimated, which is an *inverse* process, and difficult. Though the spatial resolution is still poor (in the order of centimeters), the temporal resolution is excellent.

The calculated location of the current sources (and sinks) are mostly indicated on an anatomical image such as MRI or CT (see figure 11.19).

Functional MRI (fMRI)

Most findings about cortical cell properties, mappings and connections have been found by electrophysiological methods in experimental animals: recordings from a single cell, or at most a few cells. Now, functional magnetic resonance imaging fMRI is able, non-invasively, to measure the small differences in blood oxygenation level when there is more uptake in capillary vessels near active neurons (BOLD fMRI: blood oxygen level dependence). fMRI starts to shed some light on the gross functional cortical activity, even in human subjects and patients, but the resolution (typically 1-3 mm in plane, 2-5 mm slice thickness) is still far from sufficient to understand the functionality at cellular level.

Functional MRI is now *the* method of choice for mapping the activity of cortical areas in humans. Knowledge of the functionality of certain brain areas is especially crucial in the preparation of complex brain surgery. Recently high resolution fMRI has been developed by Logothetis et al. using a high fieldstrength magnet (4.7 Tesla) and implanted radiofrequency coils in monkeys [Logothetis1999].

```
DisplayTogetherArray[
  Import /@ {"fMRI 01 Max Planck.jpg", "fMRI 02 Max Planck.jpg"},
  ImageSize -> 370];
```

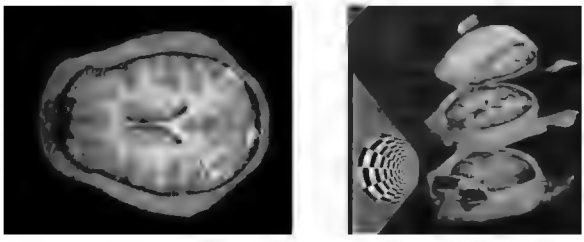


Figure 11.20 Functional magnetic resonance of the monkey brain under visual stimulation Blood Oxygen Level Dependence (BOLD) technique, field strength 4.7 Tesla. Left: Clearly a marked activity is measured in the primary visual cortex. Right: different cut-away views from the brain of the anesthetized monkey. Note the activation of the LGN areas in the thalamus. Measurements done at the Max Planck Institute, Tübingen, Germany, by Nikos Logothetis, Heinz Guggenberger, Shimon Peled and J. Pauls [Logothetis1999]. Images taken from [http://www.mpg.de/pri99/pri19_99.htm].

Optical imaging with voltage sensitive dyes

Recently, a powerful technique is developed for the recording of *groups* of neurons with high spatial and temporal resolution. A voltage-sensitive dye is brought in contact with the cortical surface of an animal (cat or monkey) [Blasdel1986, Ts'o et al. 1990]. The dye changes its fluorescence under small electrical changes from the neural discharge with very high spatial and temporal resolution. The cortical surface is observed with a microscope through a glass window glued on the skull. For the first time we can now functionally map large fields of (superficial) neurons. For images, movies and a detailed description of the technique see: http://www.weizmann.ac.il/brain/images/ImageGallery.html. Some examples of the cortical activity maps are shown in the next chapter.

Positron Emission Tomography (PET)

With Positron Emission Tomography imaging the patient is injected a special radioactive isotope that emits *positrons*, which quickly annihilate with electrons after their emission.

A pair of photons is created which escape at opposite directions, stimulating a pair of detectors in the ring of detectors around the patient. The line at which the annihilation took place is detected with a coincidence circuit checking all detectors. The isotopes are often short-lived, and are often created in a special nearby cyclotron. The method is particularly powerful in labeling specific target substances, and is used in brain mapping, oncology, neurology and cardiology. Figure 11.21 shows an example of brain activity with visual stimulation.

```
Show[Import["PET brainfunction 02.gif"], ImageSize -> 240];
```

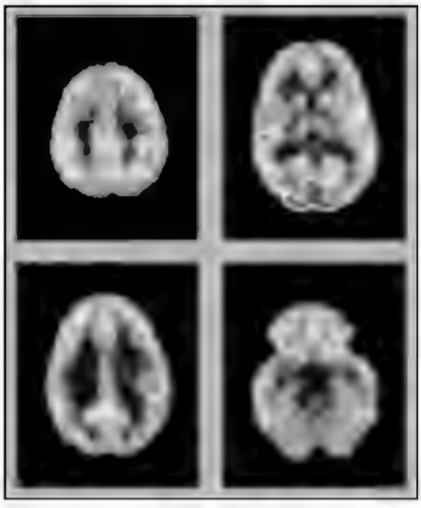


Figure 11.21 Example of a transversal PET image of the brain after visual stimulation. From www.crump.ucla.edu/lpp/. Note the marked activity in the primary visual cortex area.

11.6 Summary of this chapter

The thalamus in the midbrain is the first synaptic station of the optic nerve. It acts as an essential relay and distribution center for all sensorial information. The lateral geniculate nucleus is located at the lower part, and consists of 4 parvocellular layers receiving input from the small retinal midget ganglion cells, and 2 layers with magnocellular cells, receiving input from the larger retinal parasol cells. The parvocellular layers are involved with shape, the magnocellular cells are involved in motion detection.

No preference for a certain scale induces the notion of treating each scale with the same processing capacity, or number of receptive fields. This leads to a scale-space model of retinal RF stacking: the set of smallest receptive fields form the fovea, the larger receptive fields each form a similar hexagonal tiling. With the same number of receptive fields they occupy a larger area. The superposition of all receptive field sets creates a model of retinal RF distribution which is in good accordance with the linear decrease with eccentricity of acuity, motion detection, and quite a few other psychophysical measures.
The receptive field sensitivity profile of LGN cells exhibits the same spatial pattern of on- and off-center surround cells found for the retinal ganglion cells. However, they also show a marked dynamic behaviour, which can be modeled as a modulation of the spatial sensitivity pattern over time by a Gaussian derivative. The scale-space model for such behaviour is that such a cell takes a temporal derivative. This will be further explored in chapters 17 and 20.

In summary: the receptive fields of retinal, LGN and cortical cells are sensitivity maps of *retinal* stimulation. The receptive fields of:
- ganglion cells have a 50% on-center and 50% off-center center-surround structure at a wide range of scales;
- LGN cells have a 50% on-center and 50% off-center center-surround structure at a wide range of scales; they also exhibit dynamic behaviour, which can be modeled as temporal Gaussian derivatives;
- simple cells in V1 have a structure well modeled by spatial Gaussian derivatives at a wide range of scales, differential order and orientation;
- complex cells in V1 have not a clear structure; their modeling is not clear.

The thalamic structures (as the LGN) receive massive reciprocal input from the cortical areas they project to. This input is much larger then the direct retinal input. The functionality of this feedback is not yet understood. Possible mechanisms where this feedback may play a role are geometry-driven diffusion, perceptual grouping, and attentional mechanisms with information from higher centers. This feedback is one of the primary targets for computer vision modelers to understand, as it may give a clue to bridge the gap between local and global image analysis.

12. The front-end visual system - cortical columns

"Better keep yourself clean and bright;
you are the window through which you must see the world"
-George Bernard Shaw

12.1 Hypercolumns and orientation structure

Hubel and Wiesel were the first to find the regularity of the orientation sensitivity tuning. They recorded a regular change of the orientation sensitivity of receptive fields when the electrode followed a track tangential to the cortex surface (see figure 12.1).

```
<< FrontEndVision`FEV`;
Show[
  Import["orientation tangential track 01.jpg"], ImageSize -> 320];
```

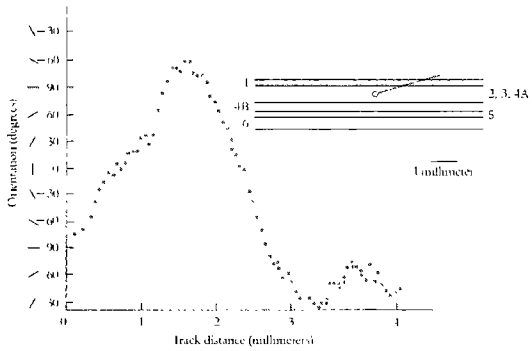


Figure 12.1 A tangential electrode tracking along the surface of the cortex displays a neat ordering of orientation selectivity of the cortical receptive fields. Horizontally the electrode track distance is displayed, vertically the angle of the prominent orientation sensitivity of the recorded cell. From [Hubel1982].

A hypercolumn is a *functional unit* of cortical structure. It is the hardware that processes a single 'pixel' in the visual field for both eyes. There are thousands of identical hypercolumns tiling the cortical surface.

The vertical structure in this small patch of cortical surface does not show much variation in orientation sensitivity of the cells, hence the name 'columns'.

```
Show[GraphicsArray[
   Import /@ {"cortical columns model.jpg", "orientation column.jpg"}],
  ImageSize -> 400];
```

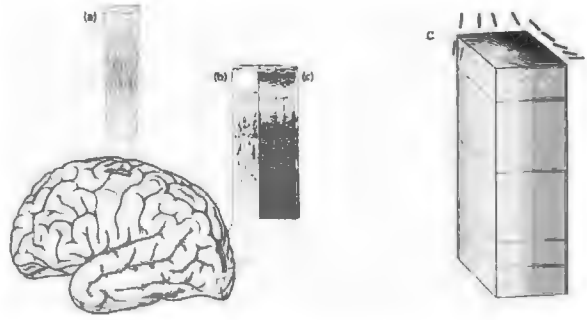


Figure 12.2 Left: Cortical columns are found at all places on the cortex. It is a fundamental organizational and structural element of topological organization. Right: A visual column in V1 contains all orientations (in a pinwheel-like structure). From [Ts'o et al. 1990].

They contain cells of all sizes, orientations, differential order, velocity magnitude and direction, disparity and color for both left and right eye. It is a highly redundant filterbank representation. The left and right eye dominance bands form somewhat irregular bands over the cortical surface (figure 11.16).

From the voltage sensitive dye methods we now know that the fine structure of the orientation sensitivity is organized in a pinwheel fashion [Bonhoeffer and Grinvald 1993] (see figure 12.3), i.e. the spokes connect cells firing at the same orientation. In these measurements the monkey is presented with a multitude of sequential lines at particular orientations.

```
Show[GraphicsArray[Import /@ {"iso-orientation contours.gif",
     "iso-orientation contours zoomed.gif"}], ImageSize -> 400];
```

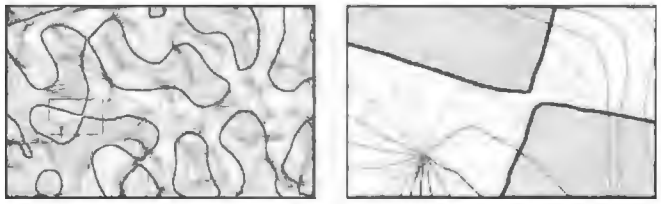


Figure 12.3 Left: Voltage sensitive dye measurements of orientation sensitivity on a small patch of V1 in the macaque monkey. Size of the cortical patch: 9x12 mm. Right: enlarged section of the rectangular area in the left figure. Shaded and unshaded areas denote the left and right eye respectively. Colored lines connect cells with equal orientation sensitivity. They appear in a pinwheel fashion with the spokes in general perpendicular to the column boundary. From [Ts'o et al. 1990].

```
Show[GraphicsArray[Import /@ {"iso-orientation contours bw.gif",
    "orientation columns model.gif"}], ImageSize -> 320];
```

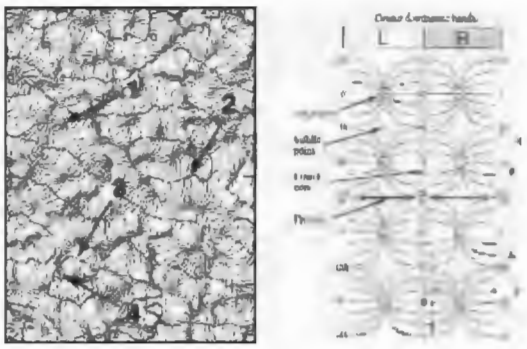


Figure 12.4 Left: The orientation columns are arranged in a highly regular columnar structure. Arrow 1 and 4: the singularity in the pinwheel iso-orientation contours is located in the center. Arrow 2 and 3: Border between left and right eye columns. Note that the iso-orientation contours are in majority perpendicular to the boundaries of the columns. Right: Model to explain the measured results. The iso-orientation lines mostly reach the ocular dominance boundary at a right angle. From [Blasdel and Salama 1986].

It is not known how the different scales (sizes of receptive fields) and the differential orders are located in the hypercolumns. The distance from the singularity in the pinwheel and the depth in the hypercolumn form possible mapping possibilities.

Injection of a pyramidal cell in layer 2 and 3 in a monkey with a dye (horseradish peroxidase) reveals that such cells make connections to cells in neighboring columns (see figure 12.5). The clusters of connections occur at intervals that are consistent with cortical column distances.

```
Show[Import["orientation coupling.gif"], ImageSize -> 220];
```

Figure 12.5 Cells of a hypercolumn at a particular orientation have facilitating connections some distance away. This distance is just to cells in neighboring hypercolumns with the same orientation sensitivity, thus enabling a strong perceptual grouping on orientation, which is essential for the perception of lines, contours and curves. From [McGuire1991], see also [Kandel2000].

```
Show[Import["orientation coupling model.gif"], ImageSize -> 240];
```

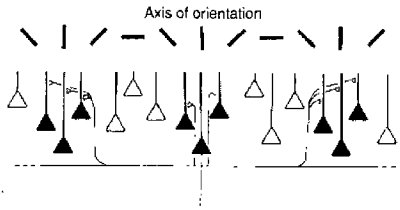


Figure 12.6 Model for the excitatory orientation coupling between neighboring columns with similar orientation. From [Kandel et al. 2000].

This may be particularly important for close-range *perceptual grouping* [Elder1998, Dubuc2001a]. It has been shown that the projections are only with those neighboring cells that have the same functional specificity (a vertical line in this case), see figure 12.6 and [Kandel2000, pp. 542] and [Gardner1999]. See also Zucker [Zucker2001a].

12.2 Stabilized retinal images

```
DensityPlot[-E^(-x^2-y^2), {x, -2, 2}, {y, -2, 2}, PlotRange → {-2, 0},
  Epilog → {White, Point[{0, 0}]}, ImageSize -> 300];
```

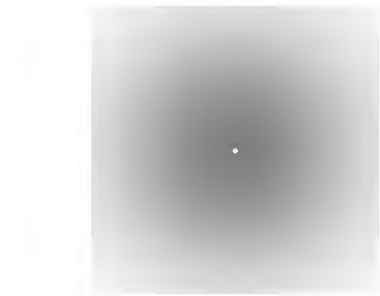


Figure 12.7 Stimulus to experience the disappearance of perception (also called visual fading) when the image is stabilized on the retina. Fixate a long time on the small central white dot. After some time the grey blob completely disappears. From [Cornsweet1970].

Worth mentioning in the context of the *biomimicking* of vision into a mathematical framework is the amazing fact that vision totally disappears in a few seconds (!) when the image is stabilized on the retina [Ditchburn1952, Gerrits1966a].

One can fix the retinal image by an optical projection of a contactlens on the eye attached to a fiber bundle carrying the image, or monitor carefully the eye movements and displace the stimulus image appropriately counteracting the eye movements. One can appreciate this phenomenon with the stimulus depicted in figure 12.7.

▲ Task 12.1 Repeat this experiment with a foreground and background of different colors. What fills in, the foreground or the background?

▲ Task 12.2 Why is a gradual slope of the boundary of the blob stimulus required?

The immediate implication of visual fading with a stabilized retinal image is that we do not perceived homogeneous areas, like a white sheet of paper. We perceive the boundaries, and *fill-in* the white of the paper and the background by measuring what happens on both sides of the intensity contour.

We continuously make very small eye movements, possibly in order to keep seeing. It has been suggested that eye movements play an important role in the perceptual grouping of 'coterminous' (non-accidental) edges [Binford1981].

```
Show[Import["micromovements.jpg"], ImageSize -> 300];
```

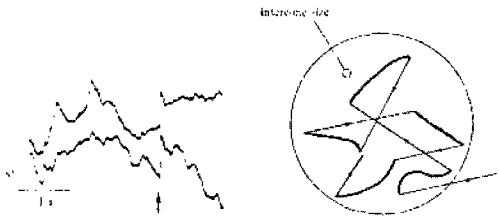


Figure 12.8 Microsaccades are made very precisely synchronously by both eyes, and are much larger than single receptor diameters.
From McCourt, www.psychology.psych.ndsu.nodak.edu.

These small eye movements are substantially larger than single receptors, and are made synchronously by both eyes (figure 12.8). There is also small *drift*.

Figure 12.9 shows some stimulus patterns that continuously seem to shimmer. This is due to your involuntary eye movements (drift, tremor and micro-saccades).

Burst in cortical neurons tend to occur after microsaccades. The investigation of the relation between cortical responses and eye movements (in particular microsaccades) is an active research area, where one investigates the problem of why we have a stable perception despite these eye movements.

```
nϕ = 60; nr = 30; Show[Graphics[
  {Table[{Black, Polygon[{{0, 0}, {Cos[ϕ], Sin[ϕ]}, {Cos[ϕ + π/nϕ], Sin[ϕ + π/nϕ]}}]},
    {ϕ, 0, 2π, 2π/nϕ}], Table[{Black, Disk[{2.2, 0}, r],
     White, Disk[{2.2, 0}, r - 1/(2 nr)]}, {r, 1, 1/(2 nr), -1/nr}]}],
 AspectRatio -> Automatic, ImageSize -> 480];
```

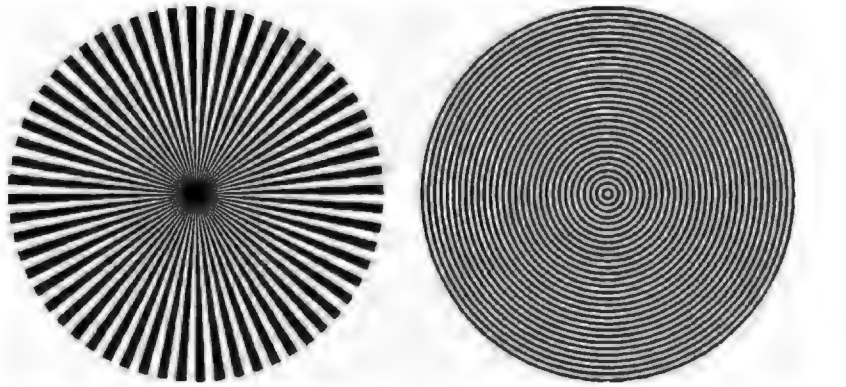


Figure 12.9 Stimulus pattern, in which a shimmering effect is seen due to our small involuntary eye movements. Note: when viewing on screen, adjust the magification of the display for best resolution, and adjust or regenerate the pattern so fresh PostScript is generated. `nϕ` is the number of segmentpairs, `nr` is the number of ringpairs.

12.3 The concept of local sign

How does a cell in the LGN and cortex know from what retinal position the incoming signals arrive? How is the map formed?

The fibers that arrive in the LGN from both retinas all look the same: covered with a white myelin sheet. This is different in our modern electronic equipment, where we carefully *label* each wire with e.g. colors in order to know exactly which wire it is and what signal it is carrying. Somehow the brain manages to find out how the wiring is done, which was formed in the embryonic stage.

This philosophical problem was first studied by Lotze [Lotze1884], a German philosopher. He coined the German term *Lokalzeichen*, which means 'local sign'. Jan Koenderink followed up on this first thinking, and wrote a beautiful paper which put the Lokalzeichen in a computer vision perspective [Koenderink1984d]. The main line of reasoning is the following:Two cells can determine if they have a neighborhood relation if they are *correlated.*

Neighboring geometrical properties have to correspond, such as intensity, contours with the same orientation, etc.

A similar situation occurs when we solve a jigsaw puzzle (see figure 12.10). We know to find the location of a piece due to its relations to its neighbors. In differential geometric terminology: There need to be similar differential structure between two neighboring pieces. The zero-th order indicates that the same intensity makes it highly likely to be connected. So does a similar gradient with the same slope and direction, and the same curvature etc. It of course applies to all descriptive features: the same color, texture etc. The *N-jet* has to be interrelated between neighboring cortical hypercolumns at many scales.

So during the formation of the visual receptive field structures the system seems to solve a huge jigsaw puzzle. Only be looking the cells are stimulated, and the system can accomplish its task. When we are born, the neurons are very redundantly wired. A cell is connected to too many of its neighbors.

Only those synapses that are necessary, remain during the formation process, because these are the ones that are actually used.

A frequently used synapse grows, a hardly or not used synapse degenerates and will never come back.

```
Show[Import["jigsaw.gif"], ImageSize -> 450];
```

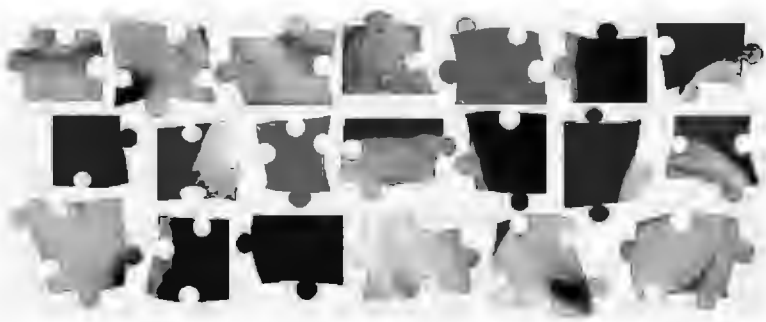


Figure 12.10 We solve a jigsaw puzzle by finding corresponding geometrical properties between pieces that have to be close together, in order to form a pair. In this example (from http://jigzone.com/) the pieces have not been rotated or occluded. In our scale-space model of the front-end visual system the pieces are blobs weighted with multi-scale Gaussian derivative functions.

Receptive fields substantially *overlap*, and they should in order to create a correlation between neighboring fibers. However, they overlap because we have a multi-scale sampling structure. At a single scale, our model presumes a tight hexagonal tiling of the plane. There is a deep notion here of the correlation between different scales, and the sampling at a single location by receptive fields of different scale.

The reconstruction of the receptive field structure when the overlap relations are known is a classical topological problem.

The methodology is beyond the scope of this introductory book. It has been solved for 1D on a circular 1D retina (to avoid boundary effects) by Koenderink, Blom and Toet [Koenderink1984c, Koenderink1984d, Toet1987] but for higher dimensions it is still unsolved.

Note: It is interesting to note that social networks, which can be seen as overlapping 'receptive fields', have recently been discovered as being 'searcheable'. The psychologist Milgram distributed small postal packages among arbitrary people in Nebraska, requesting them to send these to someone in Boston.

Because they did not know this person, there were asked to send the package to someone of which they expected he would know him. To Milgram's surprise it took on average only 6 steps for the packages to reach their target. This has recently been mathematically modeled by Watts et al. [Watts2002].

```
Show[Import["owl.gif"], ImageSize -> 250];
```

Figure 12.11 The solution of the jigsaw puzzle of figure 12.10.

12.4 Gaussian derivatives and Eigen-images

It has been shown that the so-called Eigen-images of a large series of image small patches have great similarity to partial Gaussian derivative functions [Olshausen1996, Olshausen1997]. The resulting images are also often modeled as Gabor patches and wavelets. In this section we will explain the notion of Eigen-images and study this statistical technique with Mathematica.

We read the many patches as small square subimages of δ=12x12 pixels, non-overlapping, at 17 horizontal and 17 vertical position, leading to a series of 289 patches. Figure 12.12 (next page) shows the location of the patches. These 289 images form the input set.

```
im = Import["forest06.gif"][[1, 1]]; δ = 12;
ListDensityPlot[im, Epilog ->
   {Gray, Table[Line[{{i, j}, {i + δ, j}, {i + δ, j + δ}, {i, j + δ}, {i, j}}],
     {j, 2, 256, 15}, {i, 2, 256, 15}]}, ImageSize -> 170];
```

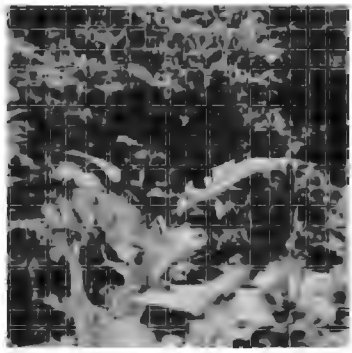


Figure 12.12 Location of the 289 small 12x12 pixel patches taken from a 256^2 image of a forest scene.

The small 12x12 images are sampled with **SubMatrix**:

```
set = Table[SubMatrix[im, {j, i}, {δ, δ}],
  {j, 2, 256, 15}, {i, 2, 256, 15}]; Dimensions[set]
```

```
{17, 17, 12, 12}
```

and converted into a matrix **m** with 289 rows of length 144. We multiply each small image with a Gaussian weighting function to simulate the process of observation, and subtract the global mean:

```
σ = 4; g = Table[Exp[-(x^2 + y^2)/(2 σ^2)], {x, -5.5, 5.5}, {y, -5.5, 5.5}];
set2 = Map[g # &, set, {2}];
m = Flatten[Map[Flatten, set2, {2}], 1]; mean = Plus @@ # / Length[#] &;
m = N[m - mean[Flatten[m]]]; Dimensions[m]
```

```
{289, 144}
```

We calculate $m^T m$, a 144^2 matrix with the **Dot** product, and check that it is a square matrix:

```
Dimensions[mTm = N[Transpose[m].m]]
```

```
{144, 144}
```

The calculation of the 144 Eigen-values of a 144^2 matrix goes fast in Mathematica. Essential is to force the calculations to be done numerically with the function **N[]**. Because **mTm** is a symmetric matrix, built from two 289x144 size matrices, we have 144 (nonzero) Eigen-values:

```
Short[Timing[evs = eigenvalues = Eigenvalues[mTm]], 5]
```

```
{0.046 Second,
 {3.08213×10^7, 9.62621×10^6, 4.30075×10^6, 2.83323×10^6, 1.42025×10^6,
  1.385×10^6, 1.20726×10^6, 890446., 811958., <<126>>, 670.255, 644.039,
  613.394, 503.244, 442.515, 366.791, 284.952, 250.393, 235.25}}
```

We calculate the **Eigenvectors** of the matrix **mTm** and construct the first Eigen-image by partitioning the resulting 144x1 vector 12 rows. All Eigen-vectors normalized: unity length.

```
eigenvectors = Eigenvectors[mTm];
eigenimages = Table[Partition[eigenvectors[[i]], δ], {i, 1, 8}];
DisplayTogetherArray[ListDensityPlot /@ eigenimages, ImageSize -> 460];
DisplayTogetherArray[ListPlot3D /@ eigenimages, ImageSize -> 460];
```

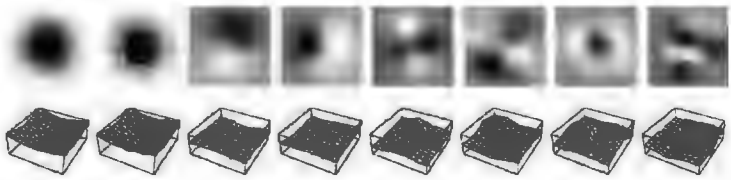


Figure 12.13 The first 8 Eigen-images of the 289 patches from figure 12.10.

Note the resemblance of the first Eigen-image to the zeroth order Gaussian blob, and the second and third Eigen-image to the first order Gaussian derivatives $\frac{\partial G}{\partial x}$ and $\frac{\partial G}{\partial y}$, and the 4th, 5th and 6th Eigen-image to the second order Gaussian derivatives $\frac{\partial^2 G}{\partial x^2}$ under 120 degree different angles.
We will derive in chapter 19 that a second order Gaussian derivative in *any* direction can be constructed from 3 second order partial derivative kernels each 120 degrees rotated (this is the *steerability* of Gaussian kernels, they form a *basis*).

The Eigen-images reflect the basis functions in which the spatial structure of the images can be expressed. The natural basis for spatial image structure are the spatial derivatives emerging in the local Taylor expansion. When there is no coherent structure, such as in white noise, we get Eigen-images that reflect just noise. Here are the Eigen-images for white noise (we take the same 289 12x12 patches again):

```
noise = Table[Random[], {256}, {256}]; δ = 12;
set = Table[SubMatrix[noise, {j, i}, {δ, δ}], {j, 3, 256, 15}, {i, 3, 256, 15}];
m = Flatten[Map[Flatten, set, {2}], 1];
m = N[m - mean[Flatten[m]]]; mTm = N[Transpose[m].m];
{eigenvaluesn, eigenvectorsn} = Eigensystem[mTm];
eigenimagesn = Table[Partition[eigenvectorsn[[i]], δ], {i, 1, 8}];

DisplayTogetherArray[ListDensityPlot /@ eigenimagesn, ImageSize -> 460];
```

Figure 12.14 The first 8 Eigen-images of 289 patches of 12x12 pixels of white noise. Note that none of the Eigen-images contains any structure.

Note that the distribution of the Eigen-values for noise are much different from those of a structured image. They are much smaller, and the first ones are markedly less pronounced. Here we plot both distributions:

```
DisplayTogether[
  LogListPlot[evs, PlotJoined -> True, PlotRange -> {.1, Automatic}],
  LogListPlot[eigenvaluesn, PlotJoined -> True], ImageSize -> 250];
```

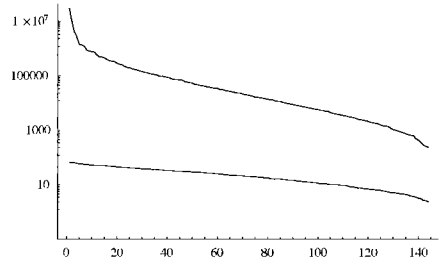


Figure 12.15 Nonzero Eigen-values for a structured image (upper) and white noise (lower).

When we extract 49 x 49 = 2401 small images of 12 x 12 pixels at each 5 pixels, so they slightly overlap, we get better statistics.

A striking result is obtained when the image contains primarily vertical structures, like trees. We then obtain Eigenpatches resembling the horizontal high order Gaussian derivatives / Gabor patches (see figure 12.16).

```
im = Import["forest02.gif"][[1, 1]]; δ = 12;
set =
  Table[SubMatrix[im, {j, i}, {δ, δ}], {j, 2, 246, 5}, {i, 2, 246, 5}];
δδ = (δ - 1) / 2; σ = δδ; g = Table[N[Exp[-(x^2 + y^2)/(2 σ^2)]],
  {x, -δδ, δδ}, {y, -δδ, δδ}]; set2 = Map[g # &, set, {2}];
m = Flatten[Map[Flatten, set2, {2}], 1]; mean = (Plus @@ #)/Length[#] &;
m = N[m - mean[Flatten[m]]]; mTm = N[Transpose[m].m];
eigenvectors = Eigenvectors[mTm];
eigenimages = Table[Partition[eigenvectors[[i]], δ], {i, 1, 25}];
```

```
Block[{$DisplayFunction = Identity}, p1 = ListDensityPlot[im]; p2 =
  Show[GraphicsArray[Partition[ListDensityPlot /@ eigenimages, 5]]];]
Show[GraphicsArray[{p1, p2}, ImageSize -> 420]];
```

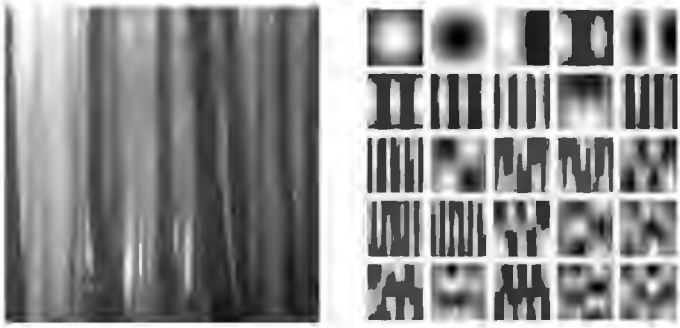


Figure 12.16 Eigen-images for 2401 slightly overlapping patches of 12x12 pixels from the image of figure 12.10. Due to the larger number of patches we get better statistics. Note that the first Eigenpatches resemble the high order horizontal derivatives

▲ Task 12.3 Show the Eigen-images for a range of patch sizes, from 5x5 to 15x15 pixels. How can the result be interpreted?

▲ Task 12.4 The visual cortex receives input from center-surround receptive fields, thus (scale-space model for front-end vision) from the Laplacian of the input image. Show the Eigen-images for the Laplacian of the input image at several scales. Interpret the results, especially with respect to the first Eigen-image.

▲ Task 12.5 Find the (color) Eigen-images for patches taken from a RGB color image.

12.5 Plasticity and self-organization

It can be speculated that the coherent structure in the collective set of first retinal images that we perceive after birth creates the internal structure of the observation mechanism in our front-end visual system. Numerous deprivation studies have shown the importance of the early visual stimulation for visual development.

The closure of one eye during the first 4 months after birth due to some illness or accident, prevents the creation of proper stereopsis (depth vision). It was shown histologically in early deprivation experiments by Hubel and Wiesel [Hubel1988a] that the involved cells in the LGN seriously degenerated in a monkey with a single eye blindfolded the first months after birth. Later deprivation showed markedly less such degeneration effects.

When we arrive on this world, we are redundantly wired. Each neuron seems to be connected to almost all its neighboring neurons. Connections that are hardly used deteriorate and disintegrate, connections that are frequently used are growing in strength (and size).

The same principle is found in the output of the neural circuitry, in the activation of muscles. Muscle cells in neonates are innervated by multiple nerve endings but finally a single ending remains (synapse elimination), exactly in the middle where the minimum of shear forces is felt.

Here the self-organizing parameter may be the shear-stress. The nasal half of the retina projects to a different half of the visual cortex than the temporal half.

The receptive fields on the vertical meridian on the retina have to communicate with each other in different brain halves. The connection is through an abundant array of fibers connecting the two cortical areas. This is the fornix, a bundle easily recognized on sagittal MR slices as the corpus callosum.
How do the millions of fibers know where their corresponding contralateral receptive field projection locations are? The answer is that they don't have to know this. The connections in the redundant and superfluous wiring after birth that are not stimulated by visual scenes on the receptive field just bordering the vertical meridian are degenerating, leaving the right system connections in due time (a few months).

▲ Task 12.6 In this section we took many patches from a single image. Show that similar Eigen-images emerge when we take patches from different images.

▲ Task 12.7 Show the Eigen-images for a variety of other images, e.g. natural scenes, faces, depth maps, ultrasound speckle images. Is there any difference between tomographic slice images (where no occlusion is present) or real world 3D → 2D projection images?

This statistical approach to the emergence of a representation of operators in the early visual system is receiving much attention today, see e.g. [Rao2001]. Keywords are principal component analysis (PCA), partial least squares (PLS), canonical correlation analysis (CCA), independent component analysis (ICA), multiple linear regression (MLR) and sparse code learning (SCL). It is beyond the scope of this book to elaborate further on this topic.

▲ Task 12.8 What determines the orientation angle, the sign and the amplitude of the emerging Eigen-images?

▲ Task 12.9 This formalism is now easily extended to images as scale-spaces. When a small stack of Gaussian scale-space images is brought into a column vector format, the same apparatus applies. Show the Eigen-images for such

scale-spaces, and set up conjectures about the self-emerging operators when the neural system is presented with large series of scale-space images as input.

12.6 Higher cortical visual areas

```
Show[Import["cortical functional areas.jpg"], ImageSize -> 430];
```

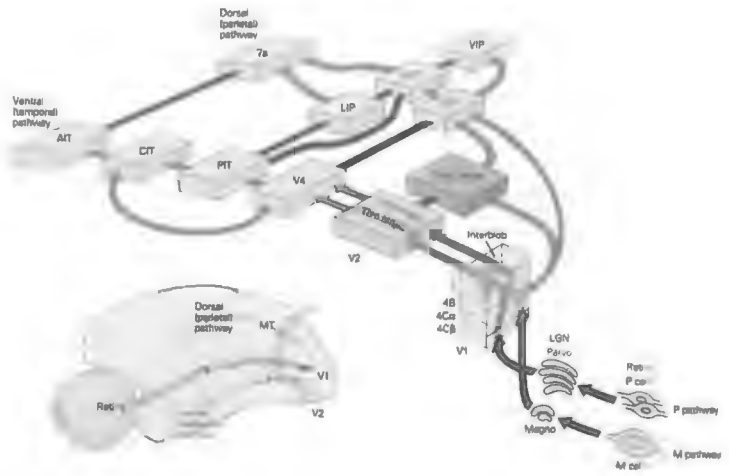


Figure 12.17 Functional diagram of the visual pathways. From [Kandel2000].

From V1 projections go to the higher visual layers in the cortex, such as V2, V3, V4 and the MT (medio-temporal) layer.
It is beyond the scope of this chapter on front-end vision to discuss all layers in detail. Figure 12.17 summarizes the principal connections. It is clear from this diagram, that the visua system has dedicated pathways through the multiple visual areas. They are related to the separate functional properties that are measured: shape, color, motion, and disparity. The current view is that there are two major pathways: a ventral pathway to the inferior temporal cortex and a dorsal pathway to the posterior parietal cortex. The projections in each pathway are hierarchical, there are strong projections between the subsequent levels as well as strong feedback connections.

The type of visual processing changes systematically from one level to the next (see the excellent overview by Robert Wurtz and Eric Kandel [Kandel2000, chapter 28]).

12.7 Summary of this chapter

The visual cortex is a somatotopic visual map of the retina. The visual cortex consists of 6 structural layers, which can be discriminated by cell type and function. The input from the LGN is in the middle layer 4, the output to higher centres is from the layers above and below. The visual cortex is a hierarchical cascade of functional areas. V1, V2, V3, V4 and up. In V1 a precise arrangement of hypercolumns is shown by both single electrode studies and voltage sensitive dye functional mapping methods.

Each hypercolumn contains the cellular machinery for complete binocular analysis of a 'pixel' in the overlapping visual field of both eyes. The cells sensitive for the same orientation are located on 'spokes' of a pinwheel-like arrangement in the hypercolumn.

The cortex has to infer spatial arrangements of incoming fibers from the mutual relations between the signals they carry. The concept of 'local sign' compares this inference with the solving of a jigsaw puzzle from the N-jet and orientation information of receptive fields in neighboring hypercolumns.

The structure of spatially and temporally coded receptive field sensitivity profiles is largely constructed after birth by seeing.

The plasticity of these formations has been shown by deprivation experiments in experimental animals, and can be modeled by self organizing neural networks. An analysis of the Eigen-images for a large number of patches taken from an image shows great similarity of the *basis* images with the Gaussian derivative receptive field models.The first layers of the visual front-end (up to V2) are well mapped and studied. This is much less the case for the higher cortical areas. A general strategy seems to be the hierarchical mapping onto subsequent layers with substantial divergent feedback connections.

12.8 Vision dictionary

Below is a short explanation of the vision related terms used in this book. For more complete listings see e.g. the *Visionary* webpages by Liden: (http://cns-web.bu.edu/pub/laliden/WWW/Visionary/Visionary.html) and the glossary by Fulton (http://www.4colorvision.com/pdf/glossary.pdf).

area 17	-	visual cortex, in the cortical area. Classification by Brodmann
afferent	-	direction for nerve signals: from the receptor to the brain
amacrine cell	-	retinal cell, likely involved in motion processing
axon	-	output fiber of a neuron
bipolar cell	-	cell in the retina with two possible synaptic projection polarities
blob	-	group of color processing cells in the center of a hypercolumn
coronal	-	vertical 3D slice direction, plane through eye centers and cheek
corticofugal	-	from the cortex away
caudal	-	on the side of the tail
cranial	-	on the side of the top (of the head)
dendrite	-	treelike branch on a cell body for reception of synaptic input

depolarization - decreasing the inner voltage of the cell body
dorsal - on the side of the back
efferent - direction for nerve fibers: from the brain to effector (muscle, gland)
excitatory - positive input increasing firing rate
fovea - the central portion of the retina with highest acuity
ganglion cell - cell type in the retina, the output cells
hypercolumn - functional unit of the cortex (typ. 1x1x2 mm)
hyperpolarization - increasing the inner voltage of the cell body
inhibitory - negative input, decreasing firing rate
lateral - located at the side
macaque - short-tailed asian monkey, often used in experiments
magno-cellular - with large cell bodies (Latin: magnus = big)
midget cells - small retinal ganglion cells, involved in shape extraction
myelin - insulating layer of a neuron's axon, white colored
nasal - located at the side of the nose
nucleus - a localized small structure in the brain
occipital - located in the back (of the head)
optic chiasm - crossing of the optic nerve
orientation column - column of cortical cells containing all local orientations
parietal - on the side
parvo-cellular - with small cell bodies (Latin: parvus = small)
parasol cell - large retinal ganglion cells involved in motion
PSTH - post-stimulus-time-histogram
psychophysics - measurement of human performance in perceptual tasks
Pacini receptor- onion-shaped pressure sensitive receptor in the skin (after Filippo Pacini (1812–1883), Italian anatomist)
receptive field - 2D spatial light sensitivity area *on the retina* with respect to the cell's firing rate
retinotopic - forming a spatial, neighbor-preserving mapping with the retina
rhodopsin - the light-sensitive protein in the rods and cones
sagittal - vertical 3D slice direction (Latin: sagitta = arrow)
soma - cell body (Greek: soma = body)
somatotopic - forming a spatial map with a surface somewhere on the body
striate - striped (Latin: stria = furrow)
striate cortex - area 17. Area 17 is striped, due to a marked stripe of white matter in layer 4 of myelinated axons
synapse - tiny pedicle where one neuron passes information to the next
temporal - located at the temple of the head (Latin: tempus = temple)
thalamus - deep brain structure in the midbrain, receiving all perceptual information
transversal - horizontal 3D slice position
ventral - on the side of the belly

12.8.1 Further reading on the web:

Some suggested webpages for further exploration:

Journal: Perception
www.perceptionweb.com/perabout.html
Space-time receptive fields in the visual system (Ohzawa, Berkeley):
neurovision.berkeley.edu/Demonstrations/VSOC/teaching/AA_RFtutorial.html
Voltage sensitive dye research at Weizmann Institute of Science, Israel:
www.weizmann.ac.il/brain/grinvald/index.html
Optical recording literature (compilation by Steve M. Potter):
www.its.caltech.edu/~pinelab/optical.html
LGN research (Dwayne Godwin):
www.wfubmc.edu/bgsm/nba/faculty/godwin/godwin.html
Center for Computational Vision and Control (early-vision models and biomedical image analysis): cvc.yale.edu/
Magneto-encephalography (MEG): www.4dneuroimaging.com/
Positron Emission Tomography (PET): www.crump.ucla.edu/lpp/
Functional Magnetic Resonance Imaging (fMRI): www.functionalmri.org/,
www.spectroscopynow.com/Spy/mri/
Medical images and illustrations: www.mic.ki.se/Medimages.html

13 Deep structure I. watershed segmentation

Erik Dam and Bart M. ter Haar Romeny

"Study the family as a family, i.e. define deep structure, the relation between structural features of different derived images" [Koenderink1984a].

13.1 Multi-scale measurements

The previous chapters have presented the notion of *scale* - any observation is, implicitly or explicitly, defined in terms of the area of support for the observation. This allows different observations at the same location that focus on different structures. A classical illustration of this concept is the observation of a tree. At fine scale the structures of the bark and the leaves are apparent. In order to see the shapes of the leaves and the twigs a higher scale is required; an even higher scale is appropriate for studying the branches, and finally the stem and the crown are best described at a very coarse scale.

A comprehensive description of the tree requires observations at all scales ranging from the cellular level to the scale corresponding to the height of the tree. However, in order to give a full description of the tree, subsequent inspection at all the relevant scales is not sufficient. Even though it is possible to measure the size of the crown and the number of leaves at specific fixed scales, inspection of the connections between the structures requires simultaneous inspection at all the intermediary scales. If we want to count the number of leaves positioned at one of the trees major branches, it is necessary to link the localization of the leaves at fine scale through the twigs (possibly at an even finer scale), the thin branches and finally reaching the desired branch at a coarser scale.

The key point is that not only do we need to connect observations at different localizations - we also need to *link* observations at different scales.
In the words of Koenderink, we must study the family of scale-space images as a family, and define the 'deep' structure. 'Deep' refers to the extra dimension of scale in a scale-space, like the sea has a surface and depth.

As demonstrated in the previous chapters, differential operators allow detection of different features at a given, fixed scale - this is the *superficial structure*. The scale can be adaptive, and different in every pixel. This brings up the notion of *scale-adaptive* systems like the geometry-driven diffusion equations (in the Perona & Malik equations the scale of the operator is adapted to the length of the gradient), and of *scale selection*.

This chapter will give examples of how linking of observations through the entire scale-space offers additional information about the observed features.
The examples cover mechanisms for automatic choice of the appropriate observation scale, localization of the proper location for features, and a robust segmentation method that takes advantage of the deep structure.
The chapter is concluded with some thoughts on how the deep structure can be used to extract the significant hierarchical information about the image structures and thereby give a potentially comprehensive and compact description of the image.

13.2 Scale selection

A first step towards exploitation of the deep structure is the automatic selection of the appropriate scale for an observation [Lindeberg1998a, Lindeberg1998b]. Differential operators are applied to the detection, localization and characterization of a large number of image features such as edges, blobs, corners, etc. The responses from these operators depend on the scale at which the operators are applied - therefore it is essential to choose the proper scale for the observation. Furthermore, the responses from these differential operators are often used to characterize the properties of the feature - for instance the strength of an edge or the size of a blob. Since the responses from the operators depend on the scale it is not trivial to compare the strength of edges detected at different scales.

As an example, we inspect the standard blob detector defined by the spatial maxima of the Laplacian, $\nabla^2 L = L_{xx} + L_{yy} = L_{ww} + L_{vv}$, where L is the luminance. The Laplacian can be considered a measure of the blob strength in a point - thereby the maxima define the blobs. In order to detect both light and dark blobs we use the absolute value of the Laplacian:

```
<< FrontEndVision`FEV`;
blobness[im_, σ_] := Abs[gD[im, 2, 0, σ] + gD[im, 0, 2, σ]];
```

This is illustrated for an image with a blob and a square, where the blobs are detected at two different scales. In order to pick the blobs from these images, we first define a function that allows extraction of the positions and values for the n largest maxima (with respect to a $2*D$ neighborhood) from an image *of arbitrary dimension*:

```
nMaxima[im_, n_] := Module[{l, d = Depth[im] - 1},
  p = Times @@ Table[(Sign[im - Map[RotateLeft, im, {i}]] + 1)
      (Sign[im - Map[RotateRight, im, {i}]] + 1), {i, 0, d - 1}];
  l = Length[Position[p, 4^d]];
  Take[Reverse[Union[{Extract[im, #], #} & /@ Position[p, 4^d]]],
   If[n < l, n, l]]]
```

With the **Sign** function we first find maxima in the columns, then in the rows, and we multiply (boolean 'and') the result to find the image maxima.The **Sign** function is -1, 0 or 1 with negative, zero or positive value of the argument, so we divide by $2^4 = 16$. We then **Extract** in **imb** the intensity values at every **Position** where **p** is 1, **Union** the result (return sorted distinguished entries) and **Reverse** this list so the largest intensity comes first, **Take** the first **n** values, and return the intensity value at every **Position** in this list.

The two blobs with largest feature detector responses are extracted from an example image at two different scales ($\sigma = 6$ and $\sigma = 10$ pixels), see figure 13.1. The blobs are illustrated by red dots and circles (at the detected blob location, radius proportional to the blob strength):

```
im = 1000 Import["squareblob.gif"][[1, 1]];
Block[{$DisplayFunction = Identity}, p1 = ListDensityPlot[-im];
(m = nMaxima[imb = blobness[im, #], 2]; Print["σ=", #, ", pos =",
      Part[#, 2] & /@ m, "\n resp = ", Part[#, 1] & /@ m];) & /@ {7, 11};
p2 = ListDensityPlot[blobness[im, #], Epilog ->
       {PointSize[.03], Hue[1], Point[Reverse[Part[#, 2]]] & /@ m,
        Circle[Reverse[Part[#, 2]], Part[#, 1]] & /@ m}] & /@ {7, 11}];
Show[GraphicsArray[Prepend[p2, p1]], ImageSize -> 300];
```

```
σ=7, pos ={{50, 15}, {20, 42}}
 resp = {11.6499, 10.6793}

σ=11, pos ={{23, 42}, {50, 15}}
 resp = {5.91633, 2.03575}
```

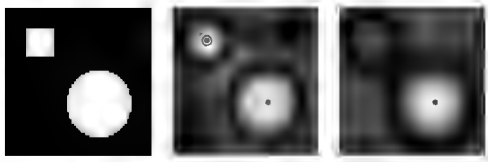


Figure 13.1 Detection of the two best blobs from a test image at two scales. At low scale ($\sigma = 7$) px the square is the most blob-like feature - the circle is too large to be properly detected at this scale. At high scale ($\sigma = 10$ px) the circle is detected as the most blob-like shape, but the response is far less than the response for the circle at low scale since the responses generally decrease with increasing scale. Note that the center coordinates displayed are (x, y) where the origin is the lower left corner. The test image is 64x64 pixels with values 0 and 1.

At both scales, the two largest blobs are detected approximately at the centers of the square and the blob. The response from the blob detector is largest for the square at the low scale. This is not surprising since the scale simply is closer to the size of the square than the blob. At the high scale the response is significantly higher for the blob than the square. However, the response for the blob at high scale is still not higher than the response for the square at low scale. Therefore, if we were to choose the single most blob-like feature in the image, we would select the square.

The problem is a well-known property of the gaussian derivatives. In general, the amplitude of a spatial gaussian derivative is degrading with increasing scale [Lindeberg1994a]. Therefore, the response from the circle is weaker simply because it is best detected at a higher scale since it is larger than the square.

13.3 Normalized feature detection

The classical solution to these problems is to express the differential operators in terms of *normalized derivatives* [Lindeberg1993h, Lindeberg1994a]. The basic idea, as we recall from chapter 3, is to measure the spatial coordinates relative to the scale - instead of using length x we use the dimensionless coordinate $\xi = x/\sigma$ which is normalized with respect to scale [Florack1994b]. The normalized derivative operator for the function f with respect to the spatial coordinate x is then $\frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial (x/\sigma)} = \sigma \frac{\partial f}{\partial x}$.
The standard feature detectors consisting of combinations of differential operators can now be reformulated as normalized detectors. Some of the most common detectors are presented in [Lindeberg1996b].

Analogously, since the Laplacian includes second order derivatives, the normalized blob detector should instead be $\sigma^2 \nabla^2 L = \sigma^2 (L_{xx} + L_{yy}) = \sigma^2 (L_{uu} + L_{vv})$, leading to the normalized feature strength measure function:

```
nblobness[im_, σ_] := Abs[σ^2 (gD[im, 2, 0, σ] + gD[im, 0, 2, σ])];
```

Reassuringly, the normalized detector points out the blob as being the most blob-like feature in the image:

```
Block[{$DisplayFunction = Identity}, p1 = ListDensityPlot[-im];
(m = nMaxima[imb = nblobness[im, #], 2]; Print["σ=", #, ", pos =",
      Part[#, 2] & /@ m, "\n resp = ", Part[#, 1] & /@ m];) & /@ {7, 11};
p2 = ListDensityPlot[nblobness[im, #], Epilog -> {PointSize[.03],
        Hue[1], Point[Reverse[Part[#, 2]]] & /@ m, Circle[
           Reverse[Part[#, 2]], Part[#, 1] / 100] & /@ m}] & /@ {7, 11}];
```

```
σ=7, pos ={{50, 15}, {20, 42}}
 resp = {570.843, 523.284}

σ=11, pos ={{23, 42}, {50, 15}}
 resp = {715.876, 246.325}
```

```
Show[GraphicsArray[Prepend[p2, p1]], ImageSize -> 300];
```

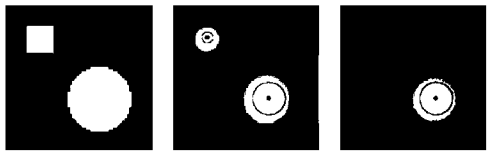


Figure 13.2 Detection of the two best blobs from a test image at two scales with normalized blob detector. The blob gets the highest response (at high scale). The square gets a lower response (at low scale). Compare with figure 13.1.

13.4 Automatic scale selection

The example above depends on choosing appropriate scales for the illustrations. However, since the normalized feature detector allows comparison of detector responses across scale, this can be done automatically. For a given scale, features are detected in an image as the spatial local maxima of the feature detector. When the same features are detected for all level in a scale-space, the features are detected on a number of consecutive scales. The scale at which a feature is best observed is the scale where the normalized feature detector has the strongest response. The means that the points must not only be local maxima for the feature detector in the spatial direction but also in the scale direction.

Since the function *nMaxima* was previously defined for an arbitrary dimension we can use that for detection of maxima in scale-space as well. Superimposed on the example image are the detected blobs illustrated with a circle with radius proportional to the detection scale:

```
im = Import["blobs.gif"][[1, 1]];
blobSS = Table[nblobness[im, Exp[τ]], {τ, 1.7, 2.7, .1}];
maxs = nMaxima[blobSS, 4]; blobSizeFactor = 1.5;
Table[Print["Blob strength: ", maxs[[i, 1]],
  ", (x,y): ", Reverse[maxs[[i, 2, {2, 3}]]]], {i, 4}];

Blob strength: 0.72431, (x,y): {57, 107}

Blob strength: 0.714045, (x,y): {94, 31}

Blob strength: 0.689219, (x,y): {27, 56}

Blob strength: 0.439206, (x,y): {97, 101}

ListDensityPlot[im, Epilog → {Hue[1], Thickness[.01], Dashing[{0.03, 0.02}],
    Table[Circle[{maxs[[i, 2, 3]], maxs[[i, 2, 2]]}, blobSizeFactor
       Exp[(maxs[[i, 2, 1]] - 1) .1 + 1.7]], {i, 4}]}, ImageSize → 130];
```

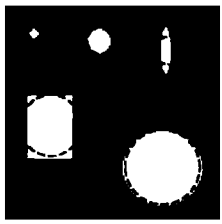


Figure 13.3 Detection of the four most blob-like features in a test image with different shapes. The blobs are superimposed as circles with a radius proportional to the scale the blobs was detected at. The output of the feature response reveals that the circles have somewhat higher feature strength (between 0.71 and 0.72) than the square (at 0.68). The scale-space is generated in 11 exponential levels from scale $\sigma = e^{1.7}$ to $\sigma = e^{2.7}$. The test image is 128 by 128 with values 0 and 1.

Inspection of the output reveals that there has been detected three blobs with similar normalized responses (between 0.69 and 0.72) and one with a significantly lower response (0.43), and that the three blobs with high responses actually correspond with the symmetric blob-like objects in the image. The low-response object is the ellipse.

It is also apparent that the detection scales and the sizes of the blobs are approximately proportional. With a model of an ideal blob, this can be formalized through analysis of the response from the blob detector. This is done for a gaussian bell-shaped model of a blob in [Lindeberg1994a] revealing that the width of the blob and the detection scale are in fact proportional.

13.4.1 λ-Normalized scale selection

Often, the normalized derivative operator has an added parameter λ which allows the operator to be tuned to the specific feature detectors. Instead of writing the normalized n-th order derivative as $L_{i_1 .. i_n - \text{norm}} = \sigma^n L_{i_1 .. i_n}$ (an often used notation, slightly different from the one used above), the normalized derivative has the extra parameter λ:

$$L_{i_1 .. i_n - \text{norm}} = \sigma^\lambda L_{i_1 .. i_n}$$

Intuitively, Lindeberg determines the free parameter λ based on an analysis of the dimension of the specific feature. For instance, for a blob λ is twice the λ for an edge since the edge is essentially only a 1-dimensional feature where the blob is 2-dimensional. For a more thorough treatment of λ-normalized derivatives see [Lindeberg1994a].

Another approach for determining the λ parameter is to use the local fractal dimension (defined in terms of the Hausdorff dimension) as a descriptor of the local complexity and thereby estimate the λ-parameter [Pedersen2000].

13.4.2 Is this really deep structure?

Conceptually, we follow the singularity points for the feature detector through scale-space and locate the scale where the normalized feature strength is maximal. This is the appropriate scale for detecting the feature and for extracting information about the feature. However, we have more information: the nice continuous behaviour across scales allows us to locate the optimal scale explicitly as the *singularities* for a local differential operator output in scale-space.

This approach allows us, e.g. for the Laplacian operator, to select the most blob-like features in the image and establish their approximate size. In the examples we have not given any explicit method for determining whether or not to consider these features for actual blobs - we have simply selected the n best. For a specific application it would be appropriate to determine a threshold for the feature strength of the blob detector - or to establish a more elaborate scheme.

In a sense, the approach ignores the deep structure: we don't explicitly follow the singularity points through scale-space.

However, when establishing the location of the blob, we rely heavily on the deep structure. The implicit assumption is that the singularity points form strings that are approximately vertical in scale-space, meaning that the location of the blob is equivalent with its location at the detection scale.

Blobs are "nice" - they move relatively little as the image is blurred. This makes the assumption reasonable for practical applications. Other feature are not so stable across scale. For these we would have to track the singularity string down in scale from the detection scale in order to establish a more precise localization of the feature. The following sections investigate this approach.

▲ Task 13.1 For a Gaussian bell-shaped blob the width of the blob and the detection scale obtained with the normalized blob detector are proportional [Lindeberg1994a]. Analyze whether this is the case for a circular step-edge blob like the ones in the illustrations. Determine whether the proportionality factor above is appropriately assigned (blobSizeFactor = 1.5).

13.5 Edge focusing

When an image is blurred through convolution with Gaussians of increasing aperture, the noise in the image is gradually removed. However, the salient features are blurred as well. Obviously, small scale structures vanish at high scale, but also the structure that remains is affected by the blurring. When the aperture is increased the local features are influenced by a large neighborhood in the image.

This process changes the appearance of objects. As we saw with the rectangles in the previous section, the shapes are simplified towards rounded shapes. The blurring also dislocates the objects and their edges.

To a large extent we avoided dislocation in the previous section because the objects in the test images had nice spacing in between - thereby the "interaction" between the objects during blurring was insignificant. For more complicated images this effect will be more pronounced.

This section investigates how to take advantage of the deep structure in order to link the simplified large scale structure to the fine scale origins. Specifically, we illustrate how the edges can be tracked down through scale in order to establish the precise location of the edge.

13.5.1 Simplification followed by focusing

In chapter 1 we saw a first example that a structure, like an edge, could emerge from the noise when we increased the scale of the operator. Both the scale of the structure and the scale of the operator had to be larger than the fine grain structure of the noise. We repeat the experiment for the detection of the gradient magnitude of a circle in additive uncorrelated (white) noise.

```
noise = Table[2550 Random[], {256}, {256}];
noisyDisk = Import["blackdisk256.gif"][[1, 1]] + noise;
DisplayTogetherArray[Prepend[ListDensityPlot[
    Sqrt[gD[noisyDisk, 1, 0, #]^2 + gD[noisyDisk, 0, 1, #]^2] & /@
  {0, 1, 2, 3}, ListDensityPlot[noisyDisk]], ImageSize → 480];
```

Figure 13.4 Left: A circular disk embedded in noise. S/N=0.1: Image intensity range [0,255], noise intensity range [0,2550]. Right: Image gradient at $\sigma = 1$, 2.7, 7.4, and 20 pixels. Image resolution 256^2. Only at large scales the contour of the disk emerges.

At scale $\sigma = e^2$ ($\simeq$ 7.4 pixels) we see the contour emerging from the noise. It is difficult to find the locations of the edges at fine scale filtering.

It was suggested by Fredrik Bergholm [Bergholm1987] to use the large scale representation of edges as a guide for the localization of edges at a finer scale. A gradient scale-space is constructed and edges are located at a coarse scale where they are clearly visible, or can easily be extracted with some thresholding (or e.g. a zero-crossing detection of the derivative of this gradient image in the gradient direction, L_{ww}). Edges at finer scales are then found by searching a small neighborhood around the coarse-scale pixel - the scale-space is traversed downwards until the finest scale is reached. The spatial location of the end of the trace is considered the location of the edge that 'lived' until the coarse scale where the search started. This procedure is named *edge focusing*.

13.5.2 Linking in 1D

The phenomenon is studied in somewhat more detail - for the sake of simplicity in 1D.

```
noisystep = Table[5.5 Random[] + 2 Tanh[2 (i - 0.7)] +
    Tanh[4 (i - 1.0)] + 1.4 Tanh[(i - 1.4)], {i, -2, 4, 0.011}];
ListPlot[noisystep, PlotRange -> All, PlotJoined -> True,
  ImageSize -> 230];
```

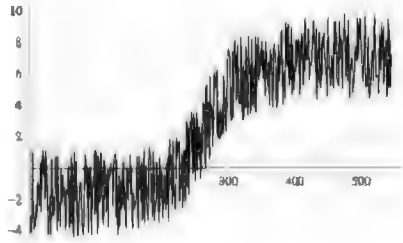


Figure 13.5 A noisy 1D discrete edge signal. Length 546 samples.

We define an edge as a point with maximal absolute incline. These are the zero-crossings for the second order derivative. We are only interested in the *location* of the edge so we don't need to consider normalizing the operators involved.

First we generate a noisy 1D edge:

The signs for the second order derivative is inspected in order to locate the edge. If there is a change in sign between the 'pixel' just left of each 'pixel' of the second derivative, we have an edge location. The graph of the sign-change of a signal as a function of scale is denoted the *signature* of the signal.

We first calculate a scale-space of the second derivative on an exponential range of scales, and take the **Sign**. By mapping the 'difference with your neighbor' function **(RotateRight[#]-#)&** on each scale level, we get the following result (the function **gDf1D** is a version of the Gaussian derivative function **gD** implemented in the Fourier domain for 1D signals):

```
scalespaceLxx = Table[gDf1D[noisystep, 2, E^τ], {τ, 0, 4.5, .015}];
signature = (RotateRight[#] - #) & /@ Sign[scalespaceLxx];
ListDensityPlot[signature, AspectRatio → .5, ImageSize -> 500];
```

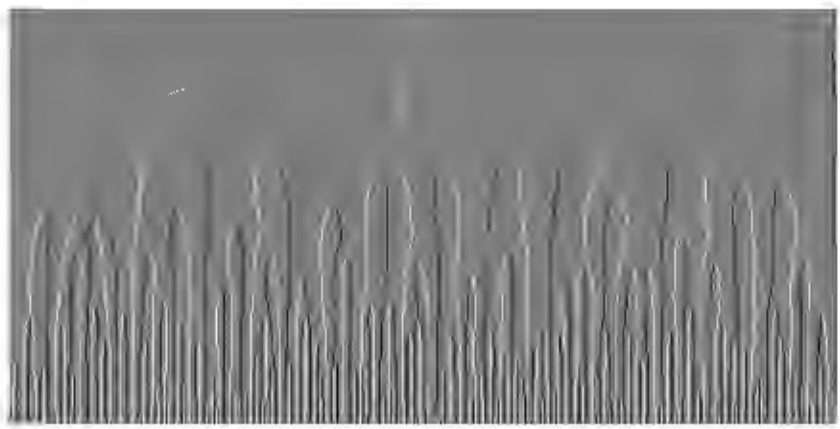


Figure 13.6 The signature function of the noisy edge. Exponential scale-space, scale range $\sigma = \text{Exp}[0 < \tau < 5]$, 251 scale levels. Edge focusing exploits the signature function by tracking the most prominent features down to finer scales. The negative (black) edge trace at the border of the signal is a consequence of the cyclic representation of the signal in the implementation.

Positive upgoing edges are white, negative edges are black in the signature plot. We notice the clear emergence of one edge, surviving much longer over scale then the other edges.

The notion of *longevity* can be viewed of a measure of importance for singularities [Witkin83]. The semantical notions of *prominence* and *conspicuity* now get a clear meaning in scale-space theory.

In a scale-space we see the emergence of the *hierarchy* of structures, in this example we see edges emerging. A second thing to notice is the arched structure of the smaller edges over scale. Positive and negative edges come together and *annihilate*.

These arches show interesting behaviour: sometimes three edges come close together, then two of them annihilate, and one edge continues. We see here a first example of the behaviour of *singularity points* over scale, in this example the behaviour of extrema for the gradient. In the next sections we will study this behaviour in much more detail: this is the analysis of the behaviour of structures over scale, the *deep structure*.

```
edgefocus[signature_, startlevel_, dir_] := Module[{a, b, c, out},
  out = 0. signature; a = Position[signature[[startlevel]], dir];
  Do[b = Position[signature[[i]], dir];
   c = Select[b, (Position[a, # - 1] ≠ {} ||
        Position[a, #] ≠ {} || Position[a, # + 1] ≠ {}) &];
   out[[i]] = ReplacePart[out[[i]], -1, c]; b = c; a = b,
   {i, startlevel - 1, 1, -1}]; out]
focused1 = edgefocus[signature, #, 2] & /@ {170, 280};
focused2 = edgefocus[signature, #, -2] & /@ {170, 280};
showarray[{focused1, focused2}, Frame -> True, FrameTicks -> False];
```

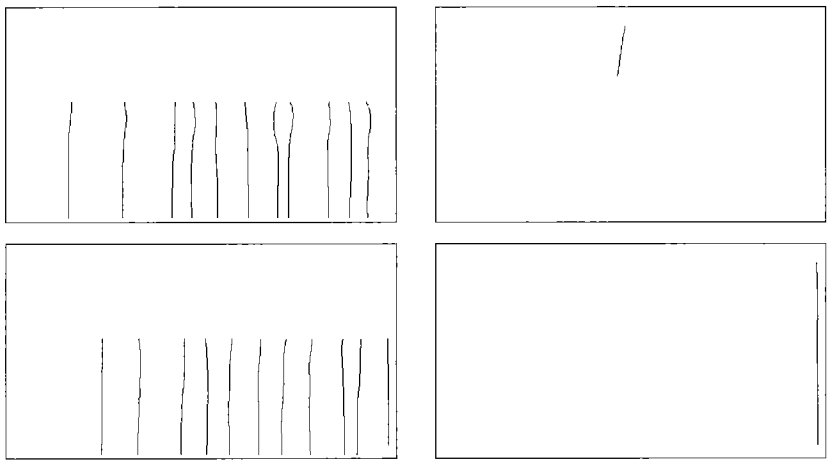


Figure 13.7 Edge focusing for the signature function of our noisy step edge. Top row: two different start levels for the search downwards for negative edges. Bottom row: idem for positive edges. In both cases a sharp edge position is reached at the bottom level, i.e. at the original image. Compare with figure 13.6.

The edge focusing itself is implemented below for 1D signals. From a signature a copy is made with zero's, and a start level is chosen from which to start the search downwards. At the first level the positions are found (**a**) of the edge direction (**dir** $= -2$ or $+2$). From the level below (**b**) those edges are selected (**c**), that have a position that is -1, 0 or $+1$ different from the position in the level above. The entries in the copy signature are then replaced at the found positions with -1's, so they plot as black lines.

Even though the linking scheme is quite simple and heuristic it reveals the potential.

The simplified large scale representation of the image is used to single out the prominent edge. The deep structure of the singularity strings are then used to link the detected edges down to the fine scale where the edges can be precisely located.

▲ Task 13.2 In 2D the zero-crossings of the second derivative in the gradient direction (L_{ww}) are curves, and form non-intersecting *surfaces* in scale-space. Develop a routine in *Mathematica* to compute and display such surfaces. Show the internal structure by plotting cut-away views.

13.6 Follicle detection in 3D ultrasound

Edge focusing is particularly advantageous in noisy data, such as diagnostic ultrasound. We discuss the application of detecting and analyzing follicles in 3D ultrasound [TerHaarRomeny1999a]. Part of the implementation is due to Kalitzin.

```
DisplayTogetherArray[Show /@
   Import /@ {"3dus probe.jpg", "3dus-slice75.gif"}, ImageSize -> 300];
```

Figure 13.8 Left: 3D ultrasound probe (Kretz Ultrasound) sweeps a 2D sector across in a few seconds, effectively acquiring a pyramidal data volume. From this a rectangular Cartesian equidistantly sampled dataset is interpolated. Right: slice through the left female ovary, showing the follicles as dark hypo-echoic (i.e. with low echo signal) spherical volumes. Size of the image 212x164 pixels, approx. 4x3 cm.

Knowledge about the status of the female reproductive system is important for fertility problems and age-related family planning. The volume of these fertility requests in our emancipated society is steadily increasing.
The number of the female egg cells (follicles) in both ovaries decreases roughly linearly from 10^6 at birth to none at the start of the menopause. The detection, counting, shape analysis and growth response to hormonal stimulation of follicles is an important diagnostic procedure for ovarian aging.

This procedure is however labour-intensive and error prone, making a computer aided analysis system a welcome addition. Intravaginal 3D ultrasound imaging of the follicles in the ovary is the modality of choice.

```
zdim = ydim = xdim = 128; noise = Compile[{zdim, ydim, xdim},
  n = Table[Random[], {zdim}, {ydim}, {xdim}], {{n, _Real, 3}}];
follicle = Compile[{z0, y0, x0, r, zdim, ydim, xdim},
   f = Table[If[(x - x0)^2 + (y - y0)^2 + (z - z0)^2 < r^2, 0., 1.],
     {z, 1, zdim}, {y, 1, ydim}, {x, 1, xdim}],
   {{f, _Real, 3}, {x, _Real}, {y, _Real}, {z, _Real}}];
testset = follicle[60, 65, 50, 10, zdim, ydim, xdim] +
   follicle[35, 25, 55, 7, zdim, ydim, xdim] +
   follicle[35, 85, 95, 5, zdim, ydim, xdim] + noise[zdim, ydim, xdim];
DisplayTogetherArray[Table[ListDensityPlot[testset[[i]],
    PlotRange → {2, 4}, Frame → True, FrameTicks → False,
    FrameStyle → Red, PlotLabel -> "slice" <> ToString[i]],
   {i, 1, 90, 11}], ImageSize → 500];
```

Figure 13.9 Some slices from the artificial 3D ultrasound noisy testset with 3 follicles.

```
testsetblurred = gDn[testset, {0, 0, 0}, {3, 3, 3}];
```

We use the function **nMaxima** (defined in section 13.2) to find the n largest maxima in the testset. The minus sign is to find the minima.

```
nMaxima[im_, n_] := Module[{l, d = Depth[im] - 1},
  p = Times @@ Table[(Sign[im - Map[RotateLeft, im, {i}]] + 1)
      (Sign[im - Map[RotateRight, im, {i}]] + 1), {i, 0, d - 1}];
  l = Length[Position[p, 4^d]];
  Take[Reverse[Union[{Extract[im, #], #} & /@ Position[p, 4^d]]],
   If[n < l, n, l]]]
detected = nMaxima[-testsetblurred, 3]
```

```
{{-2.50496, {60, 65, 50}},
 {-2.66311, {35, 25, 55}}, {-2.93738, {35, 85, 95}}}
```

Indeed, the right minima positions are found. An alternative method (described in [TerHaarRomeny1999a]) is the use of 3D *winding numbers*. Winding numbers are explained in chapter 15).

We next check if they are surrounded by a sphere (an example of *model-based segmentation*), by tracking linear rays of 30 pixels long, starting in the detected minimum position, and over 7 polar (or colatitudinal, $0 < \theta < \pi$) and 5 azimuthal (or longitudinal, $0 < \phi < 2\pi$) angles. We sample equidistantly along these rays the 3D ultrasound intensities using cubic 3D polynomial interpolation (third order is the default interpolation order, linear interpolation is acquired by adding the option **InterpolationOrder→1**):

```
interpolation = ListInterpolation[testset];
θstep = π / 8; φstep = 2 π / 5;
rays[z_, y_, x_] := Module[{φ, θ, r}, Table[N[
     interpolation[z + r Cos[φ] Cos[θ], y + r Sin[φ] Cos[θ], x + r Sin[θ]]],
    {θ, -Pi / 2 + θstep, Pi / 2 - θstep, θstep},
    {φ, 0, 2 Pi - φstep, φstep}, {r, 1, 30}]];
```

The tracking rays are first visualized, to be sure we have the proper sampling in 3D. Lines of 30 pixels long are drawn in the 7 polar and the 5 azimuthal directions for each minimum found, the end is indicated with a blue dot:

```
star[z_, y_, x_] := Module[{φ, θ, r},
   Graphics3D[Table[{Wheat, Line[{{z, y, x}, {z + r Cos[φ] Cos[θ],
         y + r Sin[φ] Cos[θ], x + r Sin[θ]}}], Blue, AbsolutePointSize[3],
      Point[{z + r Cos[φ] Cos[θ], y + r Sin[φ] Cos[θ], x + r Sin[θ]}]},
     {θ, -Pi / 2 + θstep, Pi / 2 - θstep, θstep},
     {φ, 0, 2 Pi - φstep, φstep}, {r, 30, 30}], AspectRatio → 1]];
Show[Apply[star, Last /@ detected, 2],
  PlotRange → {{1, xdim}, {1, ydim}, {1, zdim}}, ImageSize -> 270];
```

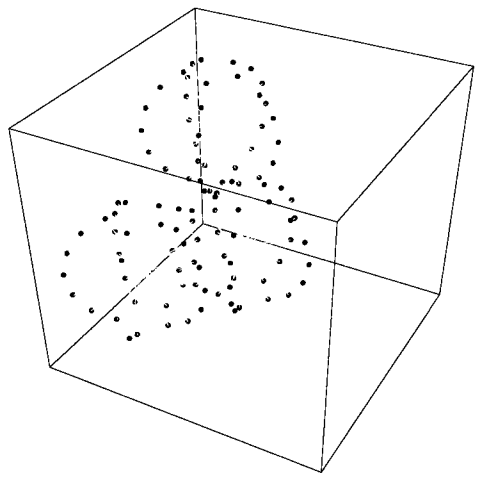
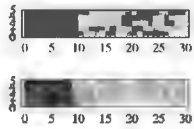


Figure 13.10 From each detected minimum rays are cast, along which the 3D US intensity is sampled. The blue dots mark the rays' endpoints.

Let us investigate the interpolated sampled intensity profiles along the radiating rays from the minimum point {60,65,50}:

```
profiles = rays[60, 65, 50];
ListDensityPlot[#, PlotRange → {2, 4},
    Frame → True, ImageSize -> 200] & /@ profiles;
```



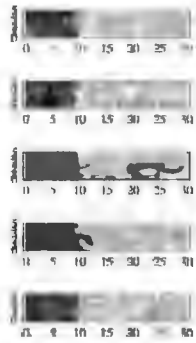


Figure 13.11 Sets of 5 3D ultrasound intensity profiles for each of the 7 polar directions. The origin, i.e. the start point of the ray in the detected minimum, is to the left. Ray length is 30 pixels.

For each ray the position of the follicle boundary is found by edge focusing (function defined in section 13.5.2) for the largest edge along the ray:

```
edgefocus[signature_, startlevel_, dir_] := Module[{a, b, c, out},
  out = 0. signature; a = Position[signature[[startlevel]], dir];
  Do[b = Position[signature[[i]], dir];
   c = Select[b, (Position[a, # - 1] ≠ {} ||
        Position[a, #] ≠ {} || Position[a, # + 1] ≠ {}) &];
   out[[i]] = ReplacePart[out[[i]], -1, c]; b = c; a = b,
   {i, startlevel - 1, 1, -1}]; out]

findedgelocation[track_] := Module[{scalespaceLxx},
   scalespaceLxx = Table[gDf1D[track, 2, Eτ], {τ, 0, 2, .06}];
signature = (RotateRight[#] - #) & /@ Sign[scalespaceLxx];
   Extract[Position[First[edgefocus[signature, 15, 2]], -1], {1, 1}]
  ];
```

This finds them all:

```
outr = Map[findedgelocation, profiles, {2}]

{{10, 11, 11, 10, 10}, {10, 10, 10, 10, 9},
 {10, 10, 11, 11, 10}, {10, 10, 10, 10, 10},
 {10, 10, 10, 10, 10}, {11, 10, 10, 9, 10}, {10, 10, 3, 11, 10}}
```

The proper 3D coordinates of the edge on the ray are found by converting the polar coordinates to the Cartesian coordinates. We put the center of the follicle in the position of the found minimum, and check the result visually.

```
Clear[f]; f[r_, {nθ_, nϕ_}] := N[{z + r Cos[nϕ ϕstep] Cos[-π / 2 + nθ θstep],
   y + r Sin[nϕ ϕstep] Cos[-π / 2 + nθ θstep], x + r Sin[-π / 2 + nθ θstep]}];
{z, y, x} = {60, 65, 50}; positions = MapIndexed[f, outr, {2}];
Show[Graphics3D[{Wheat, Map[Line[{{60, 65, 50}, #}] &,
     Flatten[positions, 1]], Red, AbsolutePointSize[5],
    Map[Point, positions, {2}]}], ImageSize → 215];
```

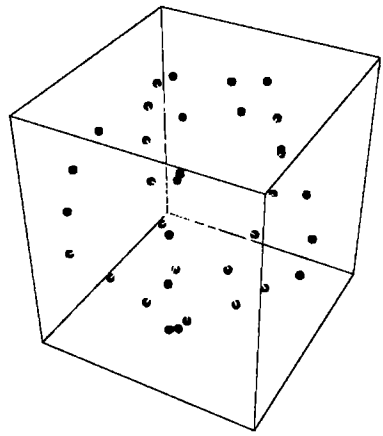


Figure 13.12 Follicle boundary points (red points) as detected by the edge focusing algorithms along each ray.

13.6.1 Fitting spherical harmonics to 3D points

A convenient representation for 3D point clouds on a detected surface form the *spherical harmonics*. These orthogonal polynomials $Y_l^m(\theta, \phi)$ are the angular portion of the solution in spherical coordinates to the Laplace's equation $\nabla^2 \Psi = \Delta \Psi = 0$ (see mathworld.wolfram.com/SphericalHarmonic.html).

The set of spherical harmonics up to second order is given by:

```
fitset =
 Flatten[Table[SphericalHarmonicY[l, m, θ, ϕ], {l, 0, 2}, {m, -l, l, 1}]]
```

$$\left\{\frac{1}{2\sqrt{\pi}},\ \frac{1}{2} e^{-i\phi}\sqrt{\frac{3}{2\pi}}\,\text{Sin}[\theta],\ \frac{1}{2}\sqrt{\frac{3}{\pi}}\,\text{Cos}[\theta],\right.$$

$$-\frac{1}{2} e^{i\phi}\sqrt{\frac{3}{2\pi}}\,\text{Sin}[\theta],\ \frac{1}{4} e^{-2i\phi}\sqrt{\frac{15}{2\pi}}\,\text{Sin}[\theta]^2,$$

$$\frac{1}{2} e^{-i\phi}\sqrt{\frac{15}{2\pi}}\,\text{Cos}[\theta]\,\text{Sin}[\theta],\ \frac{1}{4}\sqrt{\frac{5}{\pi}}\,(-1+3\,\text{Cos}[\theta]^2),$$

$$\left.-\frac{1}{2} e^{i\phi}\sqrt{\frac{15}{2\pi}}\,\text{Cos}[\theta]\,\text{Sin}[\theta],\ \frac{1}{4} e^{2i\phi}\sqrt{\frac{15}{2\pi}}\,\text{Sin}[\theta]^2\right\}$$

Mathematica's function Fit does a least square approximation with any set of functions. We plug in the points and the set of fit functions:

```
points = Flatten[positions, 1];
rofun[θ_, ϕ_] = Chop[Fit[points, fitset, {θ, ϕ}] // ExpToTrig, 10^-8];
rofun[θ, ϕ]
```

```
58.8381 + 1.96853 Cos[θ] - 12.0257 Cos[θ]^2 +
 4.31706 Cos[ϕ] Sin[θ] - 4.34484 Cos[θ] Cos[ϕ] Sin[θ] +
 6.409 Cos[2 ϕ] Sin[θ]^2 - 1.69899 Sin[θ] Sin[ϕ] +
 4.29894 Cos[θ] Sin[θ] Sin[ϕ] + 8.01166 Sin[θ]^2 Sin[2 ϕ]
```

This shows the detected follicle as a 3D volume surface:

```
ParametricPlot3D[rofun[θ, ϕ] {Cos[θ], Sin[θ] Cos[ϕ], Sin[θ] Sin[ϕ]},
  {θ, 0, π}, {ϕ, 0, 2 π}, AspectRatio → Automatic, ImageSize -> 150];
```

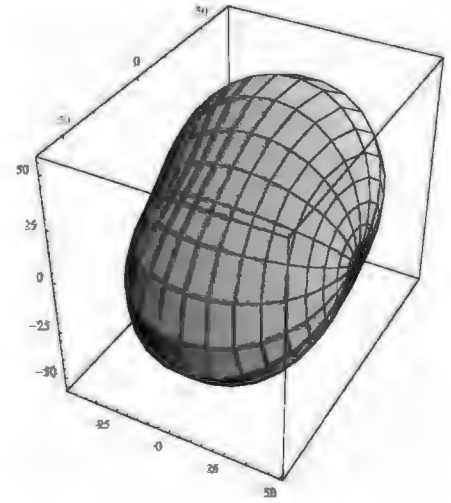


Figure 13.13 Spherical harmonic (second order) surface parametrization of one of the follicles in the noisy 3D US test dataset.

Because we have now an analytical expression for the volume, we can easily let *Mathematica* calculate the volume:

```
Clear[θ, ϕ]; volume = Integrate[rofun[θ, ϕ], {θ, 0, π}, {ϕ, 0, 2 π}]
```

```
1042.73
```

▲ Task 13.3 Fit the spherical harmonics to 4th order.

▲ Task 13.4 A high order of spherical harmonics functions as fit functions gives us not the correct result, due to overfitting. What do we mean by this?

13.7 Multi-scale segmentation

The trade-off between simplification and detail is classical. Gaussian blurring allow the significant features to emerge from the noise - the price is general dislocation and blurring of the objects of interest. As the previous sections indicate there is not need for such a black and white perception of blurring. The features can be detected at the appropriate scale where they are best distinguished from their surrounding - and then linking down through scale allow a conceptual deblurring that allow the fine scale shape and features to be inspected.

This section presents a segmentation method that takes advantage of the deep structure in order to facilitate this simplification followed by extraction of the fine scale shape. The method investigates the deep structure of watershed regions. The result is a partitioning of the image at all scales simultaneously. The regions from this multi-scale partitioning can then be used as building blocks in an interactive segmentation application.

The multi-scale watershed segmentation method presented here is due to Ole Fogh Olsen [Olsen1997]. A similar approach has been presented in [Gauch1999]. In the following, the method is presented step by step - at the end the whole method is collected into a single *Mathematica* function.

13.7.1 Dissimilarity measure in scale-space

The borders between the regions should be located where there is a large contrast in the image. The measure of the contrast - or the *dissimilarity measure* - can be defined according to the specific application. For gray-scale images, a natural and simple definition of the dissimilarity measure is the gradient magnitude squared. Below an example image and the corresponding dissimilarity image is displayed at a few scales.

```
dissimilarity[im_, σ_] := gD[im, 1, 0, σ]^2 + gD[im, 0, 1, σ]^2;
noisyShapes =
  Import["blobs.gif"][[1, 1]] + Table[Random[], {128}, {128}];
disScaleSpace = Table[dissimilarity[noisyShapes, E^τ], {τ, 1, 2.2, .3}];

Show[GraphicsArray[Flatten[{
     ListDensityPlot[noisyShapes, DisplayFunction -> Identity],
     (ListDensityPlot[disScaleSpace[[#]], DisplayFunction ->
          Identity]) & /@ {1, 2, 3, 4}}]], ImageSize -> 400];
```

Figure 13.14 Left: The 128 by 128 test image displaying some simple shapes with S/M ratio 1. Right: Gradient squared dissimilarity images for four scale levels: σ = 1.6, 2.6, 4.1, and 6.6 pixels.

13.7.2 Watershed segmentation

Imagine rain pouring down a hilly landscape. After hitting the ground, the individual drops run downhill and gather in pools. Every time a drop hits a certain spot the drop will run into the same pool. This implicitly partitions the landscape into regions of support for each pool.

A part of the landscape that leads water to a specific pool belongs to the *catchment basin* for this pool. The borders between the catchment basins are the *watersheds*. These geographic concepts was introduced in mathematics in [Cayley1859, Maxwell1870]. Aside: According to Encyclopedia Brittanica, the term *watershed* is actually a drainage basin. Furthermore, "the term has also been used synonymously with drainage divide, but this use is discouraged". However, the computer vision community is traditionally not discouraged.

We use to this principle to partition the dissimilarity image into regions. In order to calculate the catchment basins we first determine the direction of steepest descent for each pixel in a dissimilarity image. This is done by checking the difference with each of the four neighbors.

The function operate on images that have been `Flatten`'ed. Thereby the neighbors are located at offsets *1*, *-1*, *xDim*, and *-xDim*.

```
checkDirection[disIm_, bestDirection_, offset_] := Module[
   {disNeighbor, checkNeighbor,
    neighborDif, bestDif, bestOffset, disVal, disNeighborVal},
   disNeighbor = RotateLeft[disIm, offset];
   checkNeighbor[disVal_, disNeighborVal_, {bestOffset_, bestDif_}] := (
     neighborDif = disNeighborVal - disVal;
     If[neighborDif < bestDif, {offset, neighborDif}, {bestOffset, bestDif}]);
   Thread[checkNeighbor[disIm, disNeighbor, bestDirection]]];

bestDirection[disIm_, xDim_, yDim_] := Module[
   {bestDir},
   bestDir = Table[{0, 0}, {xDim yDim}];
   bestDir = checkDirection[disIm, bestDir, 1];
   bestDir = checkDirection[disIm, bestDir, -1];
   bestDir = checkDirection[disIm, bestDir, xDim];
   bestDir = checkDirection[disIm, bestDir, -xDim];
   bestDir];
```

From a given pixel, the path given by the local direction of steepest descent are to be followed until a local minimum is reached. In order to prepare for this we assign a unique label to each local minimum. The local minima are the points with no direction of steepest descent. The result is a "label" image with numbers at local minima and zeros elsewhere.

```
labelMinima[bestDir_, xyDim_] := Module[
   {labels, minima, basinLabels},
   labels = Table[0, {xyDim}];
   minima = Flatten[Position[bestDir, {0, 0}]];
   basinLabels = Range[1, Length[minima]];
   labels[[minima]] = basinLabels;
   labels];
```

The remaining pixels are then to be labelled with a region number. From a given pixel the path defined by the directions of steepest descent are followed until a local minimum is

reached. The descent path is kept in a stack. When a minimum is reached every pixel in the path stack can be labelled with the region number of the minimum (stop reading until this is trivial). Furthermore, it is not necessary to continue the path to a minimum if a pixel that has already been labelled is encountered on the way. The label of this pixel is equivalent to the label of the minimum that would eventually be reached.

```
descentToMinima[bestDir_, minLabels_, xDim_, yDim_] := Module[
   {xyDim, labels, j, stack, stackCount, pixel, basinLabel, i},
   xyDim = xDim yDim; labels = minLabels;
   For[j = 1, j <= xyDim, j++, If[labels[[j]] == 0,
     stack = Table[0, {2 Max[xDim, yDim]}]; stackCount = 1; pixel = j;
     While[labels[[pixel]] == 0, stack[[stackCount]] = pixel;
      pixel += bestDir[[pixel, 1]]; If[pixel < 1, pixel += xyDim];
      If[pixel > xyDim, pixel -= xyDim];
      stackCount++]; basinLabel = labels[[pixel]];
     For[i = 1, i < stackCount,
      i++, labels[[stack[[i]]]] = basinLabel]]];
   labels];
```

The catchment basins can now be constructed through the use of the functions above in the following manner. Since the images are **Flatten**'ed in the calculations, the final image with basin labels is **Partition**'ed.

```
makeBasins[disIm_] := Module[
   {xDim, yDim, bestDir, labels},
   {xDim, yDim} = Dimensions[disIm];
   bestDir = bestDirection[Flatten[disIm], xDim, yDim];
   labels = labelMinima[bestDir, xDim yDim];
   labels = descentToMinima[bestDir, labels, xDim, yDim];
   Partition[labels, xDim]];
```

The borders between the regions are more appropriate for illustrations than the region labels. The function below finds the pixels where the neighboring labels are different and mark these as borders:

```
makeBorders[labelImage_] := Module[
   {left, down, border},
   left = labelImage - RotateLeft[labelImage];
   down = labelImage - Map[RotateLeft, labelImage];
   border[l_, d_] := If[l == 0 && d == 0, 0, 1];
   SetAttributes[border, Listable];
   border[left, down]];
```

The basic watershed segmentation functions are illustrated with the dissimilarity scale-space of the example image.

```
basinScaleSpace = Table[makeBasins[disScaleSpace[[i]]], {i, 1, 4}];
Show[GraphicsArray[Flatten[{
     ListDensityPlot[noisyShapes, DisplayFunction -> Identity],
     (ListDensityPlot[makeBorders[basinScaleSpace[[#]]],
         DisplayFunction -> Identity]) & /@
      {1, 2, 3, 4}}]], ImageSize -> 400];
```

Figure 13.15 Left: The 128 by 128 test image. Right: Watershed catchment basins at four scale levels. Each object from the original image can be captured as a single region from the catchment basins. However, the blurring affects the shapes of the objects. Compare with figure 13.17.

We can see from the example above, that for each object in the original image, there is indeed a corresponding catchment basin at some scale level. However, the blurring has caused the captured shapes to be rounded as the edges are dislocated. In order to avoid this effect we must "deblur" the shapes by linking down through scale.

13.7.3 Linking of regions

As scale is increased, the number of catchment basins decrease - the catchment basins gradually merge into larger basins. Each catchment basin corresponds to a local minimum for the dissimilarity measure. These minima form singularity strings like the ones showed in the previous section on edge focusing. Therefore we could track these minima by a linking process similar to the focusing in that section. As we saw in that section, this linking of singularity points through scale-space is non-trivial. Therefore, we will here pursue another approach where linking is based on regions instead of points.

The conceptually simple method is to link a region at a given scale to the region at the next scale with the maximal spatial overlap.
As the blurring increases, the borders of the regions move slightly. However, the central part of the region remain within the same area. The linking scheme is illustrated in figure 13.16.

The merging of the catchment basins defines a hierarchy of regions. Each region at a given scale is linked to exactly one region at the next, higher scale level.

The linking is implemented by the function below. The parameters are the labelled basins for two adjacent scale levels. The labels must be numbered consecutively from 1.

The function is somewhat complicated. First the two basin images are combined into a single list `sortedOverlap` where each element is a number that signifies a "vote" - a fine scale region x has one pixel that overlaps with a coarse scale region y. The number is constructed such that all votes from one fine scale region are consecutive when the list is sorted. From these votes the the coarse scale region with most votes is extracted for each fine scale region.

```
Show[GraphicsArray[{Import["simple_link.jpg"],
    Import["merge_link.jpg"]}], ImageSize -> 400];
```

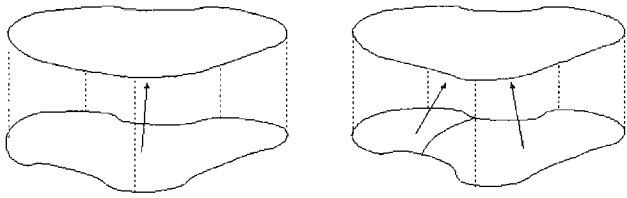


Figure 13.16 Linking of regions across scale. As scale is increased, the regions become more rounded and the borders move slightly. Left: The shape of the region simplifies but the main part of the area overlaps. Right: Two regions merge into one. Both regions link to the combined region since this is the region with maximal overlap.

The extraction is done through a linear pass of the **sortedOverlap** list.

The number of votes for the current coarse scale region from the current fine scale region is counted during the pass. Whenever there is a change in coarse scale, it is recorded whether the previous region got more votes than the one with most votes so far.

When there is a change is fine scale, the coarse scale region with most votes is the desired region to be linked to - this is recorded in **linkMap**. The pass over **sortedOverlap** is done with a **Fold** using the function **nextLinkVote** for each element in the list. The function returns a list of labels. For each fine scale label number, the list contains the coarse scale label with maximal overlap.

The linking functions (next page) are very "imperative" in programming style and not very *Mathematica*. Indeed, it can be written much more elegantly. The function below has the exact same functionality and is much shorter. However, there is one caveat. The function below has runtime proportional to the number of regions - the functions above have runtimes proportional to the size of the image.

This effectively means that the short version below is faster for linking at large scales where there a few regions. Unfortunately, it is very much slower at low scales where there are many regions. This is an example of how sometimes, in *Mathematica*, short and elegant is not always preferable.

The linking process result in a mapping between the region labels at low scale and region labels at the next scale. This linking tree can be used for the "region focusing" process. A given region at high scale can be substituted by the regions at a lower scale that link to it. This is done recursively through the scale-space of catchment basins.

```
linkBasins[fineBasins_, coarseBasins_] := Module[
   {maxFine, maxCoarse, overlapIdx, sortedOverlap,
    linkMap, LastLimit, bestCount, bestCoarse, curCount, curCoarse},
   maxFine   = Max[fineBasins]; maxCoarse = Max[coarseBasins];
   overlapIdx = (fineBasins - 1) maxCoarse + coarseBasins;
   sortedOverlap = Sort[Flatten[overlapIdx]];
   linkMap = Table[0, {maxFine}];
   next[{fineLimit_, bestCount_, bestCoarse_, curCount_, curCoarse_}, idx_] :=
    Module[
     {coarse, fine}, coarse = Mod[idx - 1, maxCoarse] + 1;
     If[idx > fineLimit,
      (* The series for one fine scale region is done - register previous *)
      fine = Quotient[fineLimit, maxCoarse];
      If[curCount > bestCount,
       linkMap[[fine]] = curCoarse, linkMap[[fine]] = bestCoarse];
      {fineLimit + maxCoarse, 0, 0, 1, coarse},
      (* Else:
        Next coarse scale region for same fine scale region encountered *)
      If[coarse == curCoarse,
       (* Coarse scale is the same so increment count *)
       {fineLimit, bestCount, bestCoarse, curCount + 1, coarse},
       (* Coarse scale new so register previous and start count from 1 *)
       If[curCount >= bestCount,
        {fineLimit, curCount, curCoarse, 1, coarse},
        {fineLimit, bestCount, bestCoarse, 1, coarse}]]]];
   {lastLimit, bestCount, bestCoarse, curCount, curCoarse} =
    Fold[next, {maxCoarse, 0, 0, 0, 0}, sortedOverlap];
   (* The last coarse scale region bas not been registered -
     check if it is best *)
   If[curCount > bestCount, linkMap[[maxFine]] = curCoarse,
    linkMap[[maxFine]] = bestCoarse]; linkMap];

linkBasinsShort[fineBasins_, coarseBasins_] := Module[
  {linkVotes, uniquePairs, PairCount,
   bestLink, possibleLinks, possibleCounts, bestPosition},
  linkVotes = Thread[({#1, #2} &)[Flatten[fineBasins], Flatten[coarseBasins]]];
  uniquePairs = Union[linkVotes];
  PairCount = Map[Count[linkVotes, #] &, uniquePairs];
  bestLink[fine_] := (
    possibleLinks = Position[uniquePairs, _?(#[[1]] == fine &), 1, Heads → False];
    possibleCounts = Extract[PairCount, possibleLinks];
    bestPosition = First[Position[possibleCounts, Max[possibleCounts]]];
    uniquePairs[[Extract[possibleLinks, bestPosition]]][[1, 2]]);
  Map[bestLink, Table[i, {i, Max[fineBasins]}]]]
```

Below, we generate a scale-space of region labels, where the catchment basins are linked down to the lowest scale level - the *localization scale*. This done from fine to coarse scale, where the linking map provided by the previous function is used to map the catchment basins into localized basins.

```
localizedScaleSpace = Table[0 image, {i, 1, 4}];
localizedScaleSpace[[1]] = basinScaleSpace[[1]];
For[level = 2, level <= 4, level++,
 linkMap =
  linkBasins[basinScaleSpace[[level - 1]], basinScaleSpace[[level]]];
 localizedScaleSpace[[level]] =
  linkMap[[#]] & /@ localizedScaleSpace[[level - 1]]]
Show[GraphicsArray[Flatten[{
     ListDensityPlot[noisyShapes, DisplayFunction -> Identity],
     (ListDensityPlot[makeBorders[localizedScaleSpace[[#]]],
         DisplayFunction -> Identity]) & /@
      {1, 2, 3, 4}}]], ImageSize -> 500];
```

Figure 13.17 Left: The 128x128 test image. Right: The localized gradient watershed regions (to be compared with figure 13.15). The middle three images show the intermediate results. The larger structures are segmented well. In particular the rectangle has been segmented with sharp corners.

The regions at the highest scale now clearly correspond to the objects in the original image. Notice how regions are merged into larger regions as scale increases - but the location of the remaining borders remain fixed due to the linking.
The obvious improvement from figure 13.15 to figure 13.17 is due to the linking - or, in other words, the deep structure.

13.7.4 The multi-scale watershed segmentation

The bits and pieces that produce the localized catchment basins can be put together in a module. The complete function takes an image and the desired list of scales.

The scale levels are a parameter since the dimensions of the image, the noise level, and the size of the objects of interest all affect the appropriate localization scale, number of scale level and scale sampling spacing. The function is named **`generateBuildingBlocks`** in order to emphasize the nature of the output.

The method does not provide a complete segmentation, where the image has been partitioned into background and a number of objects.

Instead the method provides building blocks of varying sizes that can be used for an interactive segmentation process. The user then selects the appropriate regions at the proper scales that allow construction of the desired objects.

```
generateBuildingBlocks[image_, scales_] := Module[
   {borderScaleSpace, fineBasins,
    coarseBasins, localizedBasins, linkMap},
   borderScaleSpace = Table[0 image, {Length[scales]}];
   coarseBasins = makeBasins[dissimilarity[image, scales[[1]]]];
   localizedBasins = coarseBasins;
   borderScaleSpace[[1]] = makeBorders[localizedBasins];
   For[level = 2, level <= Length[scales], level++,
    fineBasins = coarseBasins;
    dissimImage  = dissimilarity[image, scales[[level]]];
    coarseBasins = makeBasins[dissimImage];
    linkMap = linkBasins[fineBasins, coarseBasins];
    localizedBasins = Map[Function[linkMap[[#]]], localizedBasins];
    borderScaleSpace[[level]] = makeBorders[localizedBasins]];
   borderScaleSpace];
```

The test image resulting in figure 13.17 is certainly not the most difficult segmentation task The objects are quite easy to distinguish from the background. The example below with an MR brain scan is more realistic - and more challenging.

```
scales = Table[1.4^t, {t, 1, 8}]; mr = Import["mr128.gif"][[1, 1]];
borderScaleSpace = generateBuildingBlocks[mr, scales];

Show[ GraphicsArray[
   {Flatten[{ListDensityPlot[mr , DisplayFunction -> Identity],
     ListDensityPlot[borderScaleSpace[[#]] , DisplayFunction →
          Identity] & /@ {2, 4, 6, 8}}]}], ImageSize -> 500];
```

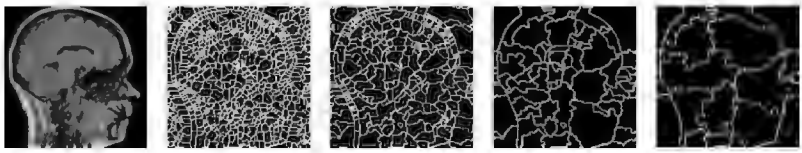


Figure 13.18 Left: An 128x128 MR brain scan. Right: The localized gradient watershed regions for selected scales from a scale-space with 8 levels from $\sigma = 1.4$ to $\sigma = 14.8$ pixels. At low scale the finer details can be selected. At coarse scale the larger anatomical structures, such as the brain, are available.

As the example illustrates, a number of building blocks must be selected interactively in order to form most anatomical objects in the figure. It is not really surprising that the method does not provide perfect partitioning of the brain anatomy. The multi-scale watershed segmentation method is completely un-committed. Therefore it can not be expected to perform perfectly for a highly specialized task.

The strength of the method is the generality. The method itself is un-committed towards any special task and can be used for n-dimensional data.

The task specific knowledge is provided by a user in an interactive program. This approach will generally not be competitive with highly specialized approaches for specific segmentation task. However, for many segmentation task where no specialized methods

exist, the multi-scale watershed segmentation method is relevant. The method have been implemented with promising results for clinical use in medical imaging [Dam2000c].

The segmentation method can be optimized in a number of ways (see [Olsen1997] for details). A possibly way of specializing the method towards specific tasks is to design suitable dissimilarity measures. Another approach is to specialize the underlying diffusion scheme - this is briefly discussed in the next section.

13.8 Deep structure and nonlinear diffusion

Linear Gaussian diffusion has a number of appealing theoretical properties. However, as we will see in chapter 21, non-linear diffusion is superior for several specific applications. Among these are edge detection [Perona1990], edge enhancement, and fingerprint enhancement [Weickert1998a]. Generally, linear Gaussian diffusion is inferior in applications where elongated structures are to be preserved during the diffusion.

The deep structure is defined by the diffusion. Non-linear diffusion allows adaptation of the local diffusion to the local geometry of the image. Thereby the deep structure can be specialized towards specific applications. It is therefore obvious to investigate whether some of the non-linear diffusion schemes allow superior performance compared to linear diffusion in deep structure applications.

This section will not give a comprehensive treatment of the effect of the diffusion scheme on the deep structure. However, we do provide the following appetizer.

13.8.1 Non-linear diffusion for multi-scale watershed segmentation

Diffusion blurs across the borders of the catchment basins and ensures that the watersheds are blurred away. The catchment basins merge into building blocks of increasing size as scale increases. However, the linear diffusion scheme treats all image regions with the same amount of blurring. This causes roundish shapes to be favoured as building blocks.

Non-linear diffusion schemes allow specification of which image features to "protect" during the diffusion. For instance, the classical Perona-Malik diffusion scheme allows specification of an edge threshold defined in terms of the gradient (see chapter 21). In areas where the gradient is above this threshold, the blurring is diminished in order to preserve the edge. For edge detection this allows simplification of the image while the desired edges are preserved.

In the multi-scale watershed segmentation method this is also interesting. The goal is to provide the user with building blocks that capture the desired image objects in just a few building blocks. For elongated objects this is problematic with linear diffusion. At low scale, elongated objects will be split into a number of short pieces. In order for these pieces to merge together into a single region a scale proportional with the length of the object is needed. However, at such a large scale, the blurring across the object is substantial. The borders of the elongated structures are likely to be blurring away due to influence from the surrounding structures.

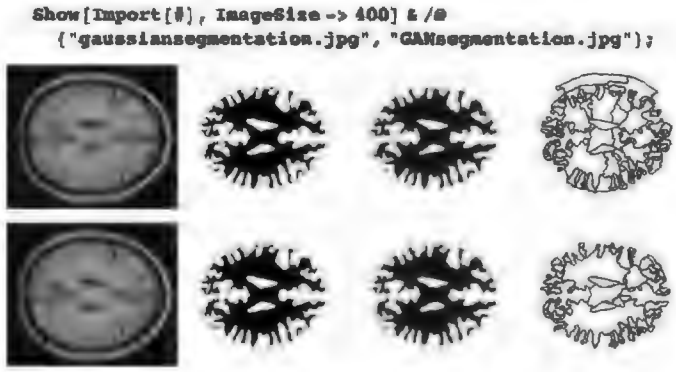


Figure 13.19 The effect of applying non-linear diffusion for multi-scale watershed segmentation. Top and bottom rows are the effects of linear and a non-linear scheme called GAN, respectively. A brain MR scan is segmented using the multi-scale watershed segmentation method. Left images: originals. The next images show the ground truth segmentation of the white matter tissue. The colored images are the status of the segmentation. Blue areas are correctly segmented, green areas are within the ground truth but not segmented, and red areas are segmented but not part of the ground truth. The images to the right illustrate the watershed segmentation building blocks that have been used for the segmentation. On average, with the GAN scheme in the segmentation method, the user is required to supply less than half the number of user interactions compared to linear Gaussian diffusion. Visually, the building blocks resulting from GAN diffusion correspond much better with the shape of the ground truth.
GAN (Generalized Anisotropic Non-linear diffusion) is a scheme that has several important diffusion schemes as special cases. Here, the parameters make the scheme similar to the Perona-Malik scheme. The illustration is from [Damon1995, Damon1997].

Non-linear diffusion minimizes the diffusion across the edges: edges of elongated structures can survive to higher scales and thereby enable merging of regions inside the objects. For the multi-scale watershed segmentation the performance has been evaluated for a number of diffusion schemes (among these *mean curvature motion* and Weickert's *anisotropic nonlinear diffusion* schemes). The results reveal that, compared to linear diffusion, a scheme similar to the Perona-Malik scheme allow the desired objects to be captured using less than half as many actions [Dam2000b]. This is illustrated in figure 13.19. Similar results have been established for the Hyperstack segmentation method [Niessen1997d].

14. Deep structure II. catastrophe theory

Erik Dam and Bart M. ter Haar Romeny

14.1 Catastrophes and singularities

The previous chapter illustrates a number of approaches that explore the deep structure. However, there are a number of caveats. The edge focusing technique implicitly assumes that the edges for the signal can be located at the adjacent lower scale level in a small neighborhood around the location at the current scale. As mentioned, no formal scheme for defining the size and shape of the neighborhood is presented. Furthermore, this method ignores the problems encountered when edge points merge or split with increasing scale.
Analogously, the multi-scale watershed segmentation depends on the behaviour of the dissimilarity measure singularities (the notion of dissimilarity is defined in chapter 13, section 6.1). Even though the linking of the watershed catchment basins is quite robust due to the matching of regions (opposed to tracking of points as in the edge focusing paradigm), the linking in the presence of merges and splits of regions is not explicitly established above. Without knowledge of how the dissimilarity singularities can behave in scale-space, we can only hope that the method will work on other images than the ones used for the illustrations.
The finding of explicit schemes for linking in the neighborhood of changes in the singularity structures requires explicit knowledge of the changes. A change in the singularity structure in denoted a *catastrophe*. Catastrophe theory (or with a broader term: singularity theory) is the theory that analyses and describes these changes. Catastrophe theory allows prediction of which changes in the singularity structure can be expected. Thereby the schemes that involve the singularities can be designed to detect and handle these events as special cases.

Actually, this analysis was done for the multi-scale watershed segmentation method in [Olsen1997].

The field of catastrophe theory is vast and rather complicated. The focus of this introduction is to give a condensed introduction of the central definitions, with an intuitive understanding of the effects we often observe in 'deep scale-space'.

14.2 Evolution of image singularities in scale-space

An important property of linear scale-space is the overall simplifying effect of increasing scale.

In general, this implies that the number of appearances of a given image feature decreases as scale increases. In particular, this is the qualitative behavior for the image singularities - blurred versions of an image will in general contain less singularities than the original one.

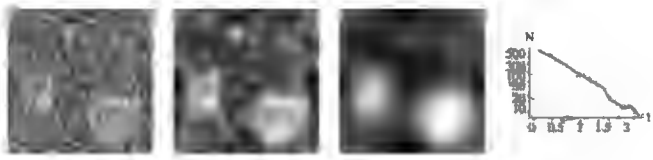
As mentioned in chapter 2 section 2.8, this notion is formalized by Florack [Florack1992a], which leads to a prediction on the number of singularities in n-dimensional signals/images. Specifically, the number of singularities can be expected to decrease with a slope of -1 for 1D signals and -2 for 2D images (in general: $-n$ for n-D signals; see chapter 1 for the reasoning to derive this).

More precisely, when the scale levels are generated with the usual exponential sampling $\sigma = \epsilon\, e^{\tau}$ the logarithm of the number of singularities decrease with these slopes as a function of the scale parameter τ.

This is illustrated for the "blobs" image from the scale selection section in the previous chapter. For simplicity we count the number of maxima instead of the number of singularities (then we can use the function **nMaxima**). Following the argument of Florack, the relative decrease in the number of maxima is equivalent to the decrease in the number of singularities.

```
<< FrontEndVision`FEV`;
countMaxima[im_] := Module[{p, d = Depth[im] - 1},
   p = Times @@ Table[ (Sign[im - Map[RotateLeft, im, {i}]] + 1)
        (Sign[im - Map[RotateRight, im, {i}]] + 1), {i, 0, d - 1}] / 4^d;
   Count[Flatten[p], 1]];
noisyblobs =
  Import["blobs.gif"][[1, 1]] + 10 Table[Random[], {128}, {128}];

levels = 15; step = 0.15;
data = Table[{step t, countMaxima[nb[t] = gDf[noisyblobs, 0, 0, E^(step t)]]},
  {t, levels}]; DisplayTogetherArray[
 Append[ListDensityPlot /@ {nb[1], nb[8], nb[15]}, LogListPlot[data,
   PlotJoined -> True, AxesLabel -> {"t", "N"}]], ImageSize -> 400];
Print["Slope = ", Coefficient[Fit[Log[data], {1, t}, t], t]];
```




```
Slope = -1.79756
```

Figure 14.1 The evolution of the number of singularities for a set of noisy 2D blobs. Images blurred with $\sigma = e^{0.15} = 1.16$, $\sigma = 3.32$ and $\sigma = 9.49$ pixels. The observed slope is close to the predicted value of -2.

The blob image from figure 13.3 is used with noise added (S/N ratio = 1/10). First we define the scale levels used (such that $\sigma = \epsilon\, e^{\text{step}\,\tau}$ where the **step** ensures sufficiently small scale

steps, and $\epsilon = 1$). We display the lowest, middle and highest levels of the selected range of 15 scales and plot the logarithm of the number of maxima as a function of **`step t`**. The slope is calculated with a linear least square **`Fit`**.
The overall effect of blurring signals and images is simplification. This is examplified by the decrease of maxima above. However, this general notion reveals nothing about *how* the singularities disappear. More specifically, it gives no insight into the local structure of the signals and images at the specific point in scale-space where a singularity is annihilated. Furthermore, we get no information about whether singularities are *created* as well. In order to investigate these matters we need a bit of mathematics - introduced in the following through some central concepts from *Catastrophe Theory*.

▲ Task 14.1. Check the expected decrease in the number of singularities for a 1D signal as it was done for a 2D image above. Use the 1D noisystep signal.

14.3 Catastrophe theory basics

The field of catastrophe theory is quite extensive and complicated. This introduction focuses on giving an intuitive understanding of the concepts most related to computer vision and image processing. Therefore, the presentation is also somewhat less strict than possible. For a comprehensive introduction see [Gilmore1981].

The singularities are central feature points of an image - or more general for a function. The singularities alone offer a good qualitative description of the structure of a function. When a function undergoes an evolution it is therefore central to capture where the set of singularities change, in order to analyze the evolution. These points are denoted *catastrophes* since this is where the qualitative structure changes.
In order to describe these events properly, a few definitions are needed. They are presented quite briefly - the concepts are then illustrated by a number of examples.

14.3.1 Functions

For a smooth ($\equiv$ infinitely differentiable, indicated with C^∞) function f the parameters are divided into n *state* and m *control* parameters: $f(x_1, .., x_n, c_1, .., c_m) \in C^\infty(\mathbb{R}^{n+m}, \mathbb{R})$. For the intuitive understanding, think for the concept of scale-space of the *state* parameters as *spatial* coordinates, and think of a single *control* parameter, namely *scale*.

14.3.2 Characterization of points

For a smooth function $f(x_1, .., x_n, c_1, .., c_m) \in C^\infty(\mathbb{R}^{n+m}, \mathbb{R})$ a given point $p \in \mathbb{R}^{n+m}$ is either:

$$\begin{aligned}
\texttt{Regular:} \quad & \exists\, l \in [1..n] \text{ such that } \frac{\partial f}{\partial x_l} \neq 0 \\
\texttt{Morse Singularity:} \quad & \frac{\partial f}{\partial x_i} = 0 \quad \text{and} \quad \texttt{Det}\left(\frac{\partial f}{\partial x_i\, x_j}\right) \neq 0 \\
\texttt{Catastrophe:} \quad & \frac{\partial f}{\partial x_i} = 0 \quad \text{and} \quad \texttt{Det}\left(\frac{\partial f}{\partial x_i\, x_j}\right) = 0
\end{aligned}$$

In the equations above, tensor notation is used for the spatial parameters with subscripts i and j (but not for l).

Note that a *regular* point is only required to be a singularity with respect to the state (or spatial) parameters. Singularities and catastrophes are found at locations where the gradient is zero, so at horizontal locations in the image intensity landscape. A catastrophe differs from a singularity in that the second order structure has a degenerate Hessian matrix, i.e. the determinant of the Hessian matrix vanishes, and the Hessian matrix is thus singular in catastrophes.
Morse singularities are non-degenerate singularities in the state (or spatial) parameters. When the determinant of the Hessian matrix for the state parameters is vanishing the singularity becomes degenerate.
A non-degenerate singularity is *stable*. This means that a slight pertubation of the function will not change the local qualitative structure of the function - there will still be a singularity of the same type near the original with a value close to the original.
Degenerate singularities are *not stable* - a slight pertubation can cause a change in the local structure of the function. Such a pertubation could be caused by a slight change in the *control* parameters. Specifically, in scale-space the singularity structure changes at degenerate singularities when the scale is changed. These are the *catastrophe points* where *creations* or *annihilations* of singularities occur.

14.3.3 Structural equivalence

Two functions are *locally structurally equivalent* at a point if a diffeomorphism (a smooth invertible function with smooth inverse) exists such that a change of the coordinate system for one function with this diffeomorphism will make the functions equal in a neighborhood around the point.
Two functions are *globally structurally equivalent* if they are *locally structurally equivalent* at all points.

We will not formalize this definition in mathematical notation. The key point is that these definitions imply that two functions are structurally equivalent if their singularity structures are corresponding - the topological ordering and the types of the singularities are equivalent.

Slight pertubations of a function will in general leave it structurally equivalent with itself. The singularities move a bit and change value, but the topological structure remains the same. However, in the presence of catastrophe points, a slight pertubation will change the singularity structure and the function no longer remains structurally equivalent to itself.

14.3.4 Local characterization of functions

Analogous to the characterization of points into three classes, the local structure of a function is characterized by the following theorems. Here, f is a given smooth function $f(x_1, .., x_n, c_1, .., c_m) \in C^\infty(\mathbb{R}^{n+m}, \mathbb{R})$.

Implicit Function Theorem:
At a given regular point the function f is locally structurally equivalent with the function g where

$$g(x_1, .., x_n, c_1, .., c_m) = x_i$$

In other words, the Implicit Function Theorem states that at a regular point the function is locally equivalent with its tangent plane.

The Morse Lemma:
At a given Morse singularity point the function f is locally structurally equivalent with the function g where

$$g(x_1, .., x_n, c_1, .., c_m) = \frac{1}{2!} x_i x_j$$

The Morse Lemma states that at Morse singularity points the local structure is defined by the second order terms.

The Splitting Lemma:
At a given catastrophe point for the function f, the Eigenvalues for the Hessian matrix can be ordered by absolute value with the first d being zero - corresponding to the degree of degeneracy. The function f is then locally structurally equivalent with the function g where

$$g(x_1, .., x_n, c_1, .., c_m) = g_{n\,m}(x_1, .., x_d) + \sum_{i=d+1}^{n} \sum_{j=d+1}^{n} \frac{1}{2!} x_i x_j$$

The *Splitting Lemma* states that the function can be split into two parts: A non-Morse part and a Morse part. Accordingly, the parameters are split into "bad" and "good" parameters. The bad parameters correspond to the degenerate directions of the Hessian matrix. However, the *Splitting Lemma* does not characterize the local structure of the non-Morse part of the function. This is done by a theorem by the French mathematician René Thom (1923-).

14.3.5 Thom's theorem

Let f be a given smooth function $f(x_1, .., x_n, c_1, .., c_m) \in C^{\omega}(\mathbb{R}^{n+m}, \mathbb{R})$. *At a catastrophe point the eigenvalues for the Hessian matrix can be ordered by absolute value with the first d being zero. The function is then locally structurally equivalent with a function g where*

$$g(x_1, .., x_n, c_1, .., c_m) = CatGerm(d) + Perturb(d, m) + \alpha\, x_i\, x_j$$

If m≤5 then CatGerm(d) is one of the Catastrophe Germs and Pert(d,m) is the corresponding Pertubation listed in the table below:

Name	Nickname	m	d	CatGerm (d)	Perturb (d, m)
A_2	Fold	1	1	x^3	$c_1 x$
$A_{\pm 3}$	Cusp	2	1	$\pm x^4$	$c_1 x + c_2 x^2$
A_4	Swallowtail	3	1	x^5	$c_1 x + c_2 x^2 + c_3 x^3$
$A_{\pm 5}$	Butterfly	4	1	$\pm x^6$	$c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4$
A_6	□	5	1	x^7	$c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5$
D_{-4}	Elliptic Umbilic	3	2	$x^2 y - y^3$	$c_1 x + c_1 y + c_3 y^2$
D_{+4}	Hyperbolic Umbilic	3	2	$x^2 y + y^3$	$c_1 x + c_1 y + c_3 y^2$
D_5	Parabolic Umbilic	4	2	$x^2 y + y^4$	$c_1 x + c_1 y + c_3 x^2 + c_4 y^2$
D_{-6}	□	5	2	$x^2 y - y^5$	$c_1 x + c_2 y + c_3 x^2 + c_4 y^2 + c_5 y^3$
D_{+6}	□	5	2	$x^2 y + y^5$	$c_1 x + c_2 y + c_3 x^2 + c_4 y^2 + c_5 y^3$
$E_{\pm 6}$	□	5	2	$x^3 \pm y^4$	$c_1 x + c_2 y + c_3 xy + c_4 y^2 + c_5 xy^2$

Figure 14.2 Table of elementary catastrophes for $m \leq 5$. The names in the first column were originally proposed by Thom [Thom1975]. The nicknames come from their visual appearance (see MathWorld [http://mathworld.wolfram.com/Catastrophe.html] with interactive plots, Gray [Gray1993], Bruce & Giblin [Bruce1984] and Scanns [Scanns2000]). The c_1 to c_5 factors are also called the control factors.

The *Catastrophe Germ* is the local structure of the function for the specific set of control parameters at the catastrophe point.

The *Perturbation* terms determine how the function behaves when the control parameters vary (in a neighborhood around the catastrophe point).

14.3.6 Generic property

A property for a system is *generic* if an open, dense subset of the system possesses the property. In probabilistic terms, a property is *generic* if it is possessed with probability one.

In this context, the term generic is used to characterize which catastrophes are generic for images or for differential operators on images - these are the so-called *generic events*.

14.3.7 Dimensionality

Analysis of the dimension of the involved spaces can often determine whether a property is generic. As an example, lets look at the set of singularities in an n-dimensional image. A singularity point is determined by all n first order partial derivatives equaling zero. This means that we have n conditions in a n-dimensional space. Under the assumption that these conditions are independent, the space where the conditions are met is a $n - n = 0$ dimensional space. This means that the set of singularities contains only isolated points - or more precisely, a singularity point is generically isolated. We know that for a 2D image the singularities are the maxima, minima and saddle points of the intensity landscape, which are easily recognized as isolated points.

What about catastrophes? The potential catastrophe point is required to be a singularity (n conditions) and then the Hessian is required to have a degenerate direction (an extra condition). This means that $n+1$ conditions are to be met resulting in an n dimensional space. Or in other words: generically, a given image contains no catastrophe points.

In scale-space we have an extra parameter - the scale.

This means that in scale-space the set of singularities is generically a 1-dimensional space (the 'path' of the singularity over scale) and that catastrophes do generically occur in isolated points.

This very short and informal treatment of the dimensionalities of the singularity and the catastrophe sets is only meant as an appetizer. In order to present the above considerations properly mathematically, the image and the set of conditions should be represented as manifolds in jet-space where the independency of the terms can be investigated through the concept of *transversality*. However, this is far beyond the scope of this introduction - for a richly illustrated interactive tutorial in *Mathematica* see [Sanns2000], for a more comprehensive treatment see [Gilmore1981, Florack1994b, Olsen2000, Dam2000, Bruce1984].

14.3.8 Illustration of the concepts

The example below shows a function with two state parameters x and y and one control parameter a. The local structure differs for three choices of the control parameter a: To the left the function has a saddle and a local minimum, in the middle a single saddle, and to the right no singularities at all. In general, the function has two singularities for negative values of a, one singularity for $a = 0$, and no singularities for positive a.

```
f[x_, y_, a_] := x^3 - 24 y^2 + a x; DisplayTogetherArray[
 Plot3D[f[x, y, #], {x, -30, 30}, {y, -30, 30}] & /@ {-300, 0, 300},
 ImageSize -> 480];
```

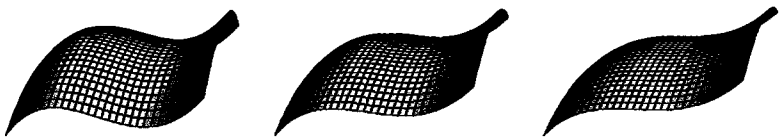


Figure 14.3 The evolution of the Fold catastrophe as a function of the control parameter a. As the control parameter changes, the structure of the function changes. For negative values of the control parameter, the function has a local maximum and a saddle point. For positive values of the control parameter, the function has no singularities in the neighborhood of the observed point. The catastrophe occurs for control parameter equal to zero.

Since the singularity structure is different for positive and negative values of a, there must be a catastrophe point for $a = 0$.

And reassuringly, for $a = 0$ the singularity at $(x, y) = (0, 0)$ is a catastrophe point as well. This is easily verified mathematically: for $a = 0$ the determinant of the Hesssian is $6x(-48)$, which is zero at the singularity point $(x, y) = (0, 0)$.

The *Splitting Lemma* states that we can split the state parameters into "good" and "bad" parameters (locally around the catastrophe point $(x, y, a) = (0, 0, 0)$ where the function changes singularity structure for increasing a).

Actually, it need not be the original state parameters that are split - the degenerate direction need not be aligned with the original axes. However, in this case the "bad" parameter is the x. As stated by the splitting lemma, the function can be split into a part with the bad parameters, and a part with the good parameters in the shape of a sum of second order monomials (a monomial is a polynomial consisting of a product of powers of variables, e.g., x, $x\,y^3$, $x^4\,y\,z^2$, etc). The non-Morse part of the function can be recognized from the list of *catastrophe germs* in *Thom's Theorem*. It is known as the *fold catastrophe*. It should be noted that - since there is only one control parameter - this is the only generic catastrophe.

```
fold[x_, a_] := x^3 + a x;
DisplayTogetherArray[
  Plot[fold[x, #], {x, -30, 30}, PlotLabel -> "a=" <> ToString[#],
     AxesLabel -> {"x", ""}, Ticks -> None] & /@
   {-300, 0, 300}, ImageSize -> 440];
```

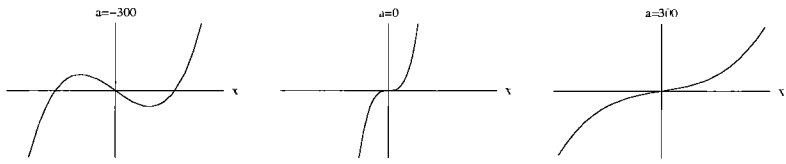


Figure 14.4 The "bad" parameter x from the previous example. The original function can be split into the Morse and the non-Morse part. The non-Morse part is structurally equivalent to the *fold* catastrophe.

The canonical fold catastrophe is further analysed below. We derive the singularity and catastrophe sets for the function fold(x, a):

```
Solve[∂x fold[x, a] == 0, {a}]
Solve[{∂x fold[x, a] == 0, ∂x,x fold[x, a] == 0}, {a, x}]
```

$$\{\{a \to -3\,x^2\}\}$$

$$\{\{a \to 0,\ x \to 0\}\}$$

The above description of the singularity set is used to plot the *fingerprint* or the *signature* of the function - the position of the singularities against the value of control parameter:

```
Plot[-3 x^2, {x, -1, 1}, ImageSize -> 150];
```

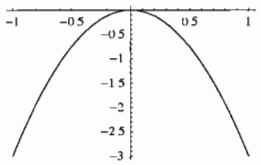


Figure 14.5 The *fold* catastrophe. The catastrophe at a=0 is where the two singularity strings meet and annihilate. A point where several singularity strings meet is also denoted a *bifurcation*. This is the *fingerprint* of the fold catastrophe. Compare with the catastrophes in the fingerprint in figure 14.11.

The *cusp* catastrophe from Thom's theorem is also commonly encountered. The canonical germ and pertubation that give rise to the cusp catastrophe is $x^4 + c_1 x^2 + c_2 x$. Since there are two control parameters, this is slightly more complicated than the fold catastrophe.

▲ Task 14.2 Illustrate the fingerprint for the cusp catastrophe (like the illustration in figure 14.5 for the fold catastrophe).

```
Clear[cusp]; cusp[x_, c1_, c2_] := x^4 + c2 x^2 + c1 x;
Show[GraphicsArray[
   Table[Plot[cusp[x, c1, c2],
     {x, -10, 10}, DisplayFunction -> Identity, Axes -> None],
    {c2, 0, -30, -15}, {c1, -80, 80, 40}]], ImageSize -> 440];
```

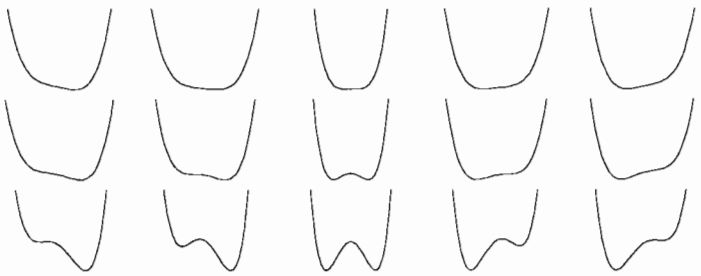


Figure 14.6 Various perturbed shapes for the cusp catastrophe germ. It has two stable states: one with a single minimum and one with a minimum,maximum,minimum triple (or the same states with maximum and minimum switched). Conceptually, one control parameter allows transition between the states by "tilting" the two minima until one minima is annihilated with the central maxima (or the reverse process) - these events are fold catastrophes. The other control parameter allows transition between the states by letting the two minima approach each other until they are merged together with the maxima into one single minimum. The cusp catastrophe point is where both control parameters are zero.

Here are a few other illustrations and some plot commands to study them (see also the package **`ImplicitPlot3D.m`** in *MathSource*: www.mathsource.com):

```
<< FrontEndVision`ImplicitPlot3D`;
DisplayTogetherArray[{
   cusp = ContourPlot[x^3 - y^2, {x, -.5, 4},
     {y, -6, 6}, Contours -> {0}, PlotLabel -> "Cusp"],
   cusp3D = ContourPlot3D[4 x^3 + 2 u x + v, {u, -2.5, 2}, {v, -2, 2},
     {x, -1, 1}, PlotPoints -> 6, PlotLabel -> "Cusp3D",
     ViewPoint -> {-4.000, 1.647, 2.524}],
   swallowtail = ParametricPlot3D[{u v^2 + 3 v^4, -2 u v - 4 v^3, u},
     {u, -2, 2}, {v, -.8, .8}, BoxRatios -> {1, 1, 1},
     PlotLabel -> "Swallowtail", ViewPoint -> {4.000, -1.390, -3.520}]},
  ImageSize -> 420, GraphicsSpacing -> {0, 0}];
```

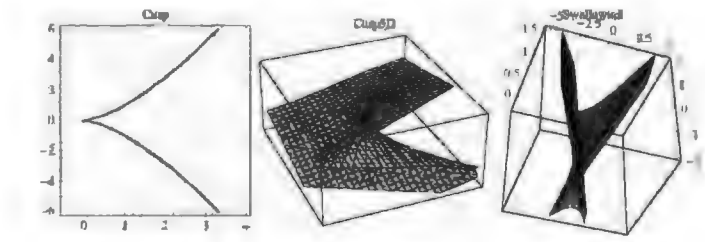


Figure 14.7 The *cusp* (2D and 3D) and the *swallowtail* catastrophe. From the wonderful booklet by [Sanns2000]. See also:
MathWorld [http://mathworld.wolfram.com/Catastrophe.html].

14.4 Catastrophe theory in scale-space

As stated earlier, the natural application of catastrophe theory towards scale-space theory is to view the spatial parameters as state parameters and the scale as the single control parameter. The differentiability of the scale-space even ensures that the functions are smooth. However, it is slightly more complicated than that. In scale-space theory, there is a fixed connection between the spatial parameters and the scale parameter given by the diffusion equation: $L_t = L_{xx} + L_{yy}$. This gives a severe restriction compared to the general space of functions with one control parameter.

Thereby, the canonical catastrophe germs listed in Thom's theorem might not apply to scale-space images. Fortunately, the work of James Damon (UNC) reveals that we can in fact still apply similar results to Thom's theorem for all practical purposes [Damon1995, Damon1997].

Another caveat in scale-space singularity analysis is the image. Images from natural scenes behave nicely, but artificial test images often possess nasty properties. Two typical examples are symmetry and areas with constant intensity. Across a symmetry axis singularities appear in pairs. This means that the expected catastrophes appear in variants where two (or more) symmetric catastrophes occur simultaneously at the same place (an example of this is shown in the next section). This apparently causes non-generic catastrophes to appear. Actually, it is not the catastrophes that are non-generic - symmetry in images is non-generic.

Areas with constant intensity in artificial test images can cause unexpected results. In theory this should cause no problems since the areas no longer have constants intensity at any given scale > 0. In practice, implementations with blurring kernels of limited size will however

leave areas with constant intensity (gradually smaller with increasing scale). A simple consequence is apparent areas of singularities - as opposed to the expected isolated points.

14.4.1 Generic events for differential operators

Thom's theorem states that the only generic catastrophe with only one control parameter is the fold.
At first hand, we would therefore not expect to encounter any other catastrophes in scale-space, where we only have scale as control parameter. However most applications, like edge detection, examine singularities for differential operators and not singularities for the raw image. Depending on the differential operator this induces other catastrophes as well.

An example of higher order singularities induced by a differential operator is illustrated in the following. The operator is the square of the first derivative of the original image (the gradient squared). In order to understand why this simple operator can induce other generic catastrophes we look at the Taylor series expansion of a one-dimensional function f (around 0 for simplicity) and the derivative of this expansion squared.

```
Clear[f]; Series[f[x], {x, 0, 4}]
```

$$f[0] + f'[0]\,x + \frac{1}{2} f''[0]\,x^2 + \frac{1}{6} f^{(3)}[0]\,x^3 + \frac{1}{24} f^{(4)}[0]\,x^4 + O[x]^5$$

When we look for singularities the zeroth order term is not interesting. Intuitively we can use the spatial parameter x as a free parameter that allows us to find points p where $f_x(p)$ is zero. Therefore we find singularities in generic signals (and images). When we have a control parameter, we can turn this extra "knob" until we find points where $f_{xx}(p)$ is zero as well. Then we find catastrophe points where the first and second order structures are vanishing - the canonical fold catastrophe germ. But since we have no more knobs to turn, we cannot get rid of the higher order structure and therefore higher order catastrophes are non-generic.

```
D[Series[f[x], {x, 0, 5}], x]^2
```

$$f'[0]^2 + 2 f'[0] f''[0]\,x + (f''[0]^2 + f'[0] f^{(3)}[0])\,x^2 + \left(f''[0] f^{(3)}[0] + \frac{1}{3} f'[0] f^{(4)}[0]\right) x^3 + \left(\frac{1}{4} f^{(3)}[0]^2 + \frac{1}{3} f''[0] f^{(4)}[0] + \frac{1}{12} f'[0] f^{(5)}[0]\right) x^4 + O[x]^5$$

The situation is somewhat different for the derivative signal squared. Again, we can use the two free parameters (x and the control parameter) to find points p where $f_x(p) = f_{xx}(p) = 0$. The remaining part of the derivative of the signal squared is then $\frac{1}{4} f_{xxx}(p)^2\, x^4 + O(x^5)$. We see that the third order structure completely disappears and we therefore generically have the cusp catastrophe present in the derivative of the signal squared. It is also worth noticing that this implies that for each fold in the original signal, there is a cusp in the derivative of the signal squared.

We first demonstrate these findings in scale-space by looking at a simple signal composed of a few sines and cosines. First the definition of the signal and the derivative squared:

```
from = -100; to = 400; resolution = 25;
f[t_] := Sin[ t / resolution ] +
   Cos[1.2 t / resolution + 0.9] + Sin[1.3 t / resolution + 0.6];
DisplayTogetherArray[Plot[#, {t, from, to}] & /@ {f[t], f'[t]^2},
  ImageSize → 470];
```

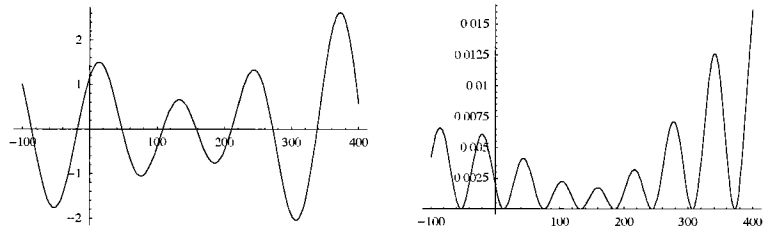


Figure 14.8 A simple signal and the corresponding derivative squared.

We investigate how this signal evolves in scale-space by convolving with a Gaussian using an exact Fourier domain implementation:

```
gausskernel[x_, σ_] := 1 / (σ √(2 π)) Exp[- x^2 / (2 σ^2)];
signal[σ_, x_] = Simplify[
   InverseFourierTransform[FourierTransform[gausskernel[x, σ], x, ω]
     FourierTransform[f[x], x, ω], ω, x], σ > 0];
dsignalsquared[σ_, x_] = (∂x signal[σ, x])^2 // Chop;
DisplayTogetherArray[
  {Plot[signal[#, x], {x, from, to}, Ticks → {Automatic, None},
     PlotLabel → "σ = " <> ToString[#]] & /@ {1, 25, 40, 50},
   Plot[dsignalsquared[#, x], {x, from, to},
     Ticks → {Automatic, None}] & /@ {1, 25, 40, 50}}, ImageSize → 500];
```

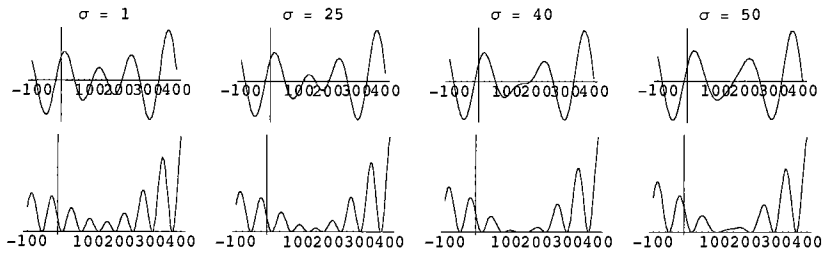


Figure 14.9 Top row: the original signal at scales $\sigma = 1, 25, 40, 50$. At the center of the signal a maximum and a minimum melt together as scale increases and are annihilated in a fold catastrophe. Bottom row: the derivative squared at the same scales. A {minimum, maximum, minimum}-triple is annihilated into a single minimum in a cusp catastrophe located where the fold is in the original signal.

In order to see that these nice derivations actually hold for real discrete images as well, we inspect a random signal. First the signal and the derivative squared is constructed.

```
random = Table[Random[], {t, from, to}];
scaledrandom[σ_] := gDf1D[random, 0, σ];
drandomsquared[σ_] := gDf1D[random, 1, σ]^2;
```

We then illustrate the evolution as scale increases:

```
view[func_] :=
  ListPlot[func[#], PlotJoined → True, Ticks → {Automatic, None},
     PlotLabel → "σ = " <> ToString[#]] & /@ {1, 25, 40, 50};
DisplayTogetherArray[{view[scaledrandom], view[drandomsquared]},
  ImageSize → 500];
```

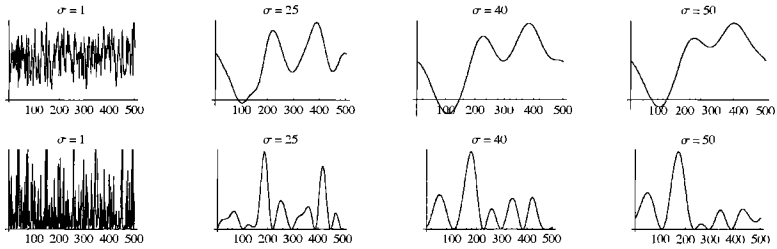


Figure 14.10 The scale-space evolution for a random signal and the derivative signal squared. The signals are displayed at scales σ = 1, 25, 40, 50 as in figure 14.9.

Finally we display the fingerprints for the random signal and the derivative of the signal squared. The fingerprints are slightly cluttered at low scale due to the inherent randomness of the signal but the fold/cusp pairs are obvious.

```
fingerprint[signal_, maxscale_] := Module[{scsig, scLx, sig},
   scsig = Table[signal[E^(t Log[maxscale]/200)], {t, 50, 200}];
   scLx = scsig - Map[RotateLeft, scsig];
   sig = Map[(RotateLeft[#] - #) &, Sign[scLx]];
   ListDensityPlot[sig, ImageSize → 440]];
fingerprint[scaledrandom, 60];
```

Figure 14.11 Fingerprints for the random signal and the derivative signal squared. For each fold catastrophe in the signal there is a corresponding cusp catastrophe in the derivative signal squared.

Thom's theorem states that the only generic catastrophe for a function with only one control parameter is the *fold*. However, as we have seen in the previous section, when we construct *differential* expressions from the original generic function we can induce higher order catastrophes as well.

A simple example is the derivative squared where the cusp catastrophe is generic.

```
fingerprint[drandomsquared, 60];
```

Other differential expressions can induce catastrophes of even higher order. Therefore, it is necessary to analyze each differential expression individually in order to reveal the generic events for the singularities as scale is increased.

14.4.2 Generic events for other differential operators

Other corners measures are studied in [Sporring1998a].

Among the investigated differential operators are the gradient magnitude used above as dissimilarity measure for the watershed segmentation. The generic events for this operator are the fold and the cusp - for both annihilations as well as creations are generic.

```
Show[Import["nongeneric_isocat.jpg"], ImageSize → 230];
```

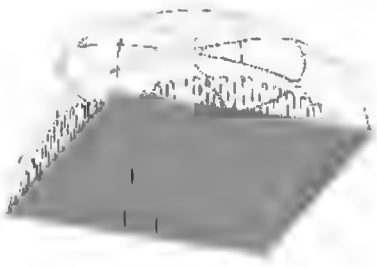


Figure 14.12 The singularities for the isophote curvature located on a specific isophote The red/blue curves are the maxima/minima. These singularities can be perceived as corners. The singularity strings are followed up through scale-space revealing non-generic catastrophes. Both the blue ring and the catastrophe at the top, where eight singularities are annihilated, display highly non-generic behavior. This is due to the non-generic test image - a perfect square. Illustration from [Dam1999].

This is derived in [Olsen1997]. Note, however, that creations only are generic in 2D and higher dimensions. Creations in 1D can never occur.

The generic events for the isophote curvature are studied in [Dam1999]. Again, the generic events are the annihilation and creation fold and cusp catastrophes. From this work we also have an ensemble of apparently non-generic catastrophes due to symmetry in the test images. An example is displayed in figure 14.12.

14.4.3 Annihilations and creations

In traditional catastrophe theory there is no preferred orientation for the control parameters. When the singularity structure for a function changes new singularities are as likely to appear as old ones are to disappear. Annihilations and creations are simply reverse events between different states for the local structure of a function. This is not the case for linear scale-space functions where scale is the control parameter. We only study the evolution of the functions for *increasing* scale. As mentioned earlier, increasing the scale results in a general simplification of the image functions. This implies that singularities are annihilated much more often that they are created.

As an example, see the fingerprint for the noisystep signal in figure 14.11. The figure shows fold annihilations - but no creations at all. This is in fact no coincidence. For a 1D signal (with no special properties or symmetries), creations are non-generic in scale-space. For images of dimension 2 (or higher) creations are generic. The famous example from the literature, where this was first discovered by Lifshitz and Pizer [Lifshitz1990], is the 'dumb-bell example' (see figure 14.13).

```
db = Table[Chop[Exp[-(x - π)^2/2] Exp[-y^2/(2 (Sin[x] + .1)^2)]],
   {y, -4, 4, .2}, {x, 0, 2 π, .04}];
DisplayTogetherArray[
ListPlot3D[#, ViewPoint -> {1.489, -2.605, 1.968}, Mesh -> False] & /@
   {db, gD[db, 0, 0, 3]}, ImageSize -> 300];
```

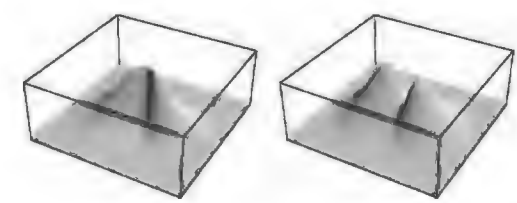


Figure 14.13 The dumb-bell image (left) consists of two blobs with a narrow bridge in between with a maximum. Blurring (right, $\sigma = 3$) has much more effect on the bridge, so two new maxima and a saddle are created under blurring. Illustration from [Lifshitz1990].

In general for catastrophes from images or differential operators on images, annihilations are much more common than creations [Kuijper2002a, Florack2000b, Kuiper2002b]. Furthermore, the scale-range where singularities can be observed resulting from a creation in scale-space is generally relatively short.

Another interesting aspect about singularities resulting from creations is their "validity". In a sense, the singularities that arise from creations have no originating structure in the original image. The feature corresponding to such a singularity can not be linked down to its "cause" in the image. Therefore, the singularities arising from creations are discarded in certain applications. An example of this is actually the linking scheme in the multi-scale watershed segmentation. Since the regions are linked from fine to coarse scale, regions resulting from creations never enter the linking tree. They are simply ignored on purpose.

14.5 Summary of this chapter

Catastrophe theory is the theoretical foundation that allows us to analyze the evolution of the singularities in scale-space. The theory allows prediction of the behavior of differential operators in scale-space. Ideally, any deep structure application should therefore include a comprehensive analysis of the generic behavior.

However, catastrophe theory is quite complicated. Therefore a strict theoretical approach has little appeal for the general image analysis community. Fortunately, a number of central results can be applied without more than a certain intuitive understanding of catastrophe theory. The purpose of this section was to provide a first step towards such a basic intuitive understanding.

15. Deep structure III. topological numbers

Bart M. ter Haar Romeny and Erik Dam

In the previous chapters we detected and followed the singularity strings through scale-space in an ac hoc manner. In the scale selection section, the detection of maxima was done by simply looking for pixels with values larger than its neighbors. In the edge focusing section, minima and maxima were detected (and distinguished) for a 1D signal by looking at sign changes for the derivative signal. Furthermore these extrema were tracked down through scale-space by simply looking for extrema in a close neighborhood in successive scale levels below. Finally, in the multi-scale segmentation section, the dissimilarity minima were represented indirectly by the catchment basins, and the linking across scale was done robustly by matching regions instead of points.
These approaches seem somewhat heuristic. However, they can be expressed in a more formal manner with solid theoretical foundations. Furthermore, the implementations can be refined in order to make them more robust. The approach presented in this section is therefore not to be considered superior to the previous. It does, however, have a number of properties that are appealing both in theory and implementation. The concept presented in the following can be studied in more detail in [Kalitzin1996a, Kalitzin1997b, Staal1999a].

15.1 Topological numbers

The *topological number* defined below is a generalization of the *winding number*. The topological number gives a characterization about the local structure of a function in a point by investigating the immediate neighborhood of the point (*divergence theorem*, see http://mathworld.wolfram.com/DivergenceTheorem.html).

Let $L(x): C^1(\mathbb{R}^n, \mathbb{R})$ be a differentiable scalar-valued function. For a given regular point p (a point with non-vanishing gradient) we define

$$\Phi(p) = \frac{L_{i_1}\, dL_{i_2} \wedge \ldots \wedge dL_{i_n}\, \epsilon^{i_1 i_2 \ldots i_n}}{(L_j(p)\, L_j(p))^{n/2}}$$

Here ϵ is the permutation tensor defined by the following equations: $\epsilon^{1\,2\,\ldots\,n} = 1$ and $\epsilon^{i_1\, i_2 \ldots i_k \ldots i_l \ldots i_n} = -\epsilon^{i_1\, i_2 \ldots i_l \ldots i_k \ldots i_n}$. This is simply a sign change depending on the number of permutations.

For a closed, oriented, n-1 dimensional hypersurface S with no singular points on the surface we define:

$$\nu_S = \int_S \Phi(p)$$

For a point p_0, the topological number $\nu(p_0)$ is defined as the topological number ν_S for a hypersurface S surrounding p_0 closely. By the informal notion "closely" we specifically mean that the region bounded by the hypersurface must contain no singularities, other than possibly p_0, for the function L. In order for this to be well-defined, it is required that the singular points for the function L are isolated.

Characterization of Points:
For a regular point the topological number is zero. For a singularity point, the topological number offer a characterization of that point (examples are given below).

Invariance:
The topological number is invariant under homotopic deformations of the n-dimensional space on which the hypersurface is embedded - provided that no singular points cross the hypersurface in the deformation. This makes it a non-pertubative and topological quantity.

Additivity:
For a region constructed as the union of a number of disjoint subregions, the topological number for the hypersurface bounding this region is the sum of the topological numbers for the subregions.

Conservation:
The invariance property implies the following conservation property. For a family of images $L(x, \sigma) : C^\infty(\mathbb{R}^{n+1}, \mathbb{R})$, depending smoothly on the deformation parameter σ, the topological number ν_S for a given hypersurface S is constant for all σ provided that no singular points crosses the hypersurface during deformation. This property is central for the tracking of singularities and the analysis of catastrophe points in scale-space.

15.1.1 Topological numbers in scale-space

The linear scale-space representation for n-dimensional image functions has three central properties that makes the concept of topological numbers applicable:

- Image functions are infinitely differentiable → continuous derivatives in scale-space, $\sigma > 0$.
- Singularities are generically isolated in a scale-space image for a given scale.
- The deformations defined by Gaussian blurring are smooth.

Thereby image functions (and their derivatives) qualify for the conditions stated above. The topological number is well-defined for scale-space image functions. When analyzing the deep structure of image in scale-space, the invariance and conservation properties for the topological number mentioned above are central. Below, we will see this with respect to tracking of singularities and the analysis of catastrophe points in scale-space.

15.1.2 Topological number for a signal

The above mathematical definition of the topological number simplifies considerably for 1D signals in scale-space. For a scale-space signal $L(x, \sigma) : C^{\infty}(\mathbb{R}^{1+1}, \mathbb{R})$, the topological number for a point p is simply $\nu(p) = \mathrm{Sign}(L_x(p_+)) - \mathrm{Sign}(L_x(p_-))$ where p_- and p_+ are points close to p such that $p_- < p < p_+$.

Here "close" has the same meaning as defined above. For regular points, the topological number is zero. For local maxima and minima the topological numbers are -2 and 2, respectively.

In the section on edge focusing, the illustration of the signature of the *noisystep* signal was produced in an ad hoc manner. Inspection of the code reveals that the illustration displays exactly the -2 and 2 values for the sign differences for the derivative of the signal. Reassuringly, the sensible ad hoc approach proves to be a simple special case of a more general principle.

15.1.3 Topological number for an image

For 2D images the expressions simplify quite a bit as well. The integrand from the topological number simplifies to the following expression:

$$\Phi(p) = \frac{L_1 \, dL_2 - L_2 \, dL_1}{L_1^{\,2} + L_2^{\,2}}$$

Using complex numbers, this can also be written as:

$$\Phi(p) = \mathrm{Im}(L_1 - i L_2) \frac{d(L_1 + i L_2)}{L_1^{\,2} + L_2^{\,2}}$$

For a discrete image, this is simply the angle between the gradient vector for two neighboring pixels. When this expression is integrated (or summed in the discrete setting) the topological number is therefore a count of the number of times the gradient vector turns around its origin as a contour surrounding a specific point is traversed. This is known as the *winding number*.

The winding number assumes values of $k\,2\pi$ for some integer k. In order to understand this intuitively, picture the gradient direction vector rotating for a moving test point that encircles a given image point.

For regular points the winding number is zero. For local extrema the winding number is $+ 2\pi$. For saddle points the winding number is $- 2\pi$. Monkey saddles can be characterized by the winding number as well - we will not look into that here.

```
<< FrontEndVision`FEV`;
ϕ = 2 π / 3; DisplayTogetherArray[
 Show[{PlotVectorField[Evaluate[{∂x #, ∂y #}], {x, -2, 2}, {y, -2, 2}],
     Graphics[{Hue[.4], Circle[{0, 0}, 1], Hue[0], Circle[{x, y}, .1],
        Evaluate[Arrow[{x - ∂x # / 3, y - ∂y # / 3}, {x + ∂x # / 3, y + ∂y # / 3}]]} /.
       {x -> Cos[ϕ], y → Sin[ϕ]}]}] & /@
  {x^2 + y^2, (x + 2)^2 + y^2, x^2 - y^2}, ImageSize -> 400];
```

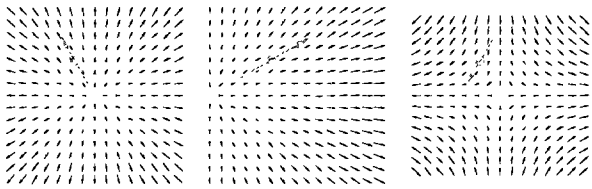


Figure 15.1 Path of the rotating gradient vector with the gradient vectorfield for three different functions. Left: $f(x, y) = x^2 + y^2$; the gradient vector rotates once forward → enclosed is a maximum, the winding number is 2π. Middle: $f(x, y) = (x + 2)^2 + y^2$; the gradient vector does not make any rotation when the path is traversed → enclosed is a regular point, winding number is zero. Right: $f(x, y) = x^2 - y^2$; the gradient vector rotates once backwards → enclosed is a saddle point, the winding number is -2π.

▲ Task 15.1 Show that the winding numbers add for the singularities within a closed path, e.g. when 2 maxima (each windingnumber $+2\pi$) and a saddlepoint (windingnumber -2π) are traversed, the rotation is $2\pi+2\pi-2\pi=2\pi$.

▲ Task 15.2 Show that the winding number for a monkeysaddle $f(x, y) = x^3 + 3xy^2$ is -4π and illustrate this with an animation. A monkeysaddle is so called to accommodate the two legs and the tail of the monkey.

▲ Task 15.3 Calculate the windingnumber of higher order monkeysaddles, given by $f(x, y) = \mathrm{Re}[(x - iy)^n]$, where $n \in \mathbb{N}$, and illustrate this with an animation.

15.1.4 The winding number on 2D images

First, we develop the routine for winding number detection in 2D images. The routine **`windingnumber2D[im,τ]`** returns a list of two elements **`{extrema,saddles}`**, i.e. the positions of the extrema and the positions of the saddles, both as triples of coordinates in scale-space.

A triple is **`{σ,y,x}`**. The routine pursues the complex notation of the integrand $\Phi(p)$ above. This is calculated in ϕ for the eight neighbors of each pixel and summed in *wn*. Each extrema and saddle is located twice - therefore the results are "pruned" through pattern matching on *wn*.
For ease of calculations the components of the gradient are represented as the real and

imaginary components of a complex number. The argument **Arg** of the ratio of two such numbers is then the angle between the gradient vectors.

```
windingnumber2D[im_, τ_] := Module[
   {grad, v, wn, extrema, saddles}, grad = gDf[im, 1, 0, E^τ] + I gDf[im, 0, 1, E^τ];
   v = RotateLeft[grad, #] & /@ {{0, 0}, {-1, 0}, {-1, -1}, {0, -1}};
   wn = 1/(2 π) Plus @@ Arg[v / RotateLeft[v]] // Round;
   {extrema, saddles} = (Insert[#, E^τ, 1] & /@ Position[wn, #]) & /@ {1, -1}];
```

We calculate the winding numbers on a sagittal MR image ($\sigma = 2.7$ pixels) and plot the extrema and saddle points as white pixels on the image::

```
im = Import["mr128.gif"][[1, 1]];
{extrema, saddles} = windingnumber2D[im, τ = 1.];
imbl = gD[im, 0, 0, Exp[1]];
white[x_] := Max[imbl]; disp[pos_] := Module[{},
  positions = pos /. {σ_, y_, x_} -> {y, x};
  ListDensityPlot[MapAt[white, imbl, positions]]];
DisplayTogetherArray[disp /@ {extrema, saddles}, ImageSize -> 265];
```

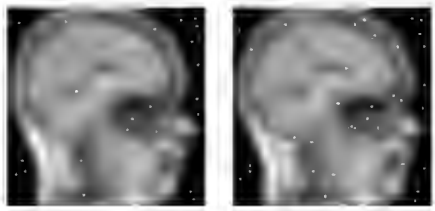


Figure 15.2 The local extrema (left) and saddle points (right) for the MR brain image at scale $\sigma = 2.7$ ($\tau = 1$) superimposed as white dots on the original image blurred to $\sigma = 2.7$.

Where the extrema are typically in the center of regions, the saddle points are located near the borders of regions. Here are the critical (self-intersecting) isophotes through the saddle points #66, #68, #69 and #72:

```
saddlephote[nr_] := Block[{$DisplayFunction = Identity, saddle},
   saddle = positions[[nr]]; pl = ListDensityPlot[imbl,
     Epilog -> {Yellow, PointSize[.03], Point[Reverse[saddle]]}];
p2 = ListContourPlot[imbl, ContourStyle -> Wheat,
     Contours -> {Extract[imbl, saddle]}];
   Show[p1, p2, DisplayFunction -> $DisplayFunction]];
DisplayTogetherArray[saddlephote /@ {66, 68, 69, 72}, ImageSize -> 400];
```

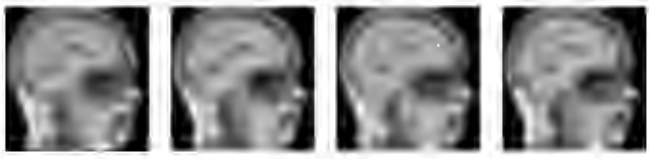


Figure 15.3 The isophote curves defined by the values at some saddle points (yellow dots) for the MR brain scan of figure 15.2 at scale $\sigma = 2.7$. Note that all these critical isophotes intersect themselves in saddlepoints.

```
criticalisophotes[im_, τ_] :=
  Module[{}, {extrema, saddles} = windingnumber2D[im, τ];
imblurred = gD[im, 0, 0, Eτ];
   saddlepositions = saddles /. {σ_, y_, x_} -> {y, x};
ListContourPlot[imblurred, ContourShading -> True,
    Contours → Extract[imblurred, saddlepositions]]];
DisplayTogetherArray[criticalisophotes[im, #] & /@ {1.5, 2, 2.5},
  ImageSize -> 300];
```

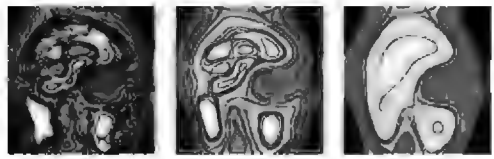


Figure 15.4 The isophote curves defined by the values at the saddle points for the MR brain scan at scales σ = 4.5 (left) and σ = 7.4 (right) pixels. Note that these *critical isophotes* intersect themselves in saddlepoints.

In the previous chapter on catastrophe theory we studied the rate of the decrease of image maxima. Using the winding number approach we can now detect extrema as well as saddle points. For the sake of comparison, we inspect the rate of decrease of the number of singularity points for the MR brain image and a white noise image:

```
wnImage = Table[windingnumber2D[im, τ], {τ, 0, 1, .1}];
{nrExtremaImage, nrSaddlesImage} =
  Map[Length, wnImage, {2}] // Transpose;
wnNoise = Table[windingnumber2D[Table[Random[], {128}, {128}], τ],
  {τ, 0, 1, .1}]; {nrExtremaNoise, nrSaddlesNoise} =
 Map[Length, wnNoise, {2}] // Transpose;
disp[nrSingularities_, text_] := Module[{},
   slopeText = "Slope for " <> text <> ": " <>
     ToString[Coefficient[Fit[10 Log[nrSingularities], {1, x}, x], x]];
   LogListPlot[nrSingularities, AxesLabel -> {"τ*10", ""},
    PlotLabel -> slopeText, DisplayFunction -> Identity]];
Show[GraphicsArray[{disp[nrExtremaImage + nrSaddlesImage, "MR image"],
    disp[nrExtremaNoise + nrSaddlesNoise, "white noise"]}],
  ImageSize -> 345];
```

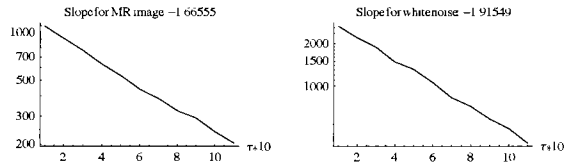


Figure 15.5 The decrease of singularities (the sum of the extrema and saddles) in a 128^2 MR image (left) and an image with white noise (right). The singularities are detected using 2D winding numbers. Especially the curve for white noise has a slope close to the theoretically predicted -2.

We see that for the MR image and the white noise the observed slope (also known as the *Hausdorff dimension*) is close to the theoretical slope of -2. The MR image has a more

deviating slope, possibly due to the lesser degree of self-similarity of the MR image at different scales (see also [Pedersen2000]).

We can increase the radius of the path around the singular point to one pixel (8 pixels on our track):

```
windingnumber2D8[im_, τ_] :=
  Module[{σ, grad, v, ϕ, a, i,
    wn, shead, stail, ehead, etail, extrema, saddles},
   σ = Exp[τ]; grad = gD[im, 1, 0, σ] + I gD[im, 0, 1, σ];
   v = wn = Table[0, {8}];
   v[[1]] = RotateLeft /@ grad; v[[2]] = RotateRight[v[[1]]];
   v[[3]] = RotateRight[grad]; v[[4]] = RotateRight /@ v[[3]];
   v[[5]] = RotateRight /@ grad; v[[6]] = RotateLeft[v[[5]]];
   v[[7]] = RotateLeft[grad]; v[[8]] = RotateLeft /@ v[[7]];
   ϕ = Table[If[i == 8, a = i, a = i + 1]; Arg[v[[i]]/v[[a]]], {i, 1, 8}];
   wn = Round[1/(2 π) Plus @@ ϕ];
   wn = wn //. {{shead___, -1, -1, stail___} → {shead, 0, -1, stail},
      {ehead___, 1, 1, etail___} → {ehead, 0, 1, etail}};
   wn = Transpose[Transpose[wn] //.
      {{shead___, -1, -1, stail___} → {shead, 0, -1, stail},
       {ehead___, 1, 1, etail___} → {ehead, 0, 1, etail}}];
   extrema = Insert[#, σ, 1] & /@ Position[wn, 1];
   saddles = Insert[#, σ, 1] & /@ Position[wn, -1]; {extrema, saddles}];
```

Sander and Zucker [Sander1992] employed winding numbers with respect to the vectorfield corresponding to the principal directions (of the principal curvatures) to detect umbilical points (as singularities in this second order vectorfield).

15.2 Topological numbers and catastrophes

The conservation property for the topological number offers a way to analyze catastrophe point.s Specifically, the conservation property (and the additivity property) states that the sum of the topological nrs for the participating singularities are constant across a catastrophe.

```
{min, max, step} = {-30, 30, .5}; {τmin, τmax, τstep} = {1, 3.5, 0.3};
im = Table[x^3 - 24 y^2 - 300 x, {x, min, max, step}, {y, min, max, step}];
DisplayTogetherArray[
  {ListPlot3D[im, ViewPoint -> {3.147, -0.630, 1.071}, Mesh -> False],
   ListDensityPlot[Transpose[im]]}, ImageSize → 400];
```

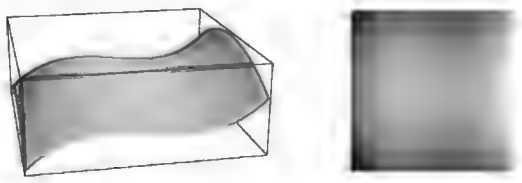


Figure 15.6 A function with a local maxima and a saddle point. The function is displayed both as a height map and a density plot.

We illustrate this by the fold catastrophe presented in the catastrophe theory section. The family of functions $x^3 - 24y^2 + ax$ has a function with a fold catastrophe for $a = 0$.

Here we let linear diffusion produce the same catastrophe on a discrete version of the function $x^3 - 24y^2 - 300x$.

The extrema and the saddle points are calculated with the winding number method:

```
wnSingularities = Table[windingnumber2D8[im, τ], {τ, 1, 3.5, .3}];
```

We display the extrema points, saddle points, and the original image via the 3D graphics primitive **Point**. This means that image points, extrema, and saddles are converted into lists of triples of {x, y, scale}. For the image the scale level is set to 0. The following code pieces are quite simple - however they are somewhat cluttered by the many conversions between scale-space coordinates and matrix indices.

```
{minL, maxL} = {Max[im], Min[im]};
imagepoints = Flatten[Table[{GrayLevel[(x^3 - 24 y^2 - 300 x - minL)/(maxL - minL)],
    Point[{Round[(x - min)/step] + 1, Round[(y - min)/step] + 1, 0}]},
  {x, min, max, step}, {y, min, max, step}], 1];

index[coord_] := coord /. {τ_, x_, y_} :> {x, y, Round[(Log[τ] - 1) / .3] + 1};
allExtrema =
 Flatten[wnSingularities /. {extrema_, saddles_} :> extrema, 1];
allSaddles =
  Flatten[wnSingularities /. {extrema_, saddles_} :> saddles, 1];

Show[Graphics3D[
   {imagepoints, RGBColor[1, 0, 0], Point[#] & /@ index[allExtrema],
    RGBColor[0, 0, 1], Point[#] & /@ index[allSaddles]}],
  ImageSize -> 400, ViewPoint -> {0.058, -1.840, 0.801}];
```

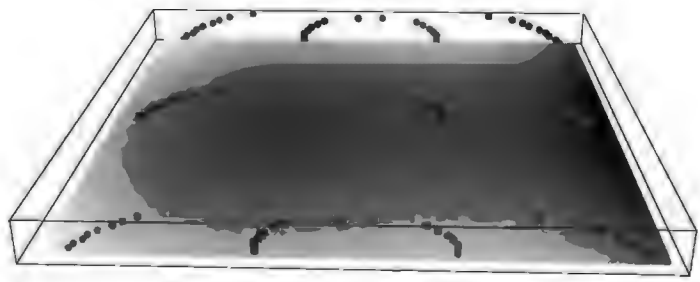


Figure 15.7 The singularity strings are displayed above the original image with scale level as the vertical axis. The red dots are extrema and the blue dots are saddles. All singularities but the two strings in the middle of the image are due to the cyclic representation of the image in the winding number computation. The red string in the center is the local maxima and the blue string is the saddle - see figure 15.6 for comparison.

The fold catastrophe in the center of the figure obeys the conservation principle of the topological number. Below the catastrophe point there are a maximum (winding number $+2\pi$) and a saddle (winding number -2π) - the sum of the winding numbers is 0.

Above the catastrophe there are no singularities (within the center region) which means that the winding number is 0. The winding number for the center region is thus preserved across the catastrophe.

15.3 The deep structure toolbox

The previous sections in this chapter have presented and applied a number of techniques for analyzing and utilizing the deep structure. The purpose of this section is to provide a summary of the open problems in deep structure analysis from a pragmatic point of view.

The desired goal is the to be able to present a complete toolbox consisting of simple, intuitive, robust and theoretically well-founded methods that handle the common challenges concerning deep structure analysis. Such a toolbox would make deep structure approaches more accessible for computer vision and image analysis researchers and programmers outside the scale-space community.

15.3.1 Detection of singularities

A number of methods have been illustrated in the previous sections. The methods locate singularities for a given n-dimensional image and for a given scale-space representation for an image as well. The methods allow characterization of the singularities as maxima, minima or saddle points.

This is the best equipped compartment in the deep structure toolbox.

15.3.2 Linking of singularities

The previous sections have shown a number of ways to detect and analyze singularities. A central aspect of exploring the singularities is the linking across scale.

The linking method for singularities in the edge focusing section is quite heuristic. A singularity is linked with the first singularity of the same type encountered at the adjacent scale level in a fixed search space around the location at the current level.

Especially in the presence of catastrophes, where the singularities will have a large horizontal movement between two scale levels, this approach is not as robust as desired.

The first step towards a more healthy approach is to adapt the search region to the local geometry.

As well as detection the location of the singularities in scale-space we can detect their local *drift velocity* [Lindeberg1992b, Lindeberg1994a]. This could be used to give an estimate for a sensible location of the search region.

For a non-degenerate singularity in the point x_0 the singularity will drift according to $\frac{\partial x_i}{\partial t} = -L_{ij}{}^{-1} L_{jt}$. This describes how the coordinates of the singularity change for increasing scale (where $t = \frac{1}{2}\sigma$). The expression is written in tensor notation, where $L_{ij}{}^{-1}$ is the inverse of the Hessian matrix and L_{jt} is the Laplacian of the gradient.

As an illustration, we equip the singularities from the singularity strings in figure 15.7 with drift velocity vectors. The differential operators are derived in *Mathematica* symbolically and then substituted with the corresponding discrete image data from the previous example.

```
Clear[L, x, y]; hessian = ( ∂x,x L[x, y]   ∂x,y L[x, y] )
                          ( ∂x,y L[x, y]   ∂y,y L[x, y] );
grad = {∂x L[x, y], ∂y L[x, y]};
laplaceGrad = ∂x,x grad + ∂y,y grad;
singDrift = -1/2 Inverse[hessian].laplaceGrad;
driftField[image_, σ_] :=
  singDrift /. Derivative[dx_, dy_][L_][x, y] :→ gD[im, dx, dy, σ];
```

This is the expression for the drift velocity of the singularity points:

```
singDrift // shortnotation
```

$$\left\{\frac{L_{xxy}\,L_{xy} - L_{xxx}\,L_{yy} - L_{xyy}\,L_{yy} + L_{xy}\,L_{yyy}}{-2\,L_{xy}^2 + 2\,L_{xx}\,L_{yy}},\ \frac{L_{xy}\,(L_{xxx} + L_{xyy}) - L_{xx}\,(L_{xxy} + L_{yyy})}{-2\,L_{xy}^2 + 2\,L_{xx}\,L_{yy}}\right\}$$

For each detected singularity we calculate a drift vector. This vector is represented by a **Line** graphics object from the singularity point in scale-space coordinates. The drift velocity vector is a first order estimate of the change in spatial coordinates for a change in scale level. Close to a fold catastrophe, the singularity string is approximately horizontal. The estimated drift for a change in scale level is therefore extremely high for the singularity points detected close to the catastrophe point.

Therefore we crop the vectors - otherwise they would point far out of the view volume.
The extrema and saddle points are picked from the **wnSingularities** variable calculated in the previous section. The calculation is done for all singularities from a scale level simultaneously such that the drift velocity field need only be computed once per scale level.

```
vectors[im_, τIdx_, extrema_, saddles_] := Module[{},
   σthis = Exp[τmin + τIdx τstep];
   σnext = Exp[τmin + τIdx τstep + τstep];
   drift = driftField[im, σthis] (σnext² - σthis²);
   crop[coordinate_] := Max[Min[coordinate, 1 + (max - min) / step], 1];
   replaceRule =
    {ττ_, x_, y_} :→ Line[{{x, y, τIdx}, {crop[x + drift[[2, x, y]]],
        crop[y + drift[[1, x, y]]], τIdx + 1}}];
   {Replace[extrema, replaceRule, 1],
    Replace[saddles, replaceRule, 1]}];
```

We limit the illustration to the two center singularity strings that meet in the fold catastrophe point. The singularity strings resulting from the cyclic borders representation are unnecessary

clutter (and their drift velocities are somewhat erratic as well due to implementation details). The selection of the center singularities is done in a simple ad hoc manner:

```
width = (max - min) / step;
centerOnly[{τ_, x_, y_}] :=
  Abs[y - width / 2] < 0.2 width && Abs[x - width / 2] < 0.2 width;
centerSing[singList_] := Select[singList, centerOnly];
```

We compute the drift velocity vectors and attach them to the singularities in the illustration:

```
driftLines = vectors[im, #, centerSing[wnSingularities[[#, 1]]],
     centerSing[wnSingularities[[#, 2]]]] & /@
   Range[1, Floor[(τmax - τmin) / τstep] + 1];
allExtremaLines = driftLines /.
   {extremaLines_, saddlesLines_} :> extremaLines;
allSaddlesLines = driftLines /.
   {extremaLines_, saddlesLines_} :> saddlesLines;
g3D = Graphics3D[{imagepoints, RGBColor[1, 0, 0],
    Point[#] & /@ index[centerSing[allExtrema]], allExtremaLines,
    RGBColor[0, 0, 1], Point[#] & /@ index[centerSing[allSaddles]],
    allSaddlesLines}, ImageSize → {440, Automatic}];
Show[g3D, ViewPoint → {0.045, -3.383, -0.009}, AspectRatio -> 0.3];
Show[g3D, ViewPoint -> {-0.700, -3.034, 1.324}];
```

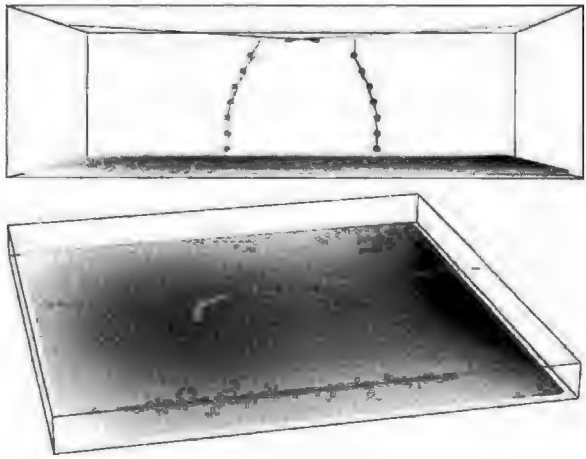


Figure 15.8 The singularity strings from figure 15.6 with drift velocity vectors added. The figure displays the same scale-space from two different angles. The drift velocities of the singularities could be applied for a first step towards more robust linking across scale. At the catastrophe point, the singularity string is horizontal in scale-space. The drift velocity vectors are therefore extremely long for the singularity points just below the catastrophe. In the illustration they are cropped to fit inside the view.

The drift velocity vectors does indicate that the search region for a linking method could be refined extensively and thereby make the method much more robust.

However, as the figure suggests, the drift velocity is not ideal for handling the linking in the presence of catastrophes. Indeed the term is undefined for degenerate singularities. Therefore more elaborate schemes are necessary to handle these cases.

An appealing approach is to exchange the drift velocity vector with a true tangent vector for the singularity string curve. This simply implies treating the singularity string as an ordinary curve in an ordinary $n+1$ dimensional space. We would then be able to track the curve equally well horizontally as vertically in scale-space. However, this is somewhat complicated and computationally expensive (or even more complicated) since we then need to sample the scale levels arbitrarily depending on the evolution of the string.

Furthermore, there is still the desire for explicit linking through higher order catastrophe points with more complicated bifurcations than the fold. All in all, linking of singularities is non-trivial and no simple, general, and robust toolbox method is readily available. However, for specific applications where only simple catastrophes occur (i.e. fold catastrophes), an approach where the search region is adapted to the differential structure of the singularity position could be extremely effective.

15.3.3 Linking of contours

In section 13.5 the reader was challenged to extend the edge focusing technique from 1D to 2D. Instead of linking points across scale the task is to link contours. This is indeed a quite challenging task.

We will propose no general method. However, it would appear that the approach in section 13.6 on multi-scale segmentation is the reasonable solution. Linking of regions with maximal overlap is conceptually far simpler than any contour linking scheme.

However, like linking of singularity points, the catastrophe points pose special problems. For the multi-scale watershed segmentation method, derivation and analysis of the generic catastrophes reveal that the maximal overlap linking scheme is indeed sensible in the presence of catastrophes. This is not necessarily true for regions defined in another manner.

15.3.4 Detection of catastrophes

Catastrophe theory reveals the catastrophes that can be expected for a given differential operator - at least in principle. The introduction to catastrophe theory given in chapter 14 is not detailed enough to allow the reader to derive the generic events for other differential expressions. See [Olsen2000] for a more thorough treatment with applications in image analysis.

Even though we know the generic catastrophes, detecting them in images is not trivial. Mathematically it is quite simple. For a given image function the catastrophes are located where $\frac{\partial f}{\partial x_i} = 0$ and $\mathrm{Det}\left(\frac{\partial f}{\partial x_i}\right) = 0$ (written in tensor notation). We know that the singularities are curves in scale-space and the zero-points for the determinant of the Hessian form a surface. The catastrophes are located where the singularity curves intersect the surface.

These differential expressions can easily be constructed using gaussian derivatives. However, we have already seen that forming singularity strings is non-trivial - particularly in the presence of catastrophe points. Furthermore, the zero-crossings for scale-spaces corresponding to these expressions will generally not coincide with the pixel grid. This makes it non-trivial to locate the intersections of the zero-crossings.

In [Kuijper99] the catastrophes are detected and their locations are determined with sub-pixel precision. Here the image is locally modelled at each pixel by a third order model. For each pixel the model then points towards a specific location for a catastrophe. For a small window, the estimated catastrophe locations are then averaged.

In the vicinity of catastrophes, the estimates are well-aligned and the averaged estimate have a small standard deviation.

Intuitively, these search windows are then moved across scale-space and catastrophes are detected where the standard deviation for the catastrophe location is locally minimal. However, this method is not quite as simple and robust as desired. The method depends on a arbitrary search window and the estimated catastrophe must be located within the search window. Possibly, this makes it particularly sensitive at higher order catastrophes.

15.3.5 General discrete geometry approach

The deep structure toolbox should ideally contain a simple unifying approach. The simple approach sketched below handles many of the problems regarding linking and detection that arises from the discretization of data in image analysis applications.

From analysis we define conditions in terms of equations and inequalities. Then we combine these to get the desired solutions. We simply need to implement this approach for discrete data. Equations and inequalities would be defined in terms of differential operators. These are naturally discretized by Gaussian derivatives as described in previous chapters. Zero-crossings in the discrete data corresponds to equations or to separators for inequalities. These zero-crossings must be determined with sub-pixel precision.

A recipe for handling these issues could be:
a) Express the problem as a set of equations and inequalities using combinations of differential operators.
b) Transform each equation or inequality such that the solution is expressed in terms of a zero-crossing.
c) Calculate a discrete "image" volume for each equation and inequality. The involved differential operators correspond to a discrete volume for a combination of Gaussian derivatives. These can be computed using the implementations from this book. The zero-crossings are then computed for each discrete volume. This results in a new discrete volume. For each voxel is established whether the values at the vertices give rise to zero-crossings within the voxel. In this case the zero-crossing surface patch is calculated through interpolation of the vertex values and possibly the values in the neighboring voxels vertices for a higher order interpolation.

The order of interpolation determines the order of continuousness across the voxel borders. These surface patches for the individual voxels represent the hypersurface that corresponds to the zero-crossing of the differential operator. The equations are directly represented by this data volume. The inequalities must also represent with part of the voxels are "within" and "outside" the set that solves the inequality.

d) Combine the equations and inequalities through intersections of the discrete representations of the zero-crossing hypersurfaces. This can be done voxel-wise.
e) If the problem was originally formulated in terms of equations the result is a discrete volume that represents a hypersurface. If the problem formulation includes inequalities the resulting data volume possibly contains a hypersurface or a set enclosed by a hypersurface.

A simple example could be detection of maxima strings in scale-space for a 2D image:
a+b) Maxima are singularities with negative second order structure. There is an inequality for each of the principal curvature directions.
c) The zero-crossings as discrete volumes arising from these equations and inequalities are calculated. For each spatial derivative equation we get a surface in scale-space. For each principal curvature direction the scale-space volume is split in two where the division is a surface in scale-space.
d) The two surfaces corresponding to zero-crossings for the spatial derivatives are intersected resulting in curve segments. The is done voxel-wise. The curve segment for a voxel will intersect the voxel in two points located on the faces of the voxel. These curve segments are intersected with the surfaces enclosing the sets determined by the signs for the second order structure. For each line segment that crosses a surface, only the part within the corresponding set is kept. Now, some curve segments will intersect the faces of their voxel zero or one time only. The other end of the line segment will be inside the voxel.
e) The result is a data volume where the singularity strings are explicitly represented as continuous line segments. Notice that the strings are directly available - no linking is necessary.

The thoughts above are not to be considered as a new unifying approach. It is merely a statement saying that such a method would be nice to have in the deep structure toolbox.

The approach sketched above has a major drawback - it is quite cumbersome to implement. However, it is geometrically simple and topologically consistent. The generality enables applications beyond detection of singularities and catastrophes.

Points, lines and surfaces are located correctly within the precision of a pixel. The precision inside a pixel is determined by the interpolation. Simple linear interpolation will do for a start - more sophisticated interpolations with a theoretically well-founded image model will provide better precision.

Such a geometry toolkit, in combination with the implementations of the Gaussian derivatives from this book, would certainly make the investigation of deep structure simpler.

▲ Task 15.4 Implement a scheme for linking singularity strings that take advantage of the drift velocity vector for determining a sensible location and shape for the search region. Possibly refine the drift velocity vector to handle singularity strings that a horizontal in scale-space (at catastrophe points). See [Kuijper1999] for inspiration.

▲ Task 15.5 Implement the deep structure geometry toolbox sketched above and extend it with function for visualizing the data. And please send a copy to the authors of this chapter ☺.

15.4 From deep structure to global structure

The previous sections have introduced a number of methods for analyzing the deep structure of images. For a specific application, such an analysis could result in a comprehensive description of the images in terms of singularity strings and catastrophe points for a number of differential operators.
Such an image description can be considered a simplified and possibly sparse representation of the original image where certain features have been selected as the salient ones. However, it is not only a simplification - it is also a specification. Where the original image contains implicit information, this image description contain explicit information about these features.

An example of such a description is the *primal sketch* [Lindeberg1994a]. Here analysis of the blobs in scale-space provides the description of the image.

The purpose of this final section of this chapter is to outline a number of applications that could take advantage of such an image description.

15.4.1 Image representations

In [Johansen1986] it was showed that the toppoints (the fold catastrophe points) provide a complete description of a 1D signal. Even though reconstruction of the original image from this description is complicated, it does however give an indication of the descriptive power of the deep structure elements.

The purpose of an image description based on the singularity strings and catastrophe points in scale-space for a set of differential operators will in general not be to offer a complete description from which the original image can be exactly reconstructed. The focus is rather to accomplish a simplified description that capture the *qualitative* properties of the image. The desired qualitative properties are defined by the application task of the image representation.

The field of information theory is central in the selection of the relevant differential operators. The set should be large enough to represent the image with sufficient precision. However, in order to provide a good simplification, the set should be minimized. The

applications all rely on a powerful simplification. A way to formalize this is the *minimum description length* (MDL) principle [Barron1998, Rissanen1978] a statistical inference principle. It says that among various possible stochastic models (or model classes) for a data sequence, one should select the model that yields the shortest code, taking into account also the bits needed to describe the model (model class) that has been used for the encoding.

Therefore it is central to establish the minimal set of differential descriptors that allows representation of the information-bearing part of the image.

This could be defined in terms of mathematics or from a more "soft" approach where an image representation is measured in terms of the ability to reconstruct an image that corresponds well with the human perception of the original image. An obvious application for such an image representation is image compression.

Below, a number of other application areas are listed that can take advantage of image representations based on deep structure analysis.

15.4.2 Hierarchical pre-segmentation

Kuijper et al. [Kuijper2001a] discovered that the scale-space surface through a so-called *scale-space saddle*, defined by the simultaneous vanishing of the gradients in the spatial *and* the scale direction (thus of the vanishing of the Laplacian), lead to a remarkable pre-segmentation of the image without prior knowledge. In 2D images the saddle points are the points where the characteristic isophotes cross (the only isophotes that intersect themselves go through saddlepoints). They form in this way natural 'separatrices' of image structure. This works in a similar fashion in the deep structure: the scale-space saddlepoints are the only points where the 'hulls' of iso-intensity surfaces in scale-space touch each other. This shows the surface through the saddle of a scale-space *germ* (for details see [Kuijper2002b]and [Florack2000b]), using the interactive OpenGL 3D viewer for *Mathematica* (see phong.informatik.uni-leipzig.de/~kuska/mathgl3dv3/):

```
<< MathGL3d`OpenGLViewer`;
α = 1; f = x^3 + 6 x t + α (2 t + y^2) ;
```

The saddle occurs for $\{x, y, t\} = \{-\frac{1}{3}, 0, -\frac{1}{18}\}$ with intensity $-\frac{1}{27}$:

```
sol = Solve[{∂x f == 0, ∂y f == 0, ∂t f == 0}, {x, y, t}] // Flatten
saddleintensity = f /. sol
```

$$\{y \to 0,\ t \to -\frac{1}{18},\ x \to -\frac{1}{3}\}$$

$$-\frac{1}{27}$$

```
MVClear[]; α = .5;
MVContourPlot3D[f, {x, -4, 4}, {y, -4, 4}, {t, -3, 3},
  Contours → {N[saddleintensity]}, PlotPoints → 50, ImageSize -> 150];
```

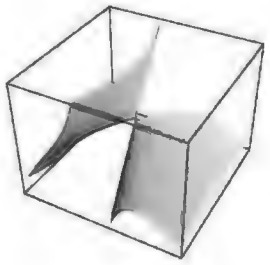


Figure 15.9 Iso-intensity contours for a scale-space saddle. In a scale-space saddle the gradient and Laplacian are zero. They form the only points in deep structure where isophote surfaces touch.

15.4.3 Perceptual grouping

Points and regions in an image provide local contributions to the overall information of the image. The human visual system does an excellent job of combining these local input into a comprehensive scene description [Elder1998]. The task of combining local information indicators into a consistent non-local description is called perceptual grouping.

An example is the task of grouping the individual pixels into regions that correspond to the object in the image. Another example is the task of connecting local edge segments into a complete edge contour.

The deep structure provides a tool for transforming the local features into non-local entities. The signature of a signal gives a simple illustration of this (see figure 13.6). Whenever two singularity strings meet in a fold catastrophe point, a non-local entity is formed - an *edge pair*. Each positive edge is associated with a negative edge. The two singularity strings define a region enclosed by the edge pair in the original image. The scale at which the edge pair are annihilated gives an upper bound for the scale of the structures within the enclosed region. An edge is essentially an local feature. However, when the deep structure of the edge is studied it offers non-local information.

Another example from this chapter is the grouping of the watershed catchment basins defined by the deep structure of the gradient magnitude.

Perceptual grouping from deep structure has been pursued by a number of researchers. The blob inclusion hierarchy of the primal sketch defines multi-scale grouping [Lindeberg1994a]. Grouping and detection of elongated structures can also be achieved through analysis of the deep structure. An example of this is [Staal1999a].

15.4.4 Matching and registration

Matching is a difficult problem in image analysis. Traditional methods are based on templates and deformable models where the correlation between an object prototype and subwindows of the image is explored. The key point is that the matching is done between an image and a prototype object image. These methods are inherently fragile to changes in the lighting conditions and the viewing positions.

A deep structure based matching algorithm would instead do the matching between the deep structure representations of the image. Since these representations are qualitative simplifications, matching is theoretically easier. The algorithmic basis is possibly matching of the deep structure singularity strings or even simpler matching of the catastrophe points. This matching should probably be done in a coarse to fine order where the necessary global and local deformations needed in order to achieve a match could be recorded. The matching would then result in not only a measure of the "distance" between the images but also a description of the transformations needed to map one image into the other. See also recent results with shape-context operators [Belongie2002].

Algorithms for matching deep structure descriptions could be inspired by the research performed within the field of *graph theory* [Shokoufandeh2002, Dickinson1999]. Interesting and promising work has been done with so-called *shocks*, singularities in curve evolutions [Siddiqi1999a], which concepts are ready to be transferred to scale-space singularities and catastrophes.

A number of applications have used approaches similar to the outline above. In [Bretzner1999a] the location and scale-range of features in scale-space is used as the qualitative model for matching in a tracking algorithm. This approach gives an algorithm which is quite robust with respect to change of view.

Registration is a task quite similar to matching. In registration the match is known beforehand. The desired result is a mapping that links points (or feature points) between the two images. As described above this mapping can also be achieved from the deep structure description matching. A coarse to fine matching on structural information is applied in [Massey1999] illustrated by registration of coastlines. The registration is robust with respect to translation, rotation and scale differences between the images.

15.4.5 Image databases

In medical imaging the rise of the PACS (Picture Archive Computer System) has created large image databases. This creates a natural need for algorithms that allow searching in these databases. In principle, a matching algorithm allows searching. The obvious problem is that explicit matching against every single image in a large database containing tens of thousands of images would not be practically feasible.

Regular databases have indexes which allow search for specific keywords. Image databases need the same concept where the search is defined in terms of a combination of keywords and image data. This is by no means a trivial problem.

Indexing implies a sorting of the descriptions. An image description could be sorted based on the coarse to fine hierarchy of the deep structure description and on the differential features used in the description. However, the specific sorting is not obvious.

A further complication is that the searching will often not seek matching of complete images. For many applications the image data used in the search is only a small part of the desired image.

Indexing based on the coarse to fine approach would possibly allow the search algorithm to enter the search space directly at the proper scale corresponding to the size of the desired image object. The problem can then be stated as the matching of a subgraph within a larger graph.

15.4.6 Image understanding

The listing of applications for a deep structure based image description is by no means exhaustive. It is only intended as a short appetizer.

The wide variety of basic image analysis problems that can be addressed suggests that future research within deep structure based image descriptions could have a profound impact on image analysis in the years to come. Both as a theoretical basis for image understanding and as the foundation for real world applications.

However, a lot of work needs to be done. A broad impact of deep structure based methods requires:
- General methods for establishing the optimal differential descriptors.
- Clarification on which deep structure descriptors in terms of properties for singularities and catastrophes that should be used in the image representation.
- Development of algorithms for matching, indexing and searching these deep structure descriptions.
- Simple, intuitive formulations and applications of the theoretical achievements in order to make them available to the general computer vision and image analysis community.

15.5 Summary of this chapter

Interesting results from vectorfield analysis can be applied to the analysis of singularities in scale-space. We discuss the *homotopy number*, which is also called the *winding number* for 2D and 3D images. A vector (e.g. the gradient) does make no rotation when a closed contour path is followed around a regular point, but it rotates once around a maximum or minimum, and rotates once backwards around a saddle point. The number of rotations (over 2π) is defined as the winding number. For 3D images this is equivalent to the number of complete space angles (over 4π).

The singularities' winding numbers add up when contained together within the closed path. The vector of its velocity through scale-space can be calculated.

The number of singularities decreases exponentially with scale, with the coefficient of the log-log slope (Hausdorff dimension) roughly equivalent to minus the spatial dimension.

Catastrophe theory is a mature field describing the events that can occur while evolving the image. Extrema always annihilate with saddle points, in dimensions higher then one creations can occur generically. A new field is emerging, where the topological, hierarchical image structure can be exploited. The study of scale-space singularities and catastrophes is the beginning hereof. The development of a 'deep structure toolbox' is an emerging and promising field in fundamental multi-scale computer vision research.

Such a hierarchical representation is the natural input for the application of graph theory for the study of topological relations. The toppoint hierarchical tree might be investigated for atlas-supported segmentation, object search in images and image content based search and retrieval (e.g. $x - y - \sigma$ triples as input for active shape models), as well as *logical filtering*, i.e. the removal of a subtree to remove a structural element out of an image (e.g. trainee radiologists learn in an early stage at X-thorax reading: "think away the ribs").

Not only the detection of scale-space toppoints is important, the reconstruction from the toppoints into an image again is just as important. Recently it has been shown that such a reconstruction is feasible [Nielsen2001a].

16. Deblurring Gaussian blur

16.1 Deblurring

To discuss an application where really high order Gaussian derivatives are applied, we study the deblurring of Gaussian blur by inverting the action of the diffusion equation, as originally described by Florack et al. [Florack et al. 1994b, TerHaarRomeny1994a].

Gaussian degradation, as deblurring with a Gaussian kernel is also coined, occurs in a large number of situations. E.g.: the point-spread-function of the human lens e.g. has a close to Gaussian shape (for a 3 mm pupil its standard deviation is about 2 minutes of arc); the atmospheric turbulence blurs astronomical images in a Gaussian fashion; and the thickness profile of thin slices made by modern spiral-CT scanners is about Gaussian, leading to Gaussian blur in a multiplanar reconstructed image such as in the sagittal or coronal plane. Surely, deblurring is of immediate importance for image restoration.

Due to the central limit theorem, stating that a concatenation of any type of transformation gives a Gaussian shape when the number of sequential transformations goes to infinity, many physical processes involving sequential local degradations show a close-to-Gaussian blurring.

There is an analytical solution for the inversion of Gaussian blur. But the reconstruction can never be exact. Many practical solutions have been proposed, involving a variety of enhancing filters (e.g. high-pass or Wiener) and Fourier methods. Analytical methods have been proposed by Kimia, Hummel and Zucker [Kimia1986, Kimia1993, Hummel1987] as well as Réti [Réti 1995a]. They replaced the Gaussian blur kernel by a highly structured Toeplitz matrix and deblurred the image by the analytical inverse of this matrix. Martens deblurred images with polynomial transforms [Martens 1990].

16.2 Deblurring with a scale-space approach

If we consider the stack of images in the scale-space, we see the images gradually blur when we increase the scale. Indeed, the diffusion equation $\frac{\partial L}{\partial t} = \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$ tells us that the change ∂L in L when we increase the scale t with a small increment ∂t is equal to the local value of the Laplacian $\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}$. From the early chapters we remember that a scale-space is infinitely differentiable due to the regularization properties of the observation process.

A natural step is to look what happens if we go to negative scales. Due to the continuity we are allowed to construct a Taylor expansion of the scale-space in any direction, including the negative scale direction. We create a Taylor series expansion of our scale-space $L(x, y, t)$ with *Mathematica*'s command **Series**, e.g. to third order around the point $t = 0$:

```
<< FrontEndVision`FEV`;
```

```
L =.; Series[L[x, y, t], {t, 0, 3}]
```

$$L[x, y, 0] + L^{(0,0,1)}[x, y, 0]\, t + \frac{1}{2} L^{(0,0,2)}[x, y, 0]\, t^2 + \frac{1}{6} L^{(0,0,3)}[x, y, 0]\, t^3 + O[t]^4$$

The derivatives to t are recognized as e.g. $L^{(0,0,1)}$. It is not possible to directly calculate the derivatives to t. But here the diffusion equation rescues us. We can replace the derivative of the image to scale with the Laplacian of the image, and that can be computed by application of the Gaussian derivatives on the image. Higher order derivatives to t have to be replaced with the repeated Laplacian operator. E.g., the second order derivative to t has to be replaced by the Laplacian of the Laplacian. To shorten our notations, we define Δ to be the Laplacian operator:

```
Δ := (∂x,x # + ∂y,y #) &
```

Here the construct of a 'pure function' in *Mathematica* is used: e.g. `(#^3) &` is a function without name that raises its argument to the third power. The repeated Laplacian operator is made with the function `Nest`:

```
Nest[f, x, 3]
```

```
f[f[f[x]]]
```

We now look for each occurrence of a derivative to t. This is the term `L^(0,0,n_) [x, y, 0]` where `n_` is anything, named `n`, the order of differentiation to t. The underscore `_` or `Blank[]` is the *Mathematica* representation for any single expression). With *Mathematica*'s powerful technique of *pattern matching* (`/.` is the `Replace` operator) we replace each occurrence of `L^(0,0,n_) [x, y, 0]` with an `n`-times-nested Laplacian operator as follows:

```
expr = Normal[Series[L[x, y, t], {t, 0, 3}]] /.
  L^(0,0,n_) [x, y, 0] :> Nest[Δ, L[x, y, 0], n]
```

$$L[x, y, 0] + t\,(L^{(0,2,0)}[x, y, 0] + L^{(2,0,0)}[x, y, 0]) + \frac{1}{2} t^2\,(L^{(0,4,0)}[x, y, 0] + 2\,L^{(2,2,0)}[x, y, 0] + L^{(4,0,0)}[x, y, 0]) + \frac{1}{6} t^3\,(L^{(0,6,0)}[x, y, 0] + 3\,L^{(2,4,0)}[x, y, 0] + 3\,L^{(4,2,0)}[x, y, 0] + L^{(6,0,0)}[x, y, 0])$$

To get the formulas better readable we apply the function `shortnotation` (defined in chapter 6, section 5), which replaces the formal notations of the derivatives by a shortform expressed in a (luminance) function `L` with appropriate subscripts through *pattern matching*:

```
expr // shortnotation
```

$$L[x, y, 0] + t\,(L_{xx} + L_{yy}) + \frac{1}{2} t^2\,(L_{xxxx} + 2\,L_{xxyy} + L_{yyyy}) + \frac{1}{6} t^3\,(L_{xxxxxx} + 3\,L_{xxxxyy} + 3\,L_{xxyyyy} + L_{yyyyyy})$$

High order of spatial derivatives appear. The highest order in this example is 6, because we applied the Laplacian operator 3 times, which itself is a second order operator. With *Mathematica* we now have the machinery to make Taylor expansions to any order, e.g. to 5:

```
expr = Normal[Series[L[x, y, t], {t, 0, 5}]] /.
  L^(0,0,n_) [x, y, 0] :> Nest[Δ, L[x, y, 0], n];
expr // shortnotation
```

$$L[x, y, 0] + t\,(L_{xx} + L_{yy}) + \frac{1}{2}\,t^2\,(L_{xxxx} + 2\,L_{xxyy} + L_{yyyy}) + \frac{1}{6}\,t^3\,(L_{xxxxxx} + 3\,L_{xxxxyy} + 3\,L_{xxyyyy} + L_{yyyyyy}) + \frac{1}{24}\,t^4\,(L_{xxxxxxxx} + 4\,L_{xxxxxxyy} + 6\,L_{xxxxyyyy} + 4\,L_{xxyyyyyy} + L_{yyyyyyyy}) + \frac{1}{120}\,t^5\,(L_{xxxxxxxxxx} + 5\,L_{xxxxxxxxyy} + 10\,L_{xxxxxxyyyy} + 10\,L_{xxxxyyyyyy} + 5\,L_{xxyyyyyyyy} + L_{yyyyyyyyyy})$$

No matter how high the order of differentiation, the derivatives can be calculated using the multi-scale Gaussian derivatives. So, as a final step, we replace by pattern matching (**/.**) the spatial derivatives in the formula above by Gaussian derivatives (**HoldForm** assures we see just the formula for **gD[]**, of which evaluation is 'hold'; **ReleaseHold** removes the hold):

```
corr =
 expr /. Derivative[n_, m_, 0][L][x, y, a_] → HoldForm[gD[im, n, m, 1]]
```

$$t\,(gD[im, 0, 2, 1] + gD[im, 2, 0, 1]) + \frac{1}{2}\,t^2\,(gD[im, 0, 4, 1] + 2\,gD[im, 2, 2, 1] + gD[im, 4, 0, 1]) + \frac{1}{6}\,t^3\,(gD[im, 0, 6, 1] + 3\,gD[im, 2, 4, 1] + 3\,gD[im, 4, 2, 1] + gD[im, 6, 0, 1]) + \frac{1}{24}\,t^4\,(gD[im, 0, 8, 1] + 4\,gD[im, 2, 6, 1] + 6\,gD[im, 4, 4, 1] + 4\,gD[im, 6, 2, 1] + gD[im, 8, 0, 1]) + \frac{1}{120}\,t^5\,(gD[im, 0, 10, 1] + 5\,gD[im, 2, 8, 1] + 10\,gD[im, 4, 6, 1] + 10\,gD[im, 6, 4, 1] + 5\,gD[im, 8, 2, 1] + gD[im, 10, 0, 1]) + L[x, y, 0]$$

Because we *de*blur, we take for $t = \frac{1}{2}\,\sigma^2$ a negative value, given by the estimated amount of blurring σ_{est} we expect we have to deblur. However, applying Gaussian derivatives on our image increases the inner scale with the scale of the applied operator, i.e. blurs it a little necessarily. So, if we calculate our repeated Laplacians say at scale $\sigma_{\text{operator}} = 4$, we need to deblur the effect of both blurrings. Expressed in t, the total deblurring 'distance' amounts to $t_{\text{deblur}} = -\frac{\sigma^2_{\text{est}} + \sigma^2_{\text{operator}}}{2}$.
We assemble our commands in a single deblurring command which calculates the amount of correction to be added to an image to deblur it:

```
deblur[im_, σest_, order_, σ_] := Module[{expr}, Δ = ∂x,x # + ∂y,y # &;
  expr = Normal[Series[L[x, y, t], {t, 0, order}]] /.
     L_^(0,0,1_) [x_, y_, t_] :> Nest[Δ, L[x, y, t], 1] /. t → -(σest^2 + σ^2)/2;
  Drop[expr, 1] /. L^(n_,m_,0) [x, y, t_] → HoldForm[gD[im, n, m, σ]]]
```

and test it, e.g. for first order:

```
im =.; deblur[im, 2, 1, σ]
```

$$\frac{1}{2}\,(-4 - \sigma^2)\,(gD[im, 0, 2, \sigma] + gD[im, 2, 0, \sigma])$$

It is a well known fact in image processing that subtraction of the Laplacian (times some constant depending on the blur) sharpens the image. We see here that this is nothing else than the first order result of our deblurring approach using scale-space theory. For higher order deblurring the formulas get more complicated and higher derivatives are involved:

```
deblur[im, 2, 3, σ]
```

$$\frac{1}{2}(-4-\sigma^2)\,(gD[im,0,2,\sigma]+gD[im,2,0,\sigma])+\frac{1}{8}(-4-\sigma^2)^2$$
$$(gD[im,0,4,\sigma]+2\,gD[im,2,2,\sigma]+gD[im,4,0,\sigma])+\frac{1}{48}(-4-\sigma^2)^3$$
$$(gD[im,0,6,\sigma]+3\,gD[im,2,4,\sigma]+3\,gD[im,4,2,\sigma]+gD[im,6,0,\sigma])$$

We generate a test image blurred with $\sigma = 2$ pixels:

```
im = Import["mr128.gif"][[1, 1]]; DisplayTogetherArray[
 ListDensityPlot /@ {im, blur = gDf[im, 0, 0, 2]}, ImageSize -> 360];
```

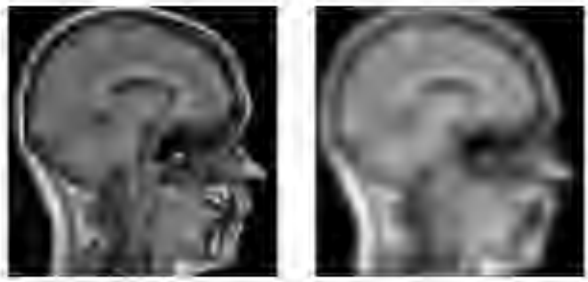


Figure 16.1 Input image for deblurring, blurred at $\sigma = 2$ pixels., resolution 128^2.

We try a deblurring for orders 4, 8, 16 and 32 (fig. 16.2 next page): A good result. Compare with figure 16.1. *Mathematica* is reasonably fast: the deblurring to 32^{nd} order involved derivatives up to order 64 (!), in a polynomial containing 560 calls to the **gD** derivative function.

The 4 calculations take together somewhat more than one minute for a 128^2 image on a 1.7 GHz 512 MB Pentium 4 under Windows 2000 (the 32^{nd} order case took 50 seconds). This counts the occurrences of **gD** in the 32^{nd} order deblur polynomial, i.e. how many actual convolutions were needed:

```
dummy =.; Length[Position[deblur[dummy, 2, 32, 4], gD]]
```

```
560
```

```
Remove[p];
p[i_] := ListDensityPlot[blur + ReleaseHold[deblur[blur, 2, i, 4]],
  PlotLabel → "order = " <> ToString[i]];
DisplayTogetherArray[{{p[4], p[8]}, {p[16], p[32]}}, ImageSize -> 450];
```

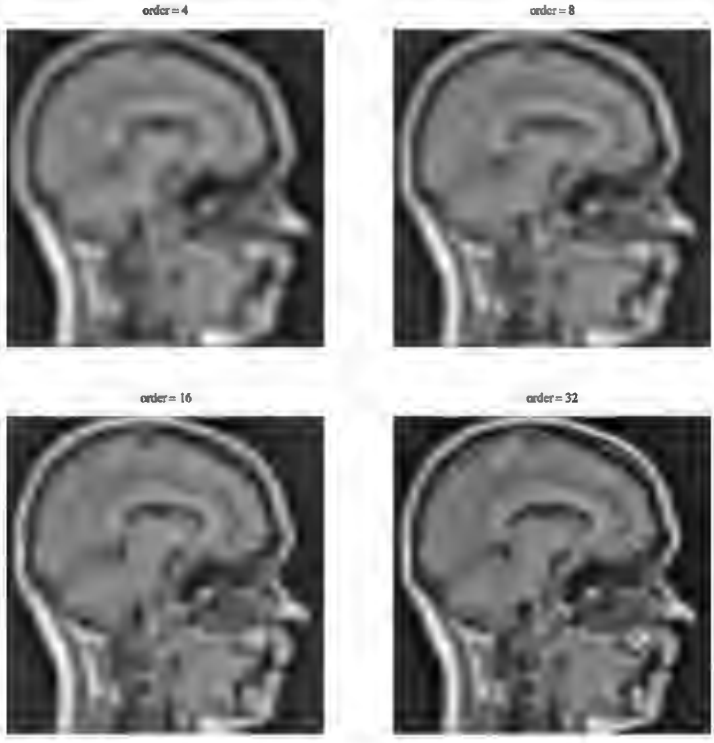


Figure 16.2 Deblurring of a blurred image (128^2 pixels, $\sigma_{blur} = 2$ pixels, left column) with different orders of approximation. The 32^{nd} order (bottom right) result comes close to the original (figure 16.1, left).

16.3 Less accurate representation, noise and holes

The method is reasonably robust to the accuracy or representation of the data. Of course, it is essential to retain as much information as possible during the blurring process. Close to precise representation (as high precision real floating point numbers) was the case in the above example. When we store the image to disk as a typical unsigned byte per pixel representation, we throw away much information. We can study the effect of such round-off by rounding each pixelvalue of the blurred image (making them integers), and do the same deblurring again:

```
roundedblur = Round[blur]; Block[{$DisplayFunction = Identity},
 p = Table[corr = deblur[roundedblur, 2, 2^i, 4] // ReleaseHold;
   ListDensityPlot[roundedblur + corr,
    PlotLabel -> "order = " <> ToString[2^i]], {i, 2, 5}]];
Show[GraphicsArray[Partition[p, 2]], ImageSize -> 450];
```

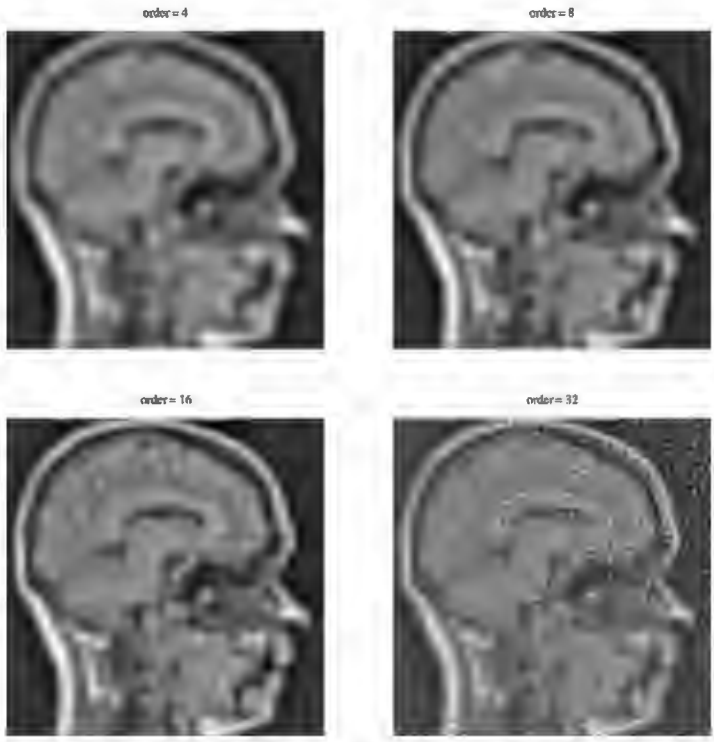


Figure 16.3 Deblurring results when the blurred image is stored as integers (intensity range of this particular image is [2-186]). Note that only the deblur results are shown. The deblurring order is indicated with each result.

Clearly the deblurring now fails for the very high order, but the results are still good till 16^{th} order.

Noise is a disaster. When we add Gaussian distributed noise with zero mean and a standard deviation of 5 intensity units, we get the following results:

```
<< Statistics`ContinuousDistributions`;
noisyblur = blur + Table[Random[NormalDistribution[0, 5]], {128}, {128}];
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[noisyblur, PlotLabel -> "noisyblur"];
  p2 = Table[corr = deblur[noisyblur, 2, 2^i, 4] // ReleaseHold;
    ListDensityPlot[noisyblur + corr,
     PlotLabel -> "order = " <> ToString[2^i]], {i, 2, 4}]];
Show[GraphicsArray[Prepend[p2, p1]], ImageSize -> 470];
```

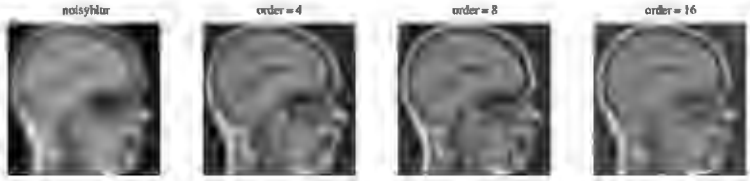


Figure 16.4 Deblurring results when the blurred image is disturbed by Gaussian additive noise (mean = 0, $\sigma_{\text{intensity}} = 5$). The deblurring order is indicated with each result.

And to conclude, we study the effect of 25 random pixels being 'blanked out', i.e. set to zero:

```
coords = Table[Random[Integer, {1, 128}], {50}, {2}];
holesblur = ReplacePart[blur, 0, coords];
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[holesblur];
  p2 = Table[corr = deblur[holesblur, 2, 2^i, 4] // ReleaseHold;
    ListDensityPlot[holesblur + corr,
     PlotLabel -> "order = " <> ToString[2^i]], {i, 2, 4}]];
Show[GraphicsArray[Prepend[p2, p1]], ImageSize -> 470];
```

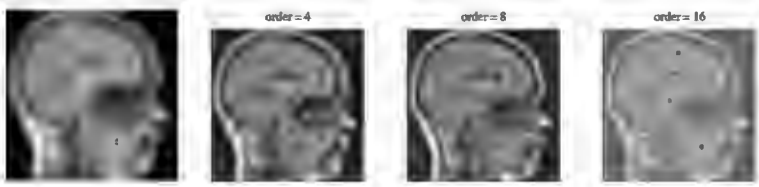


Figure 16.5 Deblurring results when the blurred image is disturbed by setting a random selection of 25 pixels to zero. The deblurring order is indicated with each result. Note that up to order 8 the 'blanked' points reconstruct well. At order 16 an overshoot occurs.

▲ Task 16.1 Experiment with deblurring images that are blurred with another kernel than the Gaussian.

▲ Task 16.2 Experiment with blurred images from an external source, e.g. find unsharp speed ticket camera images on the internet, digitize your unsharp home pictures, etc.

▲ Task 16.3 Motion blur may be simulated with anisotropic Gaussian blur, i.e. where the σ is rather different for the x and y direction. It may also be at any angle (see also chapter 19 where we discuss Gaussian kernels at arbitrary directions). Make such a blurred test image, and come up with a deblurring scheme for it.

▲ Task 16.4 In chapter 21 we discuss nonlinear diffusion equations. After having studied this chapter, it is interesting to consider how these nonlinear diffusion equations might be applied in the framework presented in this chapter, and what is the type of degradation .

16.4 Summary of this chapter

The regularization property of the Gaussian kernel makes the scale-space continuous, which means infinitely differentiable in both the spatial as the scale domain. It was proposed by Florack to expand the scale-space of a blurred image into the negative scale direction by means of a Taylor expansion. The high order derivatives to scale in this expansion can be expressed in spatial Laplacians of the image, due to the constraint of the isotropic diffusion equation. *Mathematica* turns out to be an efficient tool to do the analytic calculations of the high order Taylor expansion polynomial, in which the derivatives can be replaced by scaled Gaussian derivatives. We show some examples to real high order.

Deblurring is instable, and can only be carried out analytically when no data is lost, for example through finite intensity representation (8 bit), noise of other pixel errors. The message of this chapter is that the taking of very high order derivatives is feasible, that computer algebra is a suitable mean for implementing these calculations, and gives an example of deblurring from Gaussian blur.

17. Multi-scale optic flow

Bart ter Haar Romeny, Luc Florack, Avan Suinesiaputra

17.1 Introduction

In this chapter we focus on the quantitative extraction of small differences in an image sequence caused by motion, and in an image pair by differences in depth. We like to extract the local motion parameters as a small local shift over time or space. We call the resulting vectorfield the *optic flow* from the image sequence, a spatio-temporal feature, and we call the resulting vectorfield the *disparity map* for the stereo pair. As the application of the method described in this chapter is virtually the same for stereo disparity extraction, we will focus in the treatment on spatio-temporal optic flow.

```
<< FrontEndVision`FEV`
PlotVectorField3D[{y, -x, Sin[z]}, {x, -1, 1},
  {y, -1, 1}, {z, 0, 2}, VectorHeads → True, ImageSize -> 270];
```

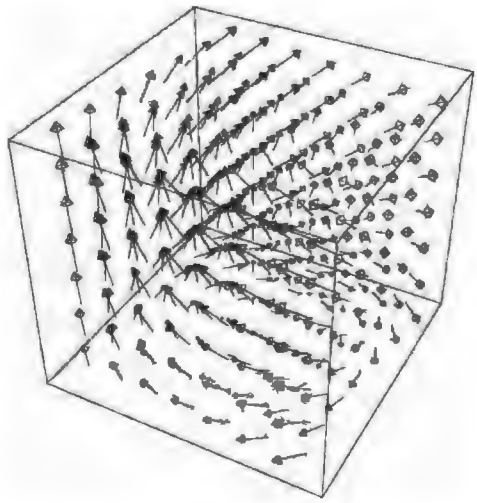


Figure 17.1 An example of a 3D (optic) flowfield. Such flowfields are used in the study of 3D motion of e.g. the heart from 3D-time magnetic resonance imaging (MRI) sequences.

We need to measure a displacement of something over some distance, in some amount of time, in some direction in order to find the vector starting at each point in the image that indicates where this point is moving to. Of course, we need to take into consideration that we are doing a physical measurement, so we need to apply the scale-space paradigm, and secondly we need to consider how constant our 'structure' is when it moves.

Many approaches have been proposed to solve the problem of finding the optic flow field of an image sequence. Three major classes of optic flow computation techniques can discriminated (see for a good overview Beauchemin and Barron [Beauchemin1995]):
- gradient based (or differential) methods;
- phase-based (or frequency domain) methods;
- correlation-based (or area) methods;
- feature-point (or sparse data) tracking methods;

In this chapter we compute the optic flow as a dense optic flow field with a multi-scale differential method. The method, originally proposed by Florack and Nielsen [Florack1998a] is known as the Multiscale Optic Flow Constrain Equation (MOFCE). This is a scale-space version of the well known computer vision implementation of the optic flow constraint equation, as originally proposed by Horn and Schunck [Horn1981]. This scale-space variation, as usual, consists of the introduction of the aperture of the observation in the process. The application to stereo has been described by Maas et al. [Maas1995a, Maas1996a].

Of course, difficulties arise when structure emerges or disappears, such as with occlusion, cloud formation etc. Then knowledge is needed about the processes and objects involved. In this chapter we focus on the scale-space approach to the *local measurement* of optic flow, as we may expect the visual front-end to do.

17.2 Motion detection with pairs of receptive fields

As a biologically motivated start, we begin with discussing some neurophysiological findings in the visual system with respect to motion detection. A popular model, based on physiological data involving spatiotemporal receptive fields, is the *Reichardt detector*, which models temporally and spatially coupled *pairs* of receptive fields.

```
pt = zero = Table[50, {64}, {64}]; pt[[32, 32]] = 100;
rf = gD[pt, 2, 0, 10] + gD[pt, 0, 2, 10]; rf = rf - Min[rf];
Block[{$DisplayFunction = Identity, xres, yres, max},
  {yres, xres} = Dimensions[rf]; max = Max[rf];
  rfleft = Graphics3D[
    ListPlot3D[zero, Map[GrayLevel, rf / max, {2}], Mesh -> False]];
  rfright = TranslateShape[rfleft, {75, 0, 0}];
  cube = Graphics3D[Cuboid[{59, 22, 10}, {79, 42, 30}]];
  sphere =
   TranslateShape[Graphics3D[Sphere[13, 25, 20]], {107, 32, 20}];
  rightarrow = Graphics3D[{Thickness[.01],
     arrow3D[{107, 32, 20}, {30, 0, 0}, True]}];
  lines = Graphics3D[{Thickness[.01],
     Line[{{32, 32, 50}, {32, 32, 20}, {107, 32, 20}, {107, 32, 50}}]}];];
```

```
Show[{rfleft, rfright, cube, sphere, lines, rightarrow},
  ViewPoint → {0.4, -5, .8},
  DisplayFunction → $DisplayFunction, Boxed → False];
```

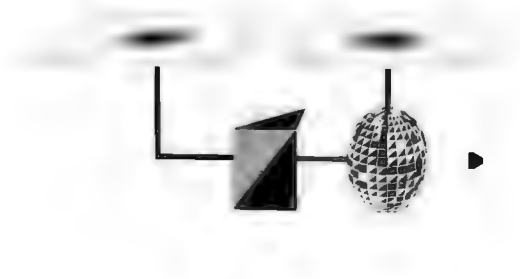


Figure 17.2 A simple model for a retinal Reichardt detector for the detection of motion. Two center-surround receptive fields are separated by a center-to-center span of length d. These receptive fields both project to the (spherical) output ganglion cell, the right one directly, the left one through an intermediate cell (depicted as a cube) which incorporates a small temporal delay τ. The ganglion cell has the highest chance of firing an action potential when the velocity of the object $v = d/\tau$.

When motion needs to be detected directly, the displacement over space in a given amount of time has to be measured. One of the strategies of the front-end visual system seems to design a detector for every occurrence of a stimulus parameter. For the detection of motion a famous proposal was done by prof. Werner Reichardt in the late fifties during his studies on fly motion detection. He proposed a simple but very effective correlation type model consisting of a *velocity and directionally tuned pair* of receptive fields. Two center-surround receptive fields in a single eye of the same size, separated by a distance (*span*) d project to the same ganglion cell. The first cell projects through an intermediate cell that introduces a small temporal delay τ, the second receptive field projects directly to the ganglion. The input synapses on the ganglion cell create excitatory post-synaptic potentials (EPSPs), small intracellular increases of the intracellular voltage with a short duration (some ms). EPSPs that arrive simultaneously superimpose and this summed potential gives a higher chance to reach the intracellular threshold voltage that leads to an action potential than a single EPSP alone. A ganglion cell here (as any neuron) thus acts as a *temporal coincidence detector*. If an object moving with velocity v passes over the two receptive fields such that the inputs to the ganglion cell arrive simultaneously we get optimal detection if $v = \frac{d}{\tau}$. We call v the tuning velocity of the receptive field pair. Figure 17.2 shows the model.

The cell pair is tuned to its characteristic velocity *and* its characteristic direction only. This means we need an abundance of such pairs to have tuning for all possible velocities and directions. There are strong indications we have indeed enormous amounts of such receptor pairs tuned for a wide perceptual range of velocities and directions. These motion sensitive cells are coined the *on-off direction selective ganglion cells* by Rodieck [Rodieck1998 pp. 319].

The motion selective ganglion cells are the parasol ganglion cells. They are the larger type of ganglion cells, and project primarily to the magnocellular layers of the LGN and to the superior colliculus (having a role in the control of gaze stabilization by eye movements). The

motion selective ganglion cells have a bistratified dendritic field that stretches out into both the on- and off-sublayers of the inner synaptic layer in the retina [Amthor1989]. It is not (yet) clear what the physiological mechanism of the delay cell might be. A likely candidate is the amacrine cell. This cell is located at the proper location (layer) in the retina, often very close to parasol cells, it has only connections to parasol ganglion cells (see figure 17.3) and it comes in 30 to 40 varieties (to accommodate for the range of delays?). In the rabbit the on-off direction selective cells are aligned with the eye muscles.

```
Show[Import["parasol amacrine connections macaque.gif"],
  Frame -> True, FrameTicks -> False, ImageSize -> 250];
```

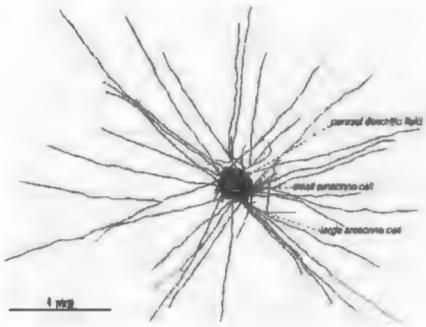


Figure 17.3 Amacrine cell processes coupled to a parasol ganglion cell. The processes were labeled after injection of the ganglion cell. From [Dacey1992], adapted from [Rodieck1998, pp. 263].

Varieties of Reichardt detectors have been proposed. E.g. one can add a delay cell between the two receptive fields in the opposite direction (in the figure 17.2 from the right to the left receptive field). This results in a bidirectional responsive motion detector and is called a *generalized* Reichardt detector. For the work of Werner Reichardt see the bibliography in http://www.kyb.tuebingen.mpg.de/re/biblio.html, and the overview papers in [Reichard1988b]. See also the extensive work by van de Grind, Koenderink and van Doorn [vandeGrind1999] and by Sperling and coworkers [Sperling1998] on physiological motion detection models.

The elegance and simplicity of the Reichardt model is appealing. Many hardware implementations exist, as e.g. CMOS chip, or analog VLSI computing device. For an interesting overview see: http://www.klab.caltech.edu/~timmer/tell96/motion_chips.html.

This set of specialized *pairs* of receptive fields forms a separate *channel* in the visual pathway for motion coding. It is again an example of the *functional separation* seen in the visual front-end. Another example of paired receptive fields is a *disparity pair* where two receptive fields, one in the left eye and one at or about at the corresponding position in the right eye, form a pair for the detection of depth and the extraction of the differential structure of depth. This implies that the theory discussed in this chapter is also applicable to the multi-scale extraction of stereo disparity and its derivatives (like slant and depth curvature). Again, we find the same ensemble *tuning*: for all disparities a dedicated set seems to be available.

17.3 Image deformation by a discrete vectorfield

As an intermezzo, we discuss the use of vector fields in the calculation of image deformation. For such a *warping*, the pixels are displaced according to a vectorfield, and the intensity of the pixels at their new location has to be calculated. This is the inverse process of the extraction of optic flow. With the understanding of warping we can easily make e.g. test images for optic flow routines, and generate warped images for *image matching*. The easiest method is to start with considering the warped image, and to measure the image intensity at the location where the pixel came from. This is often between the original pixels, so the image has to be resampled. *Mathematica* has the function **Interpolation** and for discrete data **ListInterpolation** that interpolates a function (of any dimension) to some order (default is *cubic spline interpolation*: to third order). This function comes handy for any resampling.

Here is the implementation of a warping routine for 2D images given a 2D vectorfield:

```
Unprotect[deform2D];

deform2D[im_List, vecf_List] :=
  Module[{xdim, ydim, m, imp, newx, newy},
 If[Append[Dimensions[im], 2] != Dimensions[vecf], Break[]];
 {ydim, xdim} = Dimensions[im]; m = Table[0, {ydim}, {xdim}];
 imp = ListInterpolation[im]; Do[newx = x - vecf[[x, y, 2]];
 newy = y - vecf[[x, y, 1]]; If[1 <= newx <= xdim &&
 1 <= newy <= ydim, m[[x, y]] = imp[newx, newy], m[[x, y]] = 0],
 {y, 1, ydim}, {x, 1, xdim}]; m];

? deform2D

deform2D[im, vecf] deforms a 2D image im according to a
  specified discrete vectorfield vecf with the same dimensions.
  The image is interpolated to third order with ListInterpolation.
     im = 2D input image {ydim,xdim}
     vecf = 2D discrete vectorfield {ydim,xdim,2}
```

We read a 256^2 image of a retinal fundus recording and plot the image, the vectorfield and the resulting warped image below:

```
im = Import["fundus256.gif"][[1, 1]]; {ydim, xdim} = Dimensions[im];
vecf =
  Table[-{Sin[π x/xdim], Cos[π y/ydim]} // N, {y, 1, ydim}, {x, 1, xdim}];
DisplayTogetherArray[{ListDensityPlot[im],
   PlotVectorField[-{Sin[π x/xdim], Cos[π y/ydim]},
{x, 1, xdim}, {y, 1, ydim}], ListDensityPlot[deform2D[im, 10 vecf]]},
  ImageSize → 480];
```

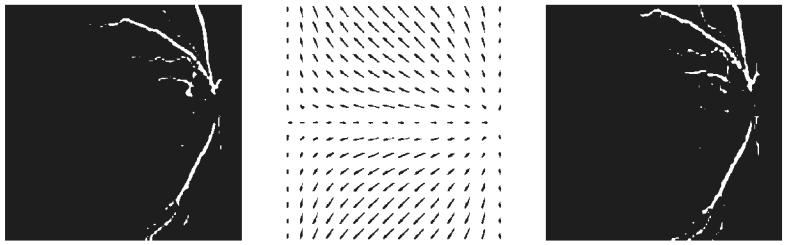


Figure 17.4 Left: Image of a retina fundus registered with a scanning laser ophthalmoscope (courtesy T. Berendschot, University Medical Center Utrecht, the Netherlands). Image resolution 256^2. Middle: the vectors of a given *warping vectorfield*. Right: warped image. The new image intensities are sampled from the (cubic spline) interpolated image at a position dislocated over the distance of the vector. To enhance the effect, the lengths of the vectors in the vectorfield are multiplied by 4.

▲ Task 17.1 Modify the warping routine so you get a function for:
- image rotation over an arbitrary angle;
- zooming in and zooming out (scaling of the image);
- affine transformation, like viewing the sheet from an oblique angle; this is often called 'perspective horizontal' or 'perspective vertical', or combinations thereof;
- spiral deformation;

17.4 The optic flow constraint equation

When we consider structure in an image, that moves with time to a new position, we need to define what we mean with 'structure'. A likely candidate is the local luminance. The classical approach to the optic flow equation was proposed by Horn [Horn1981] in his famous *optic flow constraint equation* (OFCE) for the case of a scalar image sequence, where the *total derivative* of the luminance distribution with respect to time is supposed to vanish: $\frac{dL(x,y,t)}{dt} = 0$.

Intermezzo: Total derivatives

From the *Mathematica* Help: When you find the derivative of some expression f with respect to x, you are effectively finding out how fast f changes as you vary x. Often f will depend not only on x, but also on other variables, say y and z. The results that you get then depend on how you assume that y and z vary as you change x. There are two common cases. Either y and z are assumed to stay fixed when x changes, or they are allowed to vary with x. In a standard *partial derivative* $\frac{\partial f}{\partial x}$ all variables other than x are assumed fixed. On the other hand, in the *total derivative* $\frac{df}{dt}$ all variables are allowed to change with x. In *Mathematica*, `D[f,x]` gives a partial derivative, with all other variables assumed independent of x. `Dt[f,x]` gives a total derivative, in which all variables are assumed to depend on x.

```
Dt[L[x, y, t], t] == 0

L^(0,0,1) [x, y, t] + Dt[y, t] L^(0,1,0) [x, y, t] + Dt[x, t] L^(1,0,0) [x, y, t] == 0
```

The derivatives $\frac{dx}{dt}$ and $\frac{dy}{dt}$ denote the velocity in the x-direction v^x and the velocity in the y-direction v^y respectively. We write upper indices for the dimensional component and lower indices for derivatives with respect to the dimensional component. We get: $\frac{\partial L}{\partial t} + v^x \frac{\partial L}{\partial x} + v^y \frac{\partial L}{\partial y} = 0$. This OFCE has been the basis for numerous computer vision studies into optic flow [see also Koenderink1986c, Koenderink1987d, Koenderink1992f (second order flow), and see Barron1994a for a comparison of different techniques].

There is, however, a difficulty when we start *looking at* the flow (the scale-space paradigm of observation): because we use physical apertures of finite size, the observed image is a blurred version, and the isophote landscape changes with the aperture. When the object comes closer or moves further away, the same aperture covers different areas of the object. The isophote landscape changes, as we can see in the following example where we move away from the image. The isophotes $L = 50$ of an image observed at scale $\sigma = 1$ is compared with the same isophotes observed at scale $\sigma = 2$ at a distance twice as far:

```
im = Import["mr64.gif"][[1, 1]]; {im1, im2} = gD[im, 0, 0, #] & /@ {1, 2};
DisplayTogetherArray[Show[{ListDensityPlot[im1],
   ListContourPlot[im1, Contours -> {50}, ContourStyle -> White]}],
 Show[{ListDensityPlot[im2, PlotRegion → {{0.25, 0.75}, {0.25, 0.75}}],
   ListContourPlot[im2, Contours -> {50}, ContourStyle -> White,
    PlotRegion → {{0.25, 0.75}, {0.25, 0.75}}]}], ImageSize -> 265];
```

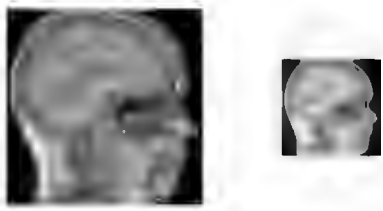


Figure 17.5 The isophote landscape of an image changes drastically when we change our aperture size. This happens when we move away or towards the scene with the same camera. Left: observation of an image with $\sigma = 1$ pix, isophotes L=50 are indicated. Right: same observation at a distance twice as far away. The isophotes L=50 have now changed.

Actually, any observation changes the isophote landscape, so there is no way out then to include the notion of observation in the derivation of the multi-scale optic flow constraint equation. The classic optic flow constraint equation only holds for the mathematical ($\sigma = 0$) case.

Another 'problem' is the fact that a change of an isophote can only be detected in the direction *normal* to the isophote. It is impossible to detect any change in the direction tangential to the isophote. Recall the gauge coordinates of chapter 6, and it becomes clear that this phenomenon is called the gauge-condition. The aperture problem (finding a solution for the two unknowns v^x and v^y from only one equation) is actual an aperture *property*

[Florack1998a]. It limits the outcome of any isophote-related optic flow study to the normal component of the flow. In section 17.6 we derive the optic flow with the *normal constraint*.

We assume that there is conservation of topological detail, i.e. optic flow field is *differentiable*. This means that there are no discontinuities in the optic flowfield, which occur e.g. at occlusion boundaries where structure is emerging or disappearing. This constraint is expressed in the fixed temporal derivative of the flowfield: $\vec{v} = \begin{pmatrix} v^t \\ v^x \\ v^y \end{pmatrix} = \begin{pmatrix} 1 \\ v^x \\ v^y \end{pmatrix}$.

17.5 Scalar and density images

Two physically different types of images should be considered when studying optic flow. When a pixelset is deformed by a motion field, the new pixel can either keep its old value (scalar flow), or its value can be corrected for the area (volume) change of the pixel undergoing the deformation (density flow). The next illustration shows the difference:

```
px1 = {{2, 2}, {2, 4}, {4, 4}, {4, 2}}; px3 = # + {10, 0} & /@ px1;
px2 = {{5, .5}, {5, 4.5}, {9, 5}, {9, 1}}; px4 = # + {10, 0} & /@ px2;
Show[Graphics[{GrayLevel[.8],
    Polygon /@ {px1, px2, px3}, Blue, MapThread[Arrow, {px1, px2}],
    Text["scalar flow", {4, 3}], GrayLevel[.4], Polygon[px4], Red,
    MapThread[Arrow, {px3, px4}], Text["density flow", {14, 3}]}],
  AspectRatio -> Automatic, ImageSize -> 400];
```

Figure 17.6 Left: when a pixel or voxel deforms to a new size due to some flow, the pixel/voxel intensity remains the same with scalar flow. Right: with density flow the intensity changes with the inverse of the area (volume) change of the pixel (voxel).

Examples of scalar images are: range (depth) images, CT images, T1 and T2 MRI images; examples of density images are: proton density MRI images, CCD camera images, light microscope images etc.

17.6 Derivation of the multi-scale optic flow constraint equation

The following derivation is due to Florack and Nielsen [Florack1994d, Florack1998a, Nielsen1998a, Florack2000c] and further implemented and refined by Niessen et al. [Niessen1995a, Niessen1996c, Niessen1996d, Niessen1996e] for spatio-temporal optic flow, and Maas et al. [Maas1995a, Maas1996a] for stereo disparity.

We derive the equation for 2D. We *observe* the luminance distribution with the spatio-temporal Gaussian kernel $g(x, y, t; \sigma, \tau)$. The spatial scales are σ_x and σ_y, the temporal scale is τ.

```
Clear[σx, σy, τ];
g[x_, y_, t_] := 1/Sqrt[2 π σx^2] 1/Sqrt[2 π σy^2] 1/Sqrt[2 π τ^2] E^(-x^2/(2 σx^2) - y^2/(2 σy^2) - t^2/(2 τ^2));
```

The observation F is the convolution of the dynamic image sequence $L(x, y, t)$ with the Gaussian kernel in the spatial and temporal domain: $F(x, y, t; \sigma, \tau) = L \otimes g = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L(x, y, t)\, g(x, y, t, \sigma, \tau)\, dx\, dy\, dt$.

In order to follow constant intensities over time, we need the *Lie-derivative* of the observation with respect to the vectorfield of the motion to be zero. Lie derivatives capture variations of space-time quantities along the integral flow of some vectorfield. In every point the direction of the differential operator is specified by the direction of the local vector of the vectorfield. To take a Lie derivative one therefore has to know this vectorfield. In the following we only consider the first order Lie derivative of an image, which will give us a *linear* model of optic flow. This however is no restriction, and should not be confused with the spatiotemporal differential order we are interested in.

The Lie derivative (denoted with the symbol $\mathcal{L}_{\vec{v}}$) of a function $F(g)$ with respect to a vectorfield $\vec{v}$ is defined as $\mathcal{L}_{\vec{v}} F(g)$. The optic flow constraint equation (OFCE) states that the luminance does not change when we take the derivative along the vectorfield of the motion:

$$\mathcal{L}_{\vec{v}} F(g) \equiv 0$$

The Lie derivative of F with respect to the vectorfield $\vec{v}$ for scalar images is equivalent to the directional derivative of F in the direction of $\vec{v}$: $\mathcal{L}_{\vec{v}} F(g) = \partial_{\mu} F(g).\vec{v} = \vec{\nabla} F.\vec{v}$ where $\vec{\nabla}$ is the nabla operator $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$ in 2D, and $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial x}\right)$ in 3D.

We can derive this as follows. In the scalar intensity function $F(\vec{y})$ we study a small excursion in the direction of the velocity $\vec{v}$ by substituting $\vec{y} \mapsto \vec{x} + \vec{v}\, t$. We get $F(\vec{x} + \vec{v}\, t) = F(\vec{x}) + \vec{\nabla} F.\vec{v}\, t + O(t^2)$. The Lie derivative is by definition: $\mathcal{L}_{\vec{v}} F(\vec{x}) = \lim_{t \downarrow 0} \frac{F(\vec{x}+\vec{v}\, t) - F(\vec{x})}{t} = \vec{\nabla} F.\vec{v} = \sum_{i=1}^{n} \frac{\partial F}{\partial x^i} v^i$. This is the rate of change of the scalar function F when moving in the direction $\vec{v}$.

However, for density images ρ the Lie derivative with respect to a vector field is equivalent to the derivative of the density function together with the vectorfield: $\mathcal{L}_{\vec{v}} \rho = \partial_{\mu} (\rho\, \vec{v}) = \vec{\nabla}.(\rho\, \vec{v})$. The density function ρ is real valued and non-zero, so we may write: $\mathcal{L}_{\vec{v}} \rho = \rho\, \mathrm{Div}\, \vec{v} + \vec{v}.\vec{\nabla}\rho = 0$.

The derivation of the expression for Lie derivative for density image flow is as follows: In the density function $\rho(\vec{y})$ we study a small excursion in the direction of the velocity $\vec{v}$ by substituting $\vec{y} \mapsto \vec{x} + \vec{v}\, t$. We get $\rho(\vec{x} + \vec{v}\, t)$. Because we have a density flow, we need to consider the 'dilution' of the intensity when a small volume element $\rho(\vec{y})\, \overrightarrow{dy}$ changes in volume during the motion. The notation $\overrightarrow{dy}$ denotes the infinitesimal n-dimensional volume

element $d\,y^1\,d\,y^2\,...\mathrm{d}\,y^n$. This 'dilution' is taken into account by the determinant J of the Jacobian matrix (the determinant is called the *Jacobian*) $J = \det\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)$ of the transformation: $\rho(\vec{y})\,\overrightarrow{d\,y} \mapsto \rho(\vec{x} + \vec{v}\,t)\det\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)\overrightarrow{d\,y}$. In *Mathematica* the Jacobian matrix is conveniently calculated with **Outer**:

```
Outer[D, {a[x, y, z], b[x, y, z], c[x, y, z]}, {x, y, z}] // MatrixForm //
 shortnotation
```

$$\begin{pmatrix} a_x[x,y,z] & a_y[x,y,z] & a_z[x,y,z] \\ b_x[x,y,z] & b_y[x,y,z] & b_z[x,y,z] \\ c_x[x,y,z] & c_y[x,y,z] & c_z[x,y,z] \end{pmatrix}$$

We expand the expression: $\rho(\vec{x} + \vec{v}\,t)\det\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)\overrightarrow{d\,y}$. We first study the behaviour of the Jacobian matrix $\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)$ for small transformations, i.e. small values of t. The diagonal terms behave as $J_{ii} = 1 + \frac{\partial v^i}{\partial x^i}\,t + O(t^2)$ and the off-diagonal terms behave as $J_{ij} \overset{i \neq j}{=} \frac{\partial v^i}{\partial x^j}\,t + O(t^2)$. Combined: $\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right) = I + t\,B + O(t^2) \simeq I + t\,B$ where I is the identity matrix. For small t the Jacobian matrix is thus polynomial, and we may write for the determinant: $J = \det(I + t\,B) \simeq \det I + t\,\mathrm{trace}\,B + O(t^2)$.

Combining the expansions for $\rho(\vec{x} + \vec{v}\,t)$ and $\det\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)\overrightarrow{d\,y}$ we get:

$\rho(\vec{x} + \vec{v}\,t)\det\left(\frac{\partial \vec{y}}{\partial \vec{x}}\right)\overrightarrow{d\,y} = \left\{\rho(\vec{x}) + \vec{\nabla}\rho.\vec{v}\,t + O(t^2)\right\}\left\{1 + t\,\vec{\nabla}.\vec{v} + O(t^2)\right\}\overrightarrow{d\,y} = \left\{\rho + \rho\,t\,\vec{\nabla}.\vec{v} + \vec{\nabla}\rho.\vec{v}\,t + O(t^2)\right\}\overrightarrow{d\,y} \overset{\text{def}}{\equiv} \{\rho + \mathcal{L}_{\vec{v}}\,\rho(x).t + O(t^2)\}\,\overrightarrow{d\,y}$, from which we derive: $\mathcal{L}_{\vec{v}}\,\rho(x) = \rho\,\vec{\nabla}.\vec{v} + \vec{\nabla}\rho.\vec{v}$.

Just as we have seen in the first chapters, the *observation* of F is the convolution $F \otimes g$ where g is the Gaussian aperture and F is the spatiotemporal image distribution. Again, the differential (Lie) operator may be moved to the Gaussian kernel: $\mathcal{L}_{\vec{v}}\,F \otimes g = F \otimes (\mathcal{L}_{\vec{v}}\,g)$.

▲ Task 17.2 Explain what differential operators emerge when the vectorfield consists of unity vectors {0,1} at every point, resp. {1,0}.

For scalar images the optic flow constraint equation under the convolution is written as:

$\int\left(\vec{\nabla}F.\vec{v}\right)g\,d\vec{x} = 0$, from which we get by partial integration: $-\int F\,\vec{\nabla}.(g\,\vec{v})\,d\vec{x} = 0$, or

$-\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L(x - x', y - y', t - t')\,\mathrm{Grad}\,g(x, y, t).\,\vec{v}(x, y, t)\,dx'\,dy'\,dt' = 0$ where Grad is the *gradient operator* ($\vec{\nabla}$) and g the Gaussian kernel.

For density images the optic flow constraint equation under the convolution is written as:

$\int\vec{\nabla}.(\rho\,\vec{v})\,g\,d\vec{x} = 0$, from which we get by partial integration: $-\int\rho\left(\vec{\nabla}g.\,\vec{v}\right)d\vec{x} = 0$, or

$-\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\rho(x - x', y - y', t - t')\,\mathrm{Div}(g(x, y, t)\,\vec{v}(x, y, t))\,dx'\,dy'\,dt' = 0$ where Div is the *divergence operator* ($\vec{\nabla}.$).

The motion vectorfield $\vec{v}$ is the unknown in this equation that we like to establish from a series of observations of the image. In the following we will derive a set of equations, in which we assume some *approximated* vectorfield for the unknown flow, and find just as many equations

as we have unknowns in our approximation of the flowfield. The solution of this set of equations in each pixel gives us then the approximated flowfield.

We approximate the unknown vector field to some order m. We define the optic flow vectorfield with 3 components `{u[x,y,t],v[x,y,t],w[x,y,t]}`, where `u` is the x-component, `v` the y-component and `w` the t-component of the vectorfield. Here is the approximation to order `m = 1`:

```
Clear[u, v, w];
vectorfield[x_, y_, t_, m_] :=
 {Normal[Series[u[x, y, t], {x, 0, m}, {y, 0, m}, {t, 0, m}]],
  Normal[Series[v[x, y, t], {x, 0, m}, {y, 0, m}, {t, 0, m}]],
  Normal[Series[w[x, y, t], {x, 0, m}, {y, 0, m}, {t, 0, m}]]} /.
 {Derivative[a_, b_, c_][u_][0, 0, 0] /; (a + b + c > m) → 0};
vectorfield[x, y, t, 1]
```

$$\{u[0,0,0] + t\,u^{(0,0,1)}[0,0,0] + y\,u^{(0,1,0)}[0,0,0] + x\,u^{(1,0,0)}[0,0,0],$$
$$v[0,0,0] + t\,v^{(0,0,1)}[0,0,0] + y\,v^{(0,1,0)}[0,0,0] + x\,v^{(1,0,0)}[0,0,0],$$
$$w[0,0,0] + t\,w^{(0,0,1)}[0,0,0] + y\,w^{(0,1,0)}[0,0,0] + x\,w^{(1,0,0)}[0,0,0]\}$$

Note that the `Series` command gives us more terms because it nests the expansion. With a conditional (`/;` is a shortwrite for `Condition`) `Replace` statement (`/.`) we set all terms with order higher than `m = 1` to zero. So we keep only terms in which the total order is one. The expression becomes more readable when we write the derivatives as subscripted variables with the function `short`:

```
short[expr_] := Module[{nx, ny, nt, u}, DisplayForm[expr /.
      Derivative[nx_, ny_, nt_][L_][x_, y_, t_] -> Subscript[L,
 StringJoin[Table["x", {nx}], Table["y", {ny}], Table["t", {nt}]]] /.
     u_[0, 0, 0] -> u]] /.
  Hold[gDn[im, {nt_, ny_, nx_}, {τ, σy, σx}]] -> Subscript["L",
    StringJoin[Table["x", {nx}], Table["y", {ny}], Table["t", {nt}]]]

vectorfield[x, y, t, 1] // short
```

$$\{u + t\,u_t + x\,u_x + y\,u_y,\ v + t\,v_t + x\,v_x + y\,v_y,\ w + t\,w_t + x\,w_x + y\,w_y\}$$

In this example of a first order vectorfield we encounter an equation where 8 unknown components of the vectorfield should be solved: u, u_x, u_y, u_t, v, v_x, v_y and v_t. With only one equation this is of course impossible. However, because the Lie derivative of the image vanishes identically, so do all the partial derivatives of this. It can be shown that it is allowed to take up to M-th order derivatives provided the flow vector is approximated to M-th polynomial order as well. So we may add the equations for the vanishing Lie derivatives with respect to x, y and t, because we are studying a first order vectorfield. This gives us three extra equations. The remaining 4 have to come from *external* information. Important external information can for example be found when constraints on the flow are known, such as the constraint that only normal flow can be extracted.

The *normal constraint* in 2D is expressed as $\vec{n}.\vec{v} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \vec{\nabla} L.\vec{v} = 0$ or $-v\,L_x + u\,L_y = 0$ where L_x and L_y are constant. This is our fifth equation in the set of optic flow constraint equations to be solved for the 8 unknowns. We add the derivatives of the normal constraint

equation with respect to x, y and t, and we have the set of 8 equations complete, which can then be solved for each pixel in the temporal sequence to give the normal flow components and their first order derivatives.

Other external conditions that give additional constraint equations are for example:
- when we know the flow is just a translation in the x-direction: $v = \text{const}$, $u = 0$; $v_x = v_y = v_t = 0$;
- when we know the flow is radial, we zoom in onto a scene or fly to a vanishing point: $-v + u = 0$;
- when we know the camera rotates around the optical axis: $v L_x + u L_y = 0$;
- the smoothness constraint (e.g. Lucas and Kanade [Lucas1981]) which use a weighted least-squares solution to the optic flow problem by minimizing $\int_{\mathbb{R}^2} W(x)^2 (u L_x + v L_y + L_t)^2 \, dx \, dy$.

We end the chapter with a section on how an appropriate scale can be selected, and numerical examples showing the effectiveness of the method. We start with the derivation and implementation of the optic flow constraint equation for scalar images.

17.6.1 Scalar images, normal flow.

Mathematica has all the machinery on board to analytically calculate the Lie derivatives, and subsequently replace the spatio-temporal image derivatives by discrete convolution with the appropriate Gaussian derivatives. We load the package **`Calculus`VectorAnalysis`** with the definitions of the nabla operator and its actions on scalar- and vectorfields (divergence and gradient operators respectively). We set the coordinates to the spatiotemporal Euclidean space (x, y, t):

```
<< Calculus`VectorAnalysis`;
SetCoordinates[Cartesian[x, y, t]];
```

For scalar images the Lie derivative of the observed spatiotemporal image L is defined as $\mathcal{L}_{\vec{v}} L(g) \equiv L(\mathcal{L}_{\vec{v}}{}^T g)$ where $\mathcal{L}_{\vec{v}}{}^T g(x) = -\vec{\nabla}.(g\,\vec{v})$, so we get

$$\mathcal{L}_{\vec{v}} L(x, y, t; \sigma, \tau) =$$
$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L(x', y', t')\, \mathcal{L}_{\vec{v}}{}^T (g(x-x', y-y', t-t', \sigma, \tau)\, \vec{v}(x-x', y-y', t-t'))\, dx'\, dy'\, dt' =$$
$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L(x', y', t')\, (-\text{div}(g(x-x', y-y', t-t', \sigma, \tau)\, \vec{v}(x-x', y-y', t-t')))\, dx'\, dy'\, dt' = .$$
$$0$$

The triple integral represents the process of 3D (2D-time) convolution. We firstly calculate the (polynomial) expression for the Lie differential operator **`-Div[g[x,y,t] v[x,y,t]]`** expressed in Gaussian derivatives. By replacing the Gaussian derivatives with our familiar convolution operator **`gD`** we implement the convolution on the input image **`L[x,y,t]`**.

```
m = 1;
scalarflow = Expand[Simplify[
    {Div[g[x, y, t] vectorfield[x, y, t, m]],
      Div[∂x g[x, y, t] vectorfield[x, y, t, m]],
      Div[∂y g[x, y, t] vectorfield[x, y, t, m]],
      Div[∂t g[x, y, t] vectorfield[x, y, t, m]]} / g[x, y, t]]];
short[scalarflow]
```

$$\left\{-\frac{u\,x}{\sigma x^2}-\frac{v\,y}{\sigma y^2}-\frac{t\,w}{\tau^2}-\frac{t\,x\,u_t}{\sigma x^2}+u_x-\frac{x^2\,u_x}{\sigma x^2}-\frac{x\,y\,u_y}{\sigma x^2}-\right.$$
$$\frac{t\,y\,v_t}{\sigma y^2}-\frac{x\,y\,v_x}{\sigma y^2}+v_y-\frac{y^2\,v_y}{\sigma y^2}+w_t-\frac{t^2\,w_t}{\tau^2}-\frac{t\,x\,w_x}{\tau^2}-\frac{t\,y\,w_y}{\tau^2},$$
$$\frac{u\,x^2}{\sigma x^4}-\frac{u}{\sigma x^2}+\frac{v\,x\,y}{\sigma x^2\,\sigma y^2}+\frac{t\,w\,x}{\sigma x^2\,\tau^2}+\frac{t\,x^2\,u_t}{\sigma x^4}-\frac{t\,u_t}{\sigma x^2}+\frac{x^3\,u_x}{\sigma x^4}-$$
$$\frac{2\,x\,u_x}{\sigma x^2}+\frac{x^2\,y\,u_y}{\sigma x^4}-\frac{y\,u_y}{\sigma x^2}+\frac{t\,x\,y\,v_t}{\sigma x^2\,\sigma y^2}+\frac{x^2\,y\,v_x}{\sigma x^2\,\sigma y^2}-\frac{x\,v_y}{\sigma x^2}+$$
$$\frac{x\,y^2\,v_y}{\sigma x^2\,\sigma y^2}-\frac{x\,w_t}{\sigma x^2}+\frac{t^2\,x\,w_t}{\sigma x^2\,\tau^2}+\frac{t\,x^2\,w_x}{\sigma x^2\,\tau^2}+\frac{t\,x\,y\,w_y}{\sigma x^2\,\tau^2},$$
$$\frac{v\,y^2}{\sigma y^4}-\frac{v}{\sigma y^2}+\frac{u\,x\,y}{\sigma x^2\,\sigma y^2}+\frac{t\,w\,y}{\sigma y^2\,\tau^2}+\frac{t\,x\,y\,u_t}{\sigma x^2\,\sigma y^2}-\frac{y\,u_x}{\sigma y^2}+\frac{x^2\,y\,u_x}{\sigma x^2\,\sigma y^2}+\frac{x\,y^2\,u_y}{\sigma x^2\,\sigma y^2}+\frac{t\,y^2\,v_t}{\sigma y^4}-$$
$$\frac{t\,v_t}{\sigma y^2}+\frac{x\,y^2\,v_x}{\sigma y^4}-\frac{x\,v_x}{\sigma y^2}+\frac{y^3\,v_y}{\sigma y^4}-\frac{2\,y\,v_y}{\sigma y^2}-\frac{y\,w_t}{\sigma y^2}+\frac{t^2\,y\,w_t}{\sigma y^2\,\tau^2}+\frac{t\,x\,y\,w_x}{\sigma y^2\,\tau^2}+\frac{t\,y^2\,w_y}{\sigma y^2\,\tau^2},$$
$$\frac{t^2\,w}{\tau^4}-\frac{w}{\tau^2}+\frac{t\,u\,x}{\sigma x^2\,\tau^2}+\frac{t\,v\,y}{\sigma y^2\,\tau^2}+\frac{t^2\,x\,u_t}{\sigma x^2\,\tau^2}-\frac{t\,u_x}{\tau^2}+\frac{t\,x^2\,u_x}{\sigma x^2\,\tau^2}+\frac{t\,x\,y\,u_y}{\sigma x^2\,\tau^2}+\frac{t^2\,y\,v_t}{\sigma y^2\,\tau^2}+$$
$$\left.\frac{t\,x\,y\,v_x}{\sigma y^2\,\tau^2}-\frac{t\,v_y}{\tau^2}+\frac{t\,y^2\,v_y}{\sigma y^2\,\tau^2}+\frac{t^3\,w_t}{\tau^4}-\frac{2\,t\,w_t}{\tau^2}+\frac{t^2\,x\,w_x}{\tau^4}-\frac{x\,w_x}{\tau^2}+\frac{t^2\,y\,w_y}{\tau^4}-\frac{y\,w_y}{\tau^2}\right\}$$

In the statements above we derive the expression for the scalarflow $-\mathrm{div}\,(g(x, y, t)\,v(x, y, t))$ and for the first order derivatives of the flow $(-\mathrm{div}\,(g_x(x, y, t)\,v(x, y, t))$, $-\mathrm{div}\,(g_y(x, y, t)\,v(x, y, t))$ and $-\mathrm{div}\,(g_t(x, y, t)\,v(x, y, t)))$. We divide by the always positive function $g(x, y, t)$ in order to get the coefficients that occur in front of $g(x, y, t)$ and expand the expressions to get all polynomial factors as separate terms. We study the 4 equations of the result in short notation:

We see that the 8 components of the unknown vectorfield (u, u_x, u_y, u_t, v, v_x, v_y and v_t) show up, but with difficult to handle terms involving x, x^2, y, y^2, t, $t\,u\,x$, $t\,v\,y$, etc. The way to cope with these terms was suggested by Florack and Nielsen: Gaussian derivatives are equivalent to the Gaussian kernel multiplied with a Hermite polynomial of that order (see chapter 4 on Gaussian derivatives). So we are able to convert the set of terms above into proper Gaussian derivatives. These Gaussian derivatives convolve with the input image to get scaled derivatives of the image, which can all be *measured*. We define the Hermite function

```
hermite[x_, n_, σ_] := (-1/(σ √2))^n HermiteH[n, x/(σ √2)];
```

and construct the spatiotemporal (x, y, t) Hermite polynomial of order **n** in x, **m** in y and **k** in t (due to separability this reduces to a product):

```
polynomial[n_, m_, k_] :=
 Simplify[hermite[x, n, σx] hermite[y, m, σy] hermite[t, k, τ],
  {σx > 0, σy > 0, τ > 0}]
```

An example:

```
polynomial[1, 1, 1]
```

$$-\frac{t\,x\,y}{\sigma x^2\,\sigma y^2\,\tau^2}$$

We create a list of all Hermite polynomials having at least a first order expansion in x, y or t. We start with an empty list and append only a term when the sum of the orders is nonzero and less than or equal to 3. We get 19 combinations:

```
terms = {};
Do[If[0 < n + m + k ≤ 3, terms = Expand[Append[terms, polynomial[n, m, k]]]],
 {m, 0, 3}, {n, 0, 3}, {k, 0, 3}];
Length[terms]
terms
```

```
19
```

$$\left\{-\frac{t}{\tau^2}, \frac{t^2}{\tau^4}-\frac{1}{\tau^2}, -\frac{t^3}{\tau^6}+\frac{3\,t}{\tau^4}, -\frac{x}{\sigma x^2}, \frac{t\,x}{\sigma x^2\,\tau^2}, -\frac{t^2\,x}{\sigma x^2\,\tau^4}+\frac{x}{\sigma x^2\,\tau^2},\right.$$
$$\frac{x^2}{\sigma x^4}-\frac{1}{\sigma x^2}, -\frac{t\,x^2}{\sigma x^4\,\tau^2}+\frac{t}{\sigma x^2\,\tau^2}, -\frac{x^3}{\sigma x^6}+\frac{3\,x}{\sigma x^4}, -\frac{y}{\sigma y^2}, \frac{t\,y}{\sigma y^2\,\tau^2},$$
$$-\frac{t^2\,y}{\sigma y^2\,\tau^4}+\frac{y}{\sigma y^2\,\tau^2}, \frac{x\,y}{\sigma x^2\,\sigma y^2}, -\frac{t\,x\,y}{\sigma x^2\,\sigma y^2\,\tau^2}, -\frac{x^2\,y}{\sigma x^4\,\sigma y^2}+\frac{y}{\sigma x^2\,\sigma y^2},$$
$$\left.\frac{y^2}{\sigma y^4}-\frac{1}{\sigma y^2}, -\frac{t\,y^2}{\sigma y^4\,\tau^2}+\frac{t}{\sigma y^2\,\tau^2}, -\frac{x\,y^2}{\sigma x^2\,\sigma y^4}+\frac{x}{\sigma x^2\,\sigma y^2}, -\frac{y^3}{\sigma y^6}+\frac{3\,y}{\sigma y^4}\right\}$$

The equivalence relations between these 19 coefficients (as prefactors for the Gaussian kernel) and the corresponding Gaussian derivative functions can be found by solving 19 simultaneous equations. In the following we explain the steps to build these equations using the pattern matching capability of *Mathematica* step by step. The same machinery can then be easily applied to optic flow equations of other orders or higher approximation orders of the velocity flowfield.

We define a set of temporary variables **order[a,b,c]** capturing the order of the exponents of x, y and t as follows:

```
Clear[a, b, c]; exponents = Transpose[Exponent[terms, #] & /@ {x, y, t}] /.
  {a_, b_, c_} -> order[a, b, c]
```

```
{order[0, 0, 1], order[0, 0, 2], order[0, 0, 3],
 order[1, 0, 0], order[1, 0, 1], order[1, 0, 2], order[2, 0, 0],
 order[2, 0, 1], order[3, 0, 0], order[0, 1, 0], order[0, 1, 1],
 order[0, 1, 2], order[1, 1, 0], order[1, 1, 1], order[2, 1, 0],
 order[0, 2, 0], order[0, 2, 1], order[1, 2, 0], order[0, 3, 0]}
```

which belong to the 'pure' polynomial terms

```
vars = Exponent[terms, #] & /@ {x, y, t} /. {a_, b_, c_} -> x^a y^b t^c
```

```
{t, t^2, t^3, x, t x, t^2 x, x^2, t x^2, x^3,
 y, t y, t^2 y, x y, t x y, x^2 y, y^2, t y^2, x y^2, y^3}
```

We assign these pure polynomial terms to a set of 19 new variables **k[i]** using **MapThread**. The order of the set of replacement rules must be reversed, in order to replace the higher order terms first in the step to follow. For example in this way $x\,y^2$ is replaced before y^2. Otherwise, x would be replaced individually which would lead to wrong results. The result is a set of assignment rules:

```
rules = MapThread[Rule, {vars, Table[k[i], {i, Length[vars]}]}] // Reverse
```

```
{y^3 → k[19], x y^2 → k[18], t y^2 → k[17], y^2 → k[16], x^2 y → k[15], t x y → k[14],
 x y → k[13], t^2 y → k[12], t y → k[11], y → k[10], x^3 → k[9], t x^2 → k[8],
 x^2 → k[7], t^2 x → k[6], t x → k[5], x → k[4], t^3 → k[3], t^2 → k[2], t → k[1]}
```

The set of rules is applied to our initial set of terms:

```
kterms = terms /. rules
```

$$\left\{-\frac{k[1]}{\tau^2}, -\frac{1}{\tau^2}+\frac{k[2]}{\tau^4}, \frac{3\,k[1]}{\tau^4}-\frac{k[3]}{\tau^6}, -\frac{k[4]}{\sigma x^2}, \frac{k[5]}{\sigma x^2\,\tau^2}, \frac{k[4]}{\sigma x^2\,\tau^2}-\frac{k[6]}{\sigma x^2\,\tau^4},\right.$$
$$-\frac{1}{\sigma x^2}+\frac{k[7]}{\sigma x^4}, \frac{k[1]}{\sigma x^2\,\tau^2}-\frac{k[8]}{\sigma x^4\,\tau^2}, \frac{3\,k[4]}{\sigma x^4}-\frac{k[9]}{\sigma x^6}, -\frac{k[10]}{\sigma y^2}, \frac{k[11]}{\sigma y^2\,\tau^2},$$
$$\frac{k[10]}{\sigma y^2\,\tau^2}-\frac{k[12]}{\sigma y^2\,\tau^4}, \frac{k[13]}{\sigma x^2\,\sigma y^2}, -\frac{k[14]}{\sigma x^2\,\sigma y^2\,\tau^2}, \frac{k[10]}{\sigma x^2\,\sigma y^2}-\frac{k[15]}{\sigma x^4\,\sigma y^2},$$
$$\left.-\frac{1}{\sigma y^2}+\frac{k[16]}{\sigma y^4}, \frac{k[1]}{\sigma y^2\,\tau^2}-\frac{k[17]}{\sigma y^4\,\tau^2}, \frac{k[4]}{\sigma x^2\,\sigma y^2}-\frac{k[18]}{\sigma x^2\,\sigma y^4}, \frac{3\,k[10]}{\sigma y^4}-\frac{k[19]}{\sigma y^6}\right\}$$

and converted into a set of 19 equations (we show for brevity only the first and last equations, with **Short**):

```
set = MapThread[Equal, {kterms, exponents}]; Short[set, 6]
```

$$\left\{-\frac{k[1]}{\tau^2} == \text{order}[0,0,1], -\frac{1}{\tau^2}+\frac{k[2]}{\tau^4} == \text{order}[0,0,2], \text{<<15>>},\right.$$
$$\left.\frac{k[4]}{\sigma x^2\,\sigma y^2}-\frac{k[18]}{\sigma x^2\,\sigma y^4} == \text{order}[1,2,0], \frac{3\,k[10]}{\sigma y^4}-\frac{k[19]}{\sigma y^6} == \text{order}[0,3,0]\right\}$$

Mathematica must solve this recursively, injecting at each equation one more rule at the time, giving a set of rules for **k[i]**:

```
rk = {}; Do[rk = Flatten[Append[rk, Solve[Take[set, i], k[i]] /. rk]],
 {i, 1, Length[terms]}];
rk
```

$$\{k[1] \to -\tau^2\,\text{order}[0,0,1],\ k[2] \to \tau^2\,(1+\tau^2\,\text{order}[0,0,2]),$$
$$k[3] \to -\tau^2\,(3\,\tau^2\,\text{order}[0,0,1]+\tau^4\,\text{order}[0,0,3]),\ k[4] \to -\sigma x^2\,\text{order}[1,0,0],$$
$$k[5] \to \sigma x^2\,\tau^2\,\text{order}[1,0,1],\ k[6] \to -\tau^2\,(\sigma x^2\,\text{order}[1,0,0]+\sigma x^2\,\tau^2\,\text{order}[1,0,2]),$$
$$k[7] \to \sigma x^2\,(1+\sigma x^2\,\text{order}[2,0,0]),$$
$$k[8] \to -\sigma x^2\,(\tau^2\,\text{order}[0,0,1]+\sigma x^2\,\tau^2\,\text{order}[2,0,1]),$$
$$k[9] \to -\sigma x^2\,(3\,\sigma x^2\,\text{order}[1,0,0]+\sigma x^4\,\text{order}[3,0,0]),\ k[10] \to -\sigma y^2\,\text{order}[0,1,0],$$
$$k[11] \to \sigma y^2\,\tau^2\,\text{order}[0,1,1],\ k[12] \to -\tau^2\,(\sigma y^2\,\text{order}[0,1,0]+\sigma y^2\,\tau^2\,\text{order}[0,1,2]),$$
$$k[13] \to \sigma x^2\,\sigma y^2\,\text{order}[1,1,0],\ k[14] \to -\sigma x^2\,\sigma y^2\,\tau^2\,\text{order}[1,1,1],$$
$$k[15] \to -\sigma x^2\,(\sigma y^2\,\text{order}[0,1,0]+\sigma x^2\,\sigma y^2\,\text{order}[2,1,0]),$$
$$k[16] \to \sigma y^2\,(1+\sigma y^2\,\text{order}[0,2,0]),$$
$$k[17] \to -\sigma y^2\,(\tau^2\,\text{order}[0,0,1]+\sigma y^2\,\tau^2\,\text{order}[0,2,1]),$$
$$k[18] \to -\sigma y^2\,(\sigma x^2\,\text{order}[1,0,0]+\sigma x^2\,\sigma y^2\,\text{order}[1,2,0]),$$
$$k[19] \to -\sigma y^2\,(3\,\sigma y^2\,\text{order}[0,1,0]+\sigma y^4\,\text{order}[0,3,0])\}$$

Now we can inject the solutions for **k[i]** into the rules that convert the pure exponential terms:

```
rulef = rules /. rk; Short[rulef, 6]
```

```
{y^3 → -σy^2 (3 σy^2 order[0, 1, 0] + σy^4 order[0, 3, 0]),
 x y^2 → -σy^2 (σx^2 order[1, 0, 0] + σx^2 σy^2 order[1, 2, 0]),
  <<15>>, t^2 → τ^2 (1 + τ^2 order[0, 0, 2]), t → -τ^2 order[0, 0, 1]}
```

Partial integration gives a minus sign with odd total numbers of differentiation. We can now plug in the Gaussian derivative operators to actually calculate the derivatives from the spatiotemporal sequence. We replace the terms **order** by **Hold[gDn[im,{nx,ny,nt},{σx,σy,τ}]]**. The **Hold** function prevents immediate calculation. We need **ReplaceRepeated (//.)** because sometimes more replacements have to be done in more passes.

```
scalartmpflow = (Expand[scalarflow //. rulef]) /.
  order[a_, b_, c_] -> (-1)^(a+b+c) Hold[gDn[im, {c, b, a}, {τ, σy, σx}]];
short[scalartmpflow]
```

```
{-w L_t - u L_x - v L_y - τ^2 L_xt u_t - σx^2 L_xx u_x - σy^2 L_xy u_y -
   τ^2 L_yt v_t - σx^2 L_xy v_x - σy^2 L_yy v_y - τ^2 L_tt w_t - σx^2 L_xt w_x - σy^2 L_yt w_y,
 w L_xt + u L_xx + v L_xy + τ^2 L_xxt u_t + L_x u_x + σx^2 L_xxx u_x + σy^2 L_xxy u_y + τ^2 L_xyt v_t +
   σx^2 L_xxy v_x + L_y v_x + σy^2 L_xyy v_y + τ^2 L_xtt w_t + L_t w_x + σx^2 L_xxt w_x + σy^2 L_xyt w_y,
 u L_xy + w L_yt + v L_yy + τ^2 L_xyt u_t + σx^2 L_xxy u_x + L_x u_y + σy^2 L_xyy u_y + τ^2 L_yyt v_t +
   σx^2 L_xyy v_x + L_y v_y + σy^2 L_yyy v_y + τ^2 L_ytt w_t + σx^2 L_xyt w_x + L_t w_y + σy^2 L_yyt w_y,
 w L_tt + u L_xt + v L_yt + L_x u_t + τ^2 L_xtt u_t + σx^2 L_xxt u_x + σy^2 L_xyt u_y + L_y v_t +
   τ^2 L_ytt v_t + σx^2 L_xyt v_x + σy^2 L_yyt v_y + L_t w_t + τ^2 L_ttt w_t + σx^2 L_xtt w_x + σy^2 L_ytt w_y}
```

Note that in the approximated flowfield we introduced derivatives of u, v, and w with respect to x, y and t. Here w is the component of the velocity field in the temporal direction. The velocity component in this direction emerges when structure disappears or emerges, such as at occlusion boundaries of objects in the image. However, at this time we will require that there is no such disappearance or emergence of structure. This constraint translates to limiting the temporal component w of the flowfield to the zeroth order, which we put to unity, and all derivatives of w vanish:

```
scalardataflow = Expand[scalartmpflow /. {w[0, 0, 0] -> 1,
    Derivative[a_, b_, c_][w][0, 0, 0] -> 0}]; short[scalardataflow]
```

```
{-L_t - u L_x - v L_y - τ^2 L_xt u_t - σx^2 L_xx u_x - σy^2 L_xy u_y - τ^2 L_yt v_t -
   σx^2 L_xy v_x - σy^2 L_yy v_y, L_xt + u L_xx + v L_xy + τ^2 L_xxt u_t + L_x u_x +
   σx^2 L_xxx u_x + σy^2 L_xxy u_y + τ^2 L_xyt v_t + σx^2 L_xxy v_x + L_y v_x + σy^2 L_xyy v_y,
 u L_xy + L_yt + v L_yy + τ^2 L_xyt u_t + σx^2 L_xxy u_x + L_x u_y + σy^2 L_xyy u_y + τ^2 L_yyt v_t +
   σx^2 L_xyy v_x + L_y v_y + σy^2 L_yyy v_y, L_tt + u L_xt + v L_yt + L_x u_t + τ^2 L_xtt u_t +
   σx^2 L_xxt u_x + σy^2 L_xyt u_y + L_y v_t + τ^2 L_ytt v_t + σx^2 L_xyt v_x + σy^2 L_yyt v_y}
```

In matrix notation: $A\,\vec{v} = \vec{b}$, where A is given by the coefficients in the 4 x 8 matrix above, and $\vec{v}^T = \{u, v, u_t, v_t, u_x, v_x, u_y, v_y\}$ and $\vec{b} = \{-L_t, -L_{tt}, -L_{xt}, L_{yt}\}$.

Note that this set of equations has become quite complex. In order to estimate a flowfield approximated to first order we need to extract spatiotemporal derivatives to third order, and temporal derivatives to second order. This of course has implications for the requirements to the datasets from which the field is to be extracted. The more images in the temporal sequence the better. In the limiting case of just 2 images as in a stereo pair we have to approximate the first order temporal derivative by the mere difference, and put the second order temporal derivative to zero.

We acquired four equations with the eight unknowns u, u_x, u_y, u_t, v, v_x, v_y and v_t. The four additional equations required in order to be able to solve the eight unknowns in each pixel can only be formulated by incorporating *external* physical constraints to the flow. We choose in this example the constraint of *normal flow*, which leads to an extra four equations. The normal flow is easily derived from the regular flow by the substitution $\{u, v, 1\} \rightarrow \{-v, u, 0\}$. It can also be expressed as $-v L_x + u L_y = 0$ where L_x and L_y are constant. This equation expresses the fact that the tangential component of the velocity field vanishes. We first replace **u** into a temporary variable **vtmp**, then replace the derivatives of **u** into the derivatives of **vtmp**, then we replace the **v** into **u** and the derivatives of **v** into the derivatives of **u**, and finally replace **vtmp** back into **v**.

```
scalarnormalflow = scalartmpflow /.
    {u[0, 0, 0] -> -vtmp[0, 0, 0], Derivative[a_, b_, c_][u][0, 0, 0] ->
      -Derivative[a, b, c][vtmp][0, 0, 0], v[0, 0, 0] -> u[0, 0, 0],
     Derivative[a_, b_, c_][v][0, 0, 0] -> Derivative[a, b, c][u][0, 0, 0],
     w[0, 0, 0] -> 0, Derivative[a_, b_, c_][w][0, 0, 0] -> 0} /. vtmp -> v;
short[scalarnormalflow]
```

$$\{v L_x - u L_y - \tau^2 L_{yt} u_t - \sigma x^2 L_{xy} u_x - \sigma y^2 L_{yy} u_y + \tau^2 L_{xt} v_t + \sigma x^2 L_{xx} v_x + \sigma y^2 L_{xy} v_y,$$
$$-v L_{xx} + u L_{xy} + \tau^2 L_{xyt} u_t + \sigma x^2 L_{xxy} u_x + L_y u_x +$$
$$\sigma y^2 L_{xyy} u_y - \tau^2 L_{xxt} v_t - L_x v_x - \sigma x^2 L_{xxx} v_x - \sigma y^2 L_{xxy} v_y,$$
$$-v L_{xy} + u L_{yy} + \tau^2 L_{yyt} u_t + \sigma x^2 L_{xyy} u_x + L_y u_y + \sigma y^2 L_{yyy} u_y - \tau^2 L_{xyt} v_t -$$
$$\sigma x^2 L_{xxy} v_x - L_x v_y - \sigma y^2 L_{xyy} v_y, -v L_{xt} + u L_{yt} + L_y u_t + \tau^2 L_{ytt} u_t +$$
$$\sigma x^2 L_{xyt} u_x + \sigma y^2 L_{yyt} u_y - L_x v_t - \tau^2 L_{xtt} v_t - \sigma x^2 L_{xxt} v_x - \sigma y^2 L_{xyt} v_y\}$$

In matrix notation: $N \vec{v} = \vec{0}$. The total set of eight equations is the concatenation of the two sets, forming eight equations with eight unknowns:

```
scalarflowequations = Join[scalardataflow, scalarnormalflow];
```

17.6.2 Density images, normal flow.

For density images the Lie derivative is $\mathcal{L}_v{}^T g(x) = -\vec{\nabla} g . \vec{v} = 0$. We can now give the full derivation of the 8 constraint equations to be solved in each pixel as a single routine, for the same conditions as above: approximation of the vector field to first order, no creation of new structure and the normal flow constraint:

```
<< Calculus`VectorAnalysis`;
SetCoordinates[Cartesian[x, y, t]];

densityflowequations[order_] :=
Module[{g, densityflow0, densityflow, hermite,
   polynomial, terms, exponents, vars, rules, kterms, rulef, rk,
```

```
densitytmpflow, densitydataflow, densitynormalflow, vtmp}, im = .;
Clear[σx, σy, τ]; g[x_, y_, t_] := 1/√(2 π σx²) 1/√(2 π σy²) 1/√(2 π τ²) E^(-x²/(2 σx²) - y²/(2 σy²) - t²/(2 τ²));
Clear[u, v, w]; m = order;
vectorfield[x, y, t, m] =
  {Normal[Series[u[x, y, t], {x, 0, m}, {y, 0, m}, {t, 0, m}]],
   Normal[Series[v[x, y, t], {x, 0, m}, {y, 0, m}, {t, 0, m}]],
   Normal[Series[w[x, y, t], {x, 0, m}, {y, 0, m}, {t, 0, m}]]} /.
  {Derivative[a_, b_, c_][u_][0, 0, 0] /; (a + b + c > m) -> 0};
densityflow0 = Expand[Simplify[
   (-Grad[g[x, y, t]].vectorfield[x, y, t, m]) / g[x, y, t]]];
densityflow = If[m == 0, {densityflow0},
  Expand[Simplify[{-Grad[g[x, y, t]].vectorfield[x, y, t, m],
     -Grad[∂x g[x, y, t]].vectorfield[x, y, t, m],
     -Grad[∂y g[x, y, t]].vectorfield[x, y, t, m],
     -Grad[∂t g[x, y, t]].vectorfield[x, y, t, m]} / g[x, y, t]]]];
hermite[x_, n_, σ_] := (-1/(σ √2))^n HermiteH[n, x/(σ √2)];
polynomial[n_, m_, k_] := Simplify[
  hermite[x, n, σx] hermite[y, m, σy] hermite[t, k, τ], {σx > 0, σy > 0, τ > 0}];
terms = {}; Do[If[0 < n + m + k ≤ 3, terms = Expand[Append[terms, polynomial[n, m, k]]]],
 {m, 0, 3}, {n, 0, 3}, {k, 0, 3}];
Clear[a, b, c]; exponents = Transpose[Exponent[terms, #] & /@ {x, y, t}] /.
  {a_, b_, c_} -> order[a, b, c];
vars = Exponent[terms, #] & /@ {x, y, t} /. {a_, b_, c_} -> x^a y^b t^c;
rules = MapThread[Rule, {vars, Table[k[i], {i, Length[vars]}]}] // Reverse;
kterms = terms /. rules; set = MapThread[Equal, {kterms, exponents}];
 rk = {}; Do[rk = Flatten[Append[rk, Solve[Take[set, i], k[i]] /. rk]],
  {i, 1, Length[terms]}];
rulef = rules /. rk;
densitytmpflow = (Expand[densityflow //. rulef]) /.
  order[a_, b_, c_] -> (-1)^(a+b+c) Hold[gDn[im, {c, b, a}, {τ, σy, σx}]];
densitydataflow = Expand[densitytmpflow /.
   {w[0, 0, 0] -> 1, Derivative[a_, b_, c_][w][0, 0, 0] -> 0}];
densitynormalflow = densitytmpflow /. {u[0, 0, 0] -> -vtmp[0, 0, 0],
    Derivative[a_, b_, c_][u][0, 0, 0] -> -Derivative[a, b, c][vtmp][0, 0, 0],
    v[0, 0, 0] -> u[0, 0, 0],
    Derivative[a_, b_, c_][v][0, 0, 0] -> Derivative[a, b, c][u][0, 0, 0],
    w[0, 0, 0] -> 0, Derivative[a_, b_, c_][w][0, 0, 0] -> 0} /. vtmp -> v;
Join[densitydataflow, densitynormalflow]
```

The density flow equations for zeroth order lead to the classical Horn and Schunck optic flow constraint equations [Horn1981]:

```
densityflowequations[0] // short
```

$$\{L_t + u\,L_x + v\,L_y,\ -v\,L_x + u\,L_y\}$$

For a first order approximation of the unknown flow field we get much more derivatives, up to third order, and eight equations:

```
densityflowequations[1] // short
```

$$\{L_t + u\,L_x + v\,L_y + \tau^2\,L_{xt}\,u_t + u_x + \sigma x^2\,L_{xx}\,u_x + \sigma y^2\,L_{xy}\,u_y + \tau^2\,L_{yt}\,v_t +$$
$$\sigma x^2\,L_{xy}\,v_x + v_y + \sigma y^2\,L_{yy}\,v_y,\ -L_{xt} - u\,L_{xx} - v\,L_{xy} - \tau^2\,L_{xxt}\,u_t - 2\,L_x\,u_x -$$
$$\sigma x^2\,L_{xxx}\,u_x - \sigma y^2\,L_{xxy}\,u_y - \tau^2\,L_{xyt}\,v_t - \sigma x^2\,L_{xxy}\,v_x - L_y\,v_x - L_x\,v_y - \sigma y^2\,L_{xyy}\,v_y,$$
$$-u\,L_{xy} - L_{yt} - v\,L_{yy} - \tau^2\,L_{xyt}\,u_t - \sigma x^2\,L_{xxy}\,u_x - L_y\,u_x - L_x\,u_y -$$
$$\sigma y^2\,L_{xyy}\,u_y - \tau^2\,L_{yyt}\,v_t - \sigma x^2\,L_{xyy}\,v_x - 2\,L_y\,v_y - \sigma y^2\,L_{yyy}\,v_y,$$
$$-L_{tt} - u\,L_{xt} - v\,L_{yt} - L_x\,u_t - \tau^2\,L_{xtt}\,u_t - L_t\,u_x - \sigma x^2\,L_{xxt}\,u_x -$$
$$\sigma y^2\,L_{xyt}\,u_y - L_y\,v_t - \tau^2\,L_{ytt}\,v_t - \sigma x^2\,L_{xyt}\,v_x - L_t\,v_y - \sigma y^2\,L_{yyt}\,v_y,$$
$$-v\,L_x + u\,L_y + \tau^2\,L_{yt}\,u_t + \sigma x^2\,L_{xy}\,u_x + u_y + \sigma y^2\,L_{yy}\,u_y - \tau^2\,L_{xt}\,v_t - v_x -$$
$$\sigma x^2\,L_{xx}\,v_x - \sigma y^2\,L_{xy}\,v_y,\ v\,L_{xx} - u\,L_{xy} - \tau^2\,L_{xyt}\,u_t - \sigma x^2\,L_{xxy}\,u_x - L_y\,u_x -$$
$$L_x\,u_y - \sigma y^2\,L_{xyy}\,u_y + \tau^2\,L_{xxt}\,v_t + 2\,L_x\,v_x + \sigma x^2\,L_{xxx}\,v_x + \sigma y^2\,L_{xxy}\,v_y,$$
$$v\,L_{xy} - u\,L_{yy} - \tau^2\,L_{yyt}\,u_t - \sigma x^2\,L_{xyy}\,u_x - 2\,L_y\,u_y - \sigma y^2\,L_{yyy}\,u_y +$$
$$\tau^2\,L_{xyt}\,v_t + \sigma x^2\,L_{xxy}\,v_x + L_y\,v_x + L_x\,v_y + \sigma y^2\,L_{xyy}\,v_y,$$
$$v\,L_{xt} - u\,L_{yt} - L_y\,u_t - \tau^2\,L_{ytt}\,u_t - \sigma x^2\,L_{xyt}\,u_x - L_t\,u_y - \sigma y^2\,L_{yyt}\,u_y +$$
$$L_x\,v_t + \tau^2\,L_{xtt}\,v_t + L_t\,v_x + \sigma x^2\,L_{xxt}\,v_x + \sigma y^2\,L_{xyt}\,v_y\}$$

All eight equations in this set have to vanish, leading to a simultaneous set of eight equations with eight unknowns. In the next section we test the procedure on a spatio-temporal sequence of known deformation.

▲ Task 17.3 Compare the above derived equations with the results of Otte and Nagel [Otte1994], by taking the limit of $\sigma_x \to 0$, $\sigma_y \to 0$ and $\tau \to 0$. For a detailed description of this comparison see [Florack1998a].

17.7 Testing the optic flow constraint equations

In order to test the multi-scale optic flow constraint equations, we have developed a convenient *package* OFCE.m, which defines the commands to calculate the first order flowfield, and to display the result as a vectorfield.

The package, written by A. Suinesiaputra, is a good example of how to make complex functions available in notebooks, and is read by:

```
<< FrontEndVision`OFCE`;
```

As a test stimulus for scalar flow, we create a movie of deforming vessels, which we take from the fundus image of figure 17.4. We take a 64x64 section around the fovea, where we have vessels in different directions. We use the same vectorfield for warping as in figure 17.4.

```
im = Take[in = Import["fundus256.gif"][[1, 1]],
   {128, 128 + 64}, {180, 180 + 64}];
{ydim, xdim} = Dimensions[im];
DisplayTogetherArray[
  {ListDensityPlot[in, Epilog → {RGBColor[1, 1, 1], Line[{{180, 128},
        {180 + 64, 128}, {180 + 64, 128 + 64}, {180, 128 + 64}, {180, 128}}]]},
   ListDensityPlot[im]}, ImageSize -> 300];
```

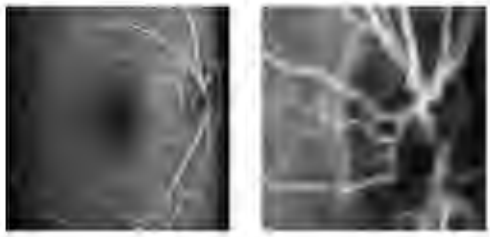


Figure 17.7 For the scalar optic flow test image we select a 64x64 pixel subregion around the fovea of the fundus image.

This generates the warping sequence:

```
vecf = Table[
   N[-{Sin[(π x) / xdim], Cos[(y π) / ydim]}], {y, 1, ydim}, {x, 1, xdim}];
stim = Table[deform2D[im, i vecf], {i, 0, 4, 1 / 5}];
```

We then calculate the first order optic flow vectorfield for scalar images, with spatial scale $\sigma = 2.0$ pixels and temporal scale $\tau = 1.0$ pixel. The function **FirstOFCE[]** is defined in the package **MOFCE.m**.

```
opticflowScalar =
  FirstOFCE[stim, 2.0, 1.0, FlowType → scalar, Singularities → ZeroVector];

Calculate first order optic flow at (σ,τ)=(2.,1.)

create scalar OFCE matrix .....

create OFCE result vector .....

....solving....
```

The stimulus and the vectorfield plots are calculated and displayed, where we omit the first and last two images of the series, because of boundary effects:

```
stimulusplot = ListDensityPlot[#, PlotRange → {Min[stim], Max[stim]},
    DisplayFunction -> Identity] & /@ stim; vectorfieldplot =
 VectorPlot[#, HeadScaleFactor → 2, Sampling → {3, 3}, AspectRatio → 1,
    PlotRange → {{1, 64}, {1, 64}}, ColorFunction → Automatic,
    DisplayFunction → Identity] & /@ Take[20 opticflowScalar, {4, 14}];

Show[#, PlotRange → {{1, 64}, {1, 64}},
    DisplayFunction → $DisplayFunction, ImageSize -> 220] & /@
  Transpose[{Take[stimulusplot, {4, 14}], vectorfieldplot}];
```

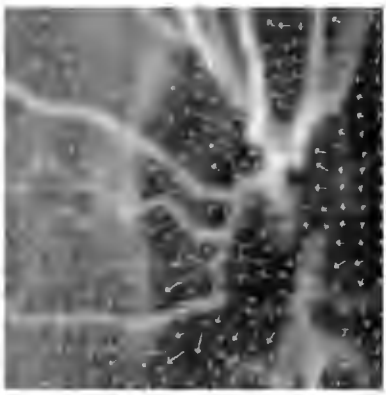


Figure 17.8 The resulting vectorfield for the scalar optic flow test sequence of a warped fundus image. The vectors have been plotted 20 times their actual length, to show the direction and magnitude better. Note that most velocities and their directions are found correctly, and that the normal constraint is the main reason for some (well understood) deviation.

17. 8 Cleaning up the vector field

The option **Averaging** in the function **VectorPlot** denotes the size of a small neighborhood over which the velocity vectors are uniformly averaged. This value must be a nonnegative odd integer. When the **Average** window size is 5 we get a much smoother vectorfield:

```
vplot3 = VectorPlot[20 #, Sampling → {2, 2}, HeadScaleFactor → 2,
     AspectRatio → 1, Averaging → 5, ColorFunction → Automatic,
     DisplayFunction → Identity] & /@ Take[opticflowScalar, {3, 14}];
Show[#, PlotRange → {{0, 65}, {0, 65}}, Background → RGBColor[0, 0, 0],
    DisplayFunction → $DisplayFunction, ImageSize -> 200] & /@
  Transpose[{Take[stimulusplot, {3, 14}], vplot3}];
```

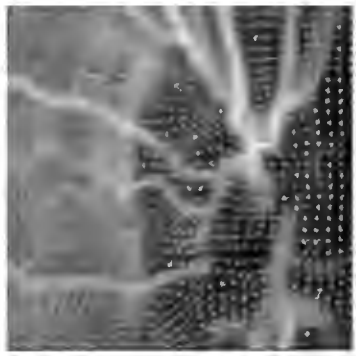


Figure 17.9 The vectorfield for the scalar optic flow test sequence of figure 17.8 with vector averaging over a 5 x 5 region. Note the much smoother result.

▲ 17.4 Show that with a 25 x 25 averaging area the original smoothing vector field is recovered. Why is this so?

▲ 17.5 Show the result with a 25 x 25 averaging area for a much faster varying vector field deformation.

However, averaging the vector field may introduce deviations from the true normal motion. A better way is to weight the surround of a vector with a Gaussian function, and multiply with a penalty function given by $\mathrm{Exp}[-\frac{K_n}{m}]$, where K_n is a normalized matrix condition number and m is just a constant. We set m = 0.1. The value of $K_n = \frac{K_{\max}}{K_{\min}}$ where $K_{\max}$ and $K_{\min}$ are the largest and smallest Eigenvalue of the matrix, so K_n will be in the range of (0..1|.

```
opticflowScalar = FirstOFCE[stim, 2.0, 1.0,
   FlowType → scalar, Singularities → ZeroVector, Smoothing -> True];

Calculate first order optic flow at (σ,τ)=(2.,1.)

create scalar OFCE matrix .....

create OFCE result vector .....

....solving....

integration of scale space ....

vplot1 =
  VectorPlot[20 #, Sampling → {2, 2}, HeadScaleFactor → 2, AspectRatio → 1,
     ColorFunction → Automatic, DisplayFunction → Identity] & /@
   Take[opticflowScalar, {4, 14}];

Show[#, PlotRange → {{0, 65}, {0, 65}}, Background → RGBColor[0, 0, 0],
    DisplayFunction → $DisplayFunction, ImageSize -> 200] & /@
  Transpose[{Take[stimulusplot, {4, 14}], vplot1}];
```

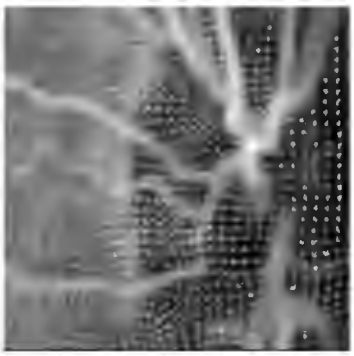


Figure 17.10 The vectorfield for the scalar optic flow test sequence of figure 17.8 with a weighted penalty based on the condition number of the matrix.

17.9 Scale selection

One of the major problems in making an effective extraction of the optic flow vectors, is the appropriate choice of *scale*. This is a general and fundamental issue in scale-space theory. Selection of scale must be treated at two levels:

First of all it is always present in the *task* we perform on the image. Do we want to segment the leaves or the tree? A careful specification of the image analysis task should make the choice of scale explicit. Secondly, we can look for automatic scale selection, by optimizing some criterion over a range of scales. For example, in feature detection one might look for the maximized output amplitude of some feature detector for each location (see Lindeberg for a detailed discussion).

For our optic flow we have a different criterion: in stead of the generation of a (sophisticated) filter output, we solve locally a set of equations, which is a *process*. When the matrix of coefficients in the matrix equation becomes singular, we know that the solution cannot be found. In other words, the further we are off from singularity, the better we can solve the set of equations. This means that we need some measure of how far we are from singularity. From linear algebra we know that the *condition number* of a matrix is a proper choice. There are a number of possibilities [from Ruskeepää1999, section 17.3.4]:

```
normL1[x_] := Max[(Plus @@ Abs[#]) & /@ Transpose[x]];
normLinf[x_] := Max[(Plus @@ Abs[#]) & /@ x];
normL2[x_] := Max[SingularValues[N[x]][[2]]];
```

The L_1-norm (**normL1**) is the largest of the absolute column sums, and the L_∞-norm (**normLinf**) is the largest of the absolute row sums. The L_2-norm (**normL2**) is the maximum of the singular values.

SingularValues[N[x]] gives the singular value decomposition, and the second component of this decomposition contains the singular values. This norm can be written as

```
normL2b[x_] := √Max[Abs[Eigenvalues[Conjugate[Transpose[x]].x]]];
```

because the norm is also the square root of the largest absolute eigenvalue of the matrix **Conjugate[Transpose[x]].x**. In fact, the singular values are simply the square roots of the eigenvalues of this matrix.

We use the *Frobenius norm* defined as:

```
FrobeniusNorm[mat_] := Module[{a, λ = DeleteCases[
      Re[√Eigenvalues[Transpose[Conjugate[mat]].mat] ], 0.]},
   a = Plus @@ λ^2; Check[If[a < 10^-50 || a^-1 === Indeterminate ||
      a^-1 === ComplexInfinity, $MaxMachineNumber, Plus @@ λ^-2], mat]];
```

We choose here for the maximum norm of the coefficient matrix over scale. Does this norm display a maximum over scale anyway? We calculate the norm for 22 spatial scales, ranging from $\sigma = 0.8$ to $\sigma = 5$ pixels in steps of 0.2, for the eighth frame in the sequence (**f=8**).

```
matL2 = Table[Lx = gDn[stim, {0, 0, 1}, {1, σ, σ}];
Ly = gDn[stim, {0, 1, 0}, {1, σ, σ}];
mat[f_] := ( Lx[[f]]  Ly[[f]]
             Ly[[f]]  -Lx[[f]] );
  Map[FrobeniusNorm, Transpose[mat[8], {3, 4, 1, 2}], {2}]
  , {σ, .8, 5, .2}]; Dimensions[matL2]

{22, 65, 65}
```

Indeed quite a few pixel locations show a maximum, when we plot the logarithm of the norm as a function of scale index for all pixels:

```
DisplayTogether[Table[
   LogListPlot[Transpose[matL2, {3, 2, 1}][[i, j]], PlotJoined -> True,
    PlotRange -> {.01, 500}, AxesLabel -> {"scale index", "norm"}],
   {i, 1, 64, 10}, {j, 1, 64, 10}], ImageSize -> 290];
```

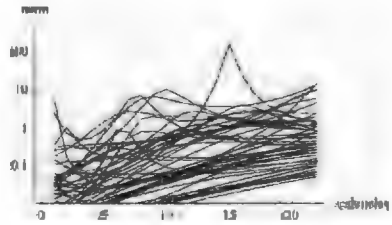


Figure 17.11 Logarithmic plot of the L_2 norm in every pixel as a function of spatial scale (range: σ = 0.8 to 5.0 pixels) for the eighth frame in the retinal fundus sequence.

The plot of the scale index as a function of location in the frame is indicative for what happens: a small scale is best at locations with image structure, such as the location of the main vessels. In more homogeneous areas a larger scale gives a better condition number.

```
ShowLegend[DisplayTogetherArray[
   {ListDensityPlot[stim[[8]]], ListDensityPlot[-Log[matL2[[8]]]]},
   DisplayFunction -> Identity], {GrayLevel[1 - #] &, 10, "σ=3.0",
   "σ=0.8", LegendPosition → {1.1, -.4}}, ImageSize -> 400];
```

Figure 17.12 Left: eighth frame of the image sequence. Right: scale index for the maximum singular value of the coefficient matrix of the optic flow equations, for the scale range σ = 0.8 to 3 pixels. Note the smaller scales for the locations rich with image structure, such as the vessels, and the use of larger scales for the more homogeneous areas.

- Task 17.6 Investigate the behaviour of other definitions of condition numbers.

- Task 17.7 Investigate the influence of the temporal scale.

- Task 17.8 Investigate the influence of both the spatial scales and the temporal scale simultaneously.

- Task 17.9 Implement scale selection by selecting in each pixel the scale with the largest Frobenius norm.

17.10 Discussion

Many authors have proposed a multi-scale approach to the determination of optic flow. Weber and Malik [Weber1995b] used a filtered differential method.

They applied a set of filters of various spectral contents, and also got a set of equations which can be solved to get a unique solution for the optic flow. Fleet and Jepson [Fleet1990] extract image component velocities from local phase information. Weickert recently introduced an approach for discontinuity preserving flow determination [Weickert1998d], and a method using a variational (energy minimizing) approach [Weickert2001a].

So far, we have extracted the derivatives in the temporal direction in the same way as in the spatial direction(s), through convolution with a Gaussian derivative kernel with an appropriate temporal scale. For a pre-recorded temporal sequence this is fine. When the Gaussian kernel extends over the border of the image, the image can be periodically extended, as we did for the spatial case. In a real-time situation however, the only information available is the past, and we cannot use the half of the Gaussian kernel that extends 'into the future'. This problem has been elegantly solved by Jan Koenderink, who proposed to resample the temporal axis with a logarithmic reparametrization, based on arguments of temporal *causality*. This is discussed in detail in chapter 20.

Of course, a local method where a measurement is done through an aperture, faces the *aperture property* (for a nice set of demo's of the aperture 'problem' see www.illusionworks.com). A common extra constraint is the 'smoothness constraint, where the local vector is considered in its relation to its direct vicinity, and should not change much over a small distance. A natural step in the context of multi-scale optic flow extraction is the use of deep structure scale space theory: the singularity points in the deep structure have no aperture problem, they form a set of natural intrinsic and hierarchically ordered multi-scale image pointset.

17.11 Summary of this chapter

Two main features stand out in the treatment of optic flow in this chapter. The classical Horn & Schunck optic flow constraint equation is brought 'under the aperture', as it should for observed variables. The application of scaled derivatives brings the approach under the scale-space paradigm.

Secondly, the notion of *Lie-derivatives* of the image intensity time series with respect to some unknown vectorfield enables us to form a set of equations from which the unknown parameters of the flow field can be calculated. The aperture problem is the consequence of the fact that we in typical cases have more unknowns then equations. However, when *additional* physical constraints are given, such as the constraint of normal or expanding flow, the solution can be found exact. The theory enables the solution for different orders of the vector field approximation.

Earlier tests of the method have shown that this physics based scale-space approach outperforms all other methods. The main reason must be the full inclusion of the physical model of the *observed* flow.

18. Color differential structure

Jan-Mark Geusebroek, Bart M. ter Haar Romeny, Jan J. Koenderink, Rein van den Boomgaard, Peter Van Osta

18.1 Introduction

Color is an important extra dimension. Information extracted from color is useful for almost any computer vision task, like segmentation, surface characterization, etc. The field of *color science* is huge [Wyszecki2000], and many theories exist. It is far beyond the scope of this book to cover even a fraction of the many different approaches. We will focus on a single recent theory, based on the color sensitive receptive fields in the front-end visual system. We are especially interested in the extraction of multi-scale differential structure in the spatial *and* the color domain of color images. This scale-space approach was recently introduced by Geusebroek et al. [Geusebroek1999a, Geusebroek2000a], based on the pioneering work of Koenderink's Gaussian derivative color model [Koenderink1998a]. This chapter presents the theory and a practical implementation of the extraction of color differential structure.

18.2 Color image formation and color invariants

What is color invariant structure? To understand that notion, we first have to study the process of color image formation.

```
<< FrontEndVision`FEV`;
p1 = Plot[Sin[.06 λ] + Sin[.1 λ] + 5, {λ, 350, 700}, PlotRange -> {2, 8},
   ImageSize -> 275, Ticks -> {Automatic, None}, AspectRatio -> .4];
p2 = DensityPlot[λ, {λ, 350, 700}, {y, 0, 2}, AspectRatio -> .1,
   ColorFunction -> (Hue[-0.001964 # + 1.375] &),
   ColorFunctionScaling -> False, Frame -> False, ImageSize -> 275];
```

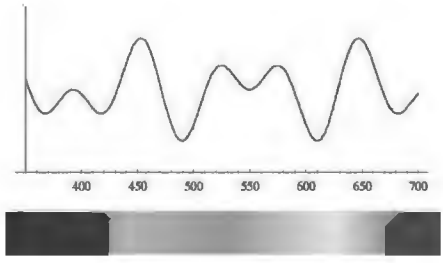


Figure 18.1 An arbitrary spectrum (distribution of the energy over the wavelengths) reflected from an object. For different colored objects we have different spectra. Horizontal scale: nm, vertical scale: energy (W/m^2).

Figure 18.1 shows an arbitrary spectrum that may fall onto the eye (or camera). The spectrum as reflected by an object is the result of light falling onto the object, of which part of the spectral energy is reflected towards the observer. Hence, the light spectrum falling onto the eye results from interaction between a light source, the object, and the observer. Color may be regarded as the measurement of spectral energy, and will be handled in the next section. Here, we only consider the interaction between light source and material.

Before we see an object as having a particular color, the object needs to be illuminated. After all, in darkness objects are simply black. The emission spectra $l(\lambda)$ of common light sources are close to Planck's formula [Wyszecki 1999]

```
h = 6.626176 10^-34; c = 2.99792458 10^8; k = 1.38066 10^-23;
l[λ_, T_] := 8 π h c λ^-5 (E^(h c / (k T λ)) - 1)^-1
```

where h is Planck's constant, k Boltzmann's constant, and c the velocity of light in vacuum. The *color temperature* of the emitted light is given by T, and typically ranges from 2,500K (warm red light) to 10,000K (cold blue light). Note that the terms "warm" and "cold" are given by artists, and refer to the sensation caused by the light. Representative white light is, by convention, chosen to be at a temperature of 6500K. However, in practice, all light sources between 2,500K and 10,000K can be found. Planck's equation is adequate for incandescent light and halogen. The spectrum of daylight is slightly different, and is represented by a *correlated color temperature*. Daylight is close enough to the Planckian spectrum to be characterized by a equivalent parameter.

The part of the spectrum reflected by a surface depends on the surface *spectral reflection function*. The spectral reflectance is a material property, characterized by a function $c(\lambda)$. For planar, matte surfaces, the spectrum reflected by the material $e(\lambda)$ is simply the multiplication between the spectrum falling onto the surface $l(\lambda)$ and the surface spectral reflectance function $c(\lambda)$: $e(\lambda) = c(\lambda)\, l(\lambda)$. For example, figure 18.2 shows the emission spectrum of three lightsources, resp. of 2500K, 4500K and 10000K.

```
Plot[{l[λ 10^-9, 2500], l[λ 10^-9, 4500], 1/50 l[λ 10^-9, 10000]},
  {λ, 0, 2000}, PlotRange -> All, ImageSize -> 200];
```

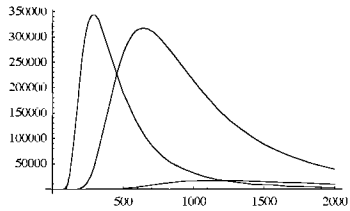


Fig 18.2 Emission spectrum of a black body light source with a temperature of resp. 2500K, 6500K and 10,000K. The emission at 10,000K is much larger than at the 2 lower temperatures, and for that reason, is plotted at 1/50 of its amplitude.

Now that we have the spectral reflectance function, we can examine how the reflected spectrum would look with a different light source. In figure 18.3 the reflected spectrum for a 2500K, 4500K and a 10,000K radiator is demonstrated.

```
Block[{$DisplayFunction = Identity},
  plots = Plot[1[λ 10^-9, #] (Sin[.06 λ] + Sin[.1 λ] + 5),
      {λ, 350, 700}, PlotRange → All, ImageSize → 300,
      Ticks → {Automatic, None}] & /@ {2500, 4500, 10000}];
Show[GraphicsArray[plots], ImageSize -> 500];
```

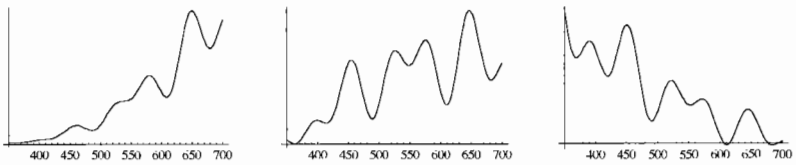


Fig 18.3 Object reflectance function for the observed spectrum shown in figure 18.1 for a resp. 2500K, 6500K and 10,000K light source. As opposed to the reflected spectrum (fig. 18.1), this object reflectance function is a material property, independent of the illumination source.

At this point it is meaningful to introduce spatial extent, hence to describe the spatio-spectral energy distribution $e(x,y,\lambda)$ that falls onto the retina. Further, for three-dimensional objects the amount of light falling onto the objects surface depends on the energy flux, thus on the local geometry. Hence *shading* (and shadow) may be introduced as being a wavelength independent multiplication factor $m(x,y)$ in the range [0...1]: $e(x, y, \lambda) = c(x, y, \lambda)\, l(\lambda)\, m(x, y)$.

Note that the illumination $l(\lambda)$ is independent of position. Hence the equation describes spectral image formation of matte objects, illuminated by a single light source. For shiny surfaces the image formation equation has to be extended with an additive term describing the Fresnel reflected light, see [Geusebroek2000b] for more details.

The *structure* of the spatio-spectral energy distribution is due to the three functions $c(.)$, $l(.)$, and $m(.)$. By making some general assumptions, these quantities may be derived from the measured image. Estimation of the object reflectance function $c(.)$ boils down to deriving material properties, the "true" color invariant which does not depend on illumination conditions. Estimation of the light source $l(.)$ is well known as the color constancy problem. Determining $m(.)$ is in fact estimating the shadows and shading in the image, and is closely related to the shape-from-shading problem.

For the extraction of color invariant properties from the spatio-spectral energy distribution we search for algebraic or differential expressions of $e(.)$, which are independent of $l(.)$ and $m(.)$. Hence the goal is to solve for differential expressions of $e(.)$ which results in a function of $c(.)$ only.

To proceed, note that the geometrical term m is only a function of spatial position. Differentiation with respect to λ, and normalization reduces the problem to only two functions: $e(x, \lambda) = c(x, \lambda) l(\lambda) \Rightarrow \frac{1}{e(x,\lambda)} \frac{\partial e(x,\lambda)}{\partial \lambda} = \frac{l_\lambda}{l} + \frac{c_\lambda}{c}$ (indices indicate differentiation). After additional differentiation to the spatial variable x or y, the first term vanishes, since $l(.)$ only depends on λ:

```
e[x, λ] = c[x, λ] l[λ];
∂x (∂λ e[x, λ] / e[x, λ])
```

```
0
```

The left-hand side, after applying the chain rule,

```
∂x (∂λ e[x, y, λ] / e[x, y, λ]) // shortnotation
```

$$\frac{e[x, y, \lambda]\, e_{xz}[x, y, \lambda] - e_x[x, y, \lambda]\, e_z[x, y, \lambda]}{e[x, y, \lambda]^2}$$

is completely expressed in spatial and spectral derivatives of the observable spatio-spectral energy distribution.

As an example in this chapter we develop the differential properties of the invariant color-edge detector $\mathcal{E} = \frac{1}{e} \frac{\partial e}{\partial \lambda}$, where the measured spectral intensity $e = e(x, y, \lambda)$. Spatial derivatives of $\mathcal{E}$, like $\frac{\partial \mathcal{E}}{\partial x}$, contain derivatives to the spatial as well as to the wavelength dimension due to the chain rule. In the next section we will see that the zero-th, first and second order derivative-to-λ kernels are acquired from the transformed RGB space of the image directly. The derivatives to the spatial coordinates are acquired in the conventional way, i.e. convolution with a spatial Gaussian kernel.

18.3 Koenderink's Gaussian derivative color model

We have seen in the previous chapters how spatial structure can be extracted from the data in the environment by measuring the set of (scaled) derivatives to some order. For the spatial domain this has led to the family of Gaussian derivative kernels, sampling the spatial intensity distribution. These derivatives naturally occur in a local Taylor expansion of the signal.

Koenderink proposed in 1986 to take a similar approach to the sampling of the color dimension, i.e. the spectral information contained in the color. If we construct the Taylor expansion of the spatio-spectral energy distribution $e(x, y, \lambda)$ of the measured light to wavelength, in the fixed spatial point (x_0, y_0), and around a central wavelength λ_0 we get (to second order):

```
Series[e[x0, y0, λ], {λ, λ0, 2}]
```

$$e[x0, y0, \lambda0] + e^{(0,0,1)}[x0, y0, \lambda0]\ (\lambda - \lambda0) + \frac{1}{2} e^{(0,0,2)}[x0, y0, \lambda0]\ (\lambda - \lambda0)^2 + O[\lambda - \lambda0]^3$$

We recall from chapter 1 that a physical measurement with a aperture is mathematically described with a convolution. So for a measurement of the luminance L with aperture function $G(x, \sigma)$ in the (here in the example 1D) spatial domain we get: $L(x; \sigma) = \int_{-\infty}^{\infty} L(x - \alpha)\, G(\alpha, \sigma)\, d\alpha$ where α is the dummy spatial shift parameter running over all possible values.

For the temporal domain we get $L(t; \sigma) = \int_{-\infty}^{\infty} L(t - \beta)\, G(\beta, \sigma)\, d\beta$ where β is the dummy temporal shift parameter running over all possible values in time (in chapter 20 we will take a close look at this temporal convolution). Based on this analogy, we might expect a measurement along the color dimension to look like: $L(\lambda; \sigma) = \int_{-\infty}^{\infty} L(\lambda - \gamma)\, G(\gamma, \sigma)\, d\gamma$ where λ is the wavelength and γ is the dummy wavelength shift parameter.

The front-end visual system has implemented the shifted spatial kernels with a grid on the retina with receptive fields, so the shifting is implemented by the simultaneous measurement of all the neighboring receptive fields. The temporal kernels are implemented as time-varying LGN and cortical receptive fields (explained in detail in chapters 11 and 20). However, in order to have a wide range of receptive fields which shift over the wavelength axis in sensitivity, would require a lot of different photo-sensitive dyes (rhodopsins) in the receptors with these different -shifted- color sensitivities.

The visual system may have opted for a cheaper solution: The convolution is calculated at just a single position on the wavelength axis, at around $\lambda_0 = 520$ nm, with a standard deviation of the Gaussian kernel of about $\sigma_\lambda = 55$ nm. The integration is done over the range of wavelengths that is covered by the rhodopsins, i.e. from about 350 nm (blue) to 700 nm (red). The values for λ_0 and σ_λ are determined from the best fit of a Gaussian to the spectral sensitivity as measured psychophysically in humans, i.e. the Hering model.

So we get for the spectral intensity $e(\vec{x}, \lambda_0; \sigma_\lambda) = \int_{\lambda_{min}}^{\lambda_{max}} e(\vec{x}, \lambda)\, G(\lambda, \lambda_0, \sigma_\lambda)\, d\lambda$. This is a 'static' convolution operation. It is not a convolution in the familiar sense, because we don't shift over the whole wavelength axis. We just do a *single* measurement with a Gaussian aperture over the wavelength axis at the position λ_0. Similarly, the derivatives to λ

$\frac{\partial e(\vec{x},\lambda_0)}{\partial \lambda} = \sigma_\lambda \int_{\lambda_{min}}^{\lambda_{max}} e(\vec{x}, \lambda)\, \frac{\partial G(\lambda,\lambda_0,\sigma_\lambda)}{\partial \lambda}\, d\lambda$ and
$\frac{\partial^2 e(\vec{x},\lambda_0)}{\partial \lambda^2} = \sigma_\lambda^2 \int_{\lambda_{min}}^{\lambda_{max}} e(\vec{x}, \lambda)\, \frac{\partial^2 G(\lambda,\lambda_0,\sigma_\lambda)}{\partial \lambda^2}\, d\lambda$
describe the first and second order spectral derivative respectively. The factors σ_λ and σ_λ^2 are included for the normalization, i.e. to make the Gaussian spectral kernels dimensionless.

Here are the graphs of the 'static' normalized Gaussian spectral kernels to second order as a function of wavelength:

```
gaussλ[λ_, σ_] = D[gauss[λ, σ], λ];
gaussλλ[λ_, σ_] = D[gauss[λ, σ], {λ, 2}];
λ0 = 520; σλ = 55;
Plot[{gauss[(λ - λ0), σλ], σλ gaussλ[(λ - λ0), σλ],
   σλ^2 gaussλλ[(λ - λ0), σλ]}, {λ, λ0 - 3 σλ, λ0 + 3 σλ},
  PlotRange → All, AxesLabel -> {"λ (nm)", ""}, ImageSize -> 250];
```

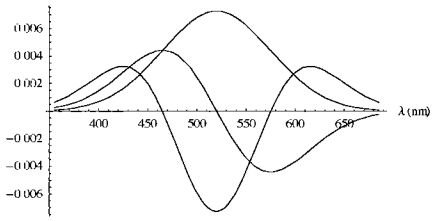


Figure 18.4 The zero-th, first and second derivative of the Gaussian function with respect to wavelength as models for the color receptive field's wavelengths sensitivity in human color vision. After [Koenderink 1986]. The central wavelength is 520 nm, the standard deviation 55 nm.

We recall from the human vision chapter that the color sensitive receptive fields come in the combinations red-green and yellow-blue center-surround receptive fields. The subtraction of yellow and blue in these receptive fields is well modeled by the first order derivative to λ, the subtraction of red and green minus the blue is well modeled by the second order derivative to λ . Alternatively, one can say that the zero-th order receptive field measures the luminance, the first order the 'blue-yellowness', and the second order the 'red-greenness'. The sensitivity of the three types of cones (L-, M- and S-cones: long, medium and short wavelength sensitivity) is given in figure 18.5.

```
Show[Import["ConesColors.gif"], ImageSize -> 200];
```

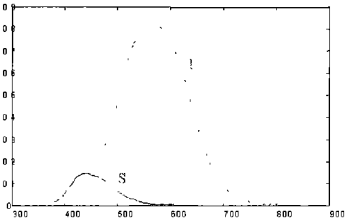


Figure 18.5 The relative spectral sensitivity of the three types of cones in the human retina. From [Geusebroek et al. 1999a].

The Gaussian model approximates the Hering basis [Hering64a] for human color vision when $\lambda_0 \simeq 520\,\text{nm}$ and $\sigma_\lambda \simeq 55\,\text{nm}$ (see figure 18.6). They also fit very well the CIE 1964 XYZ basis, which is a famous coordinate system for colors, much used in technical applications involving color.

Note: the wavelength axis is a half axis. It is known that for a *half axis* (such as with positive-only values) a logarithmic parametrization is the natural way to 'step along' the axis. E.g. the scale axis is logarithmically sampled in scale-space (remember the 'orders of magnitudes'), the intensity is logarithmically transformed in the photoreceptors, and, as we will see in chapter 20, the time axis can only be measured causally when we sample it logarithmically. We might conjecture here a better fit to the Hering model with a logarithmic wavelength axis.

```
Show[Import["HeringColormodel.gif"], ImageSize -> 300];
```

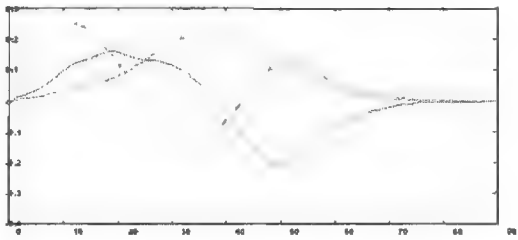


Figure 18.6 The Hering basis for the spectral sensitivity of human color receptive fields. From [Hering1964a].

The Gaussian color model needs the first three components of the Taylor expansion of the Gaussian weighted spectral energy distribution at λ_0 and scale σ_λ. An RGB camera measures the red, green and blue component of the incoming light, but this is *not* what we need for the Gaussian color model. We need a method to extract the Taylor expansion terms from the RGB values. The figure below shows the RGB color space. The black color is diagonally opposite the white cube.

```
sc = .2; n = 6; Clear[gr];
Show[gr = Graphics3D[Table[{RGBColor[x / n, y / n, z / n],
     Cuboid[{x - sc, y - sc, z - sc}, {x + sc, y + sc, z + sc}]},
    {z, 1., n}, {y, 1., n}, {x, 1., n}], Lighting -> False],
 ViewPoint -> {2.441, 1.600, 1.172}, ImageSize -> 200];
```

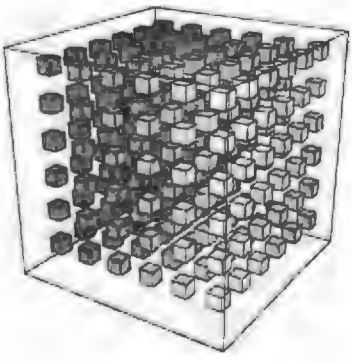


Figure 18.7 RGB color space.

LiveGraphics3D is a Java class written by Martin Kraus to real-time rotate and manipulate *Mathematica* Graphics3D objects in a browser. The package is available at

wwwvis.informatik.uni-stuttgart.de/~kraus/LiveGraphics3D/. With the command **liveFormWrite** we write the above graphic to a temporary file.

```
liveFormWrite[$TopDirectory <>
  "\\AddOns\\Applications\\FrontEndVision\\LiveGraphics3D\\data.m", gr]
```

Hit show.html to start your browser and play with the 3D structure.

This plots all pixels of a color image as points (with their RGB color) in RGB space:

```
Block[{$DisplayFunction = Identity, im, data},
  im = Import["hybiscus2.jpg"]; data = Flatten[im[[1, 1]], 1];
  p1 = Show[im, ImageSize -> 150];
  p2 = Show[Graphics3D[{RGBColor @@ #/255, Point[#]} & /@ data],
    Axes -> True, AxesLabel -> {"R", "G", "B"}]];
Show[GraphicsArray[{p1, p2}], ImageSize -> 470];
```

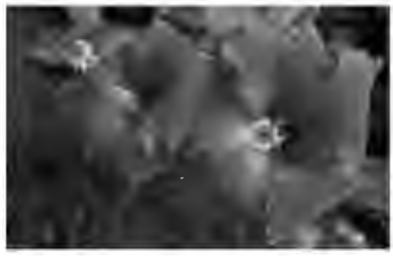


Figure 18.8 Left: color input image. Right: distribution of the pixels in the RGB space. Note the wide range of green colors (in the left cluster) and red colors (in the right cluster) due to the many shadows and lighting conditions.

An RGB camera approximates the CIE 1964 XYZ basis for colorimetry by the following linear transformation matrix:

$$\text{rgb2xyz} = \begin{pmatrix} 0.621 & 0.113 & 0.194 \\ 0.297 & 0.563 & 0.049 \\ -0.009 & 0.027 & 1.105 \end{pmatrix};$$

Geusebroek et al. [Geusebroek2000a] give the best linear transform from the XYZ values to the Gaussian color model:

$$\text{xyz2e} = \begin{pmatrix} -0.019 & 0.048 & 0.011 \\ 0.019 & 0. & -0.016 \\ 0.047 & -0.052 & 0. \end{pmatrix};$$

The resulting transform from the measured RGB input image to the sampling 'à la human vision' is the dot product of the transforms:

```
colorRF = xyz2e.rgb2xyz; colorRF // MatrixForm
```

$$\begin{pmatrix} 0.002358 & 0.025174 & 0.010821 \\ 0.011943 & 0.001715 & -0.013994 \\ 0.013743 & -0.023965 & 0.00657 \end{pmatrix}$$

Indeed, when we study this transform and plot the rows, we see the likeliness with the Gaussian derivative sensitivity (see figure 18.6).

```
Block[{$DisplayFunction = Identity},
  pl = Table[ListPlot[colorRF[[i]], PlotJoined -> False,
      PlotStyle -> PointSize[0.05], Ticks → None], {i, 3}]];
Show[GraphicsArray[pl], ImageSize -> 200];
```

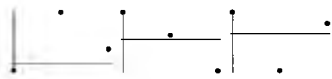


Figure 18.9 The transformations of the input RGB values to the Gaussian color model coarsely resemble the Gaussian derivatives to λ. Left: zero-th order. Middle: first order, right: second order. The three center wavelengths roughly correspond with 400, 475 and 575 nm of the Gaussian derivative functions of figure 18.4.

The set of spatio-spectral Gaussian derivative cortical simple cells thus looks like (from [Geusebroek et al. 2000a]):

```
Show[Import["ColorRFs.gif"], ImageSize -> 250];
```

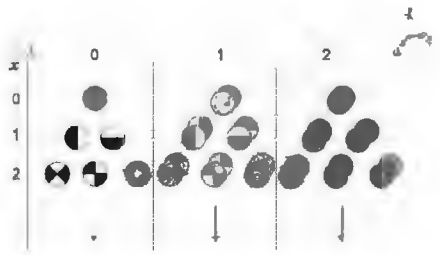


Figure 18.10The Gaussian color model for cortical receptive fields. Left: zero-th order to wavelength, measuring the luminance and the spatial derivative structure. Middle: first order derivative to wavelength, yellow/blue - spatial derivatives. Right: second order derivative to wavelength, red/green-spatial derivatives.

The Gaussian color model is an approximation, but has the attractive property of fitting very well into Gaussian scale-space theory. The notion of image structure is extended to the wavelength domain in a very natural and coherent way. The similarity with human differential-color receptive fields is more than a coincidence.

Now we have all the tools to come to an actual implementation. The RGB values of the input image are transformed into Gaussian color model space, and plugged into the spatio-spectral formula for the color invariant feature.

Next to the derivatives to wavelength we need spatial derivatives, which are computed in the regular way with spatial Gaussian derivative operators. The full machinery of e.g. gauge coordinates and invariance under specific groups of transformations is also applicable here. The next section details the implementation.

18.4 Implementation

We start with the import of an RGB color image (see figure 18.11). The color pixels are RGB triples in the very first element of the imported object:

```
image = Import["colortoys2.jpg"];
im = image[[1, 1]];
im[[1]] // Short
```

```
{{67, 104, 148}, {67, 104, 148}, {66, 103, 147}, {66, 103, 147},
 {66, 103, 147}, {66, 103, 147}, <<167>>, {119, 40, 87}, {119, 40, 87},
 {120, 41, 88}, {119, 40, 87}, {119, 40, 87}, {120, 41, 88}}
```

```
Dimensions[im]
```

```
{228, 179, 3}
```

The RGB triples are converted into measurements through the color receptive fields in the retina with the transformation matrix **colorRF** defined above:

```
colorRF = xyz2e.rgb2xyz; colorRF // MatrixForm
```

$$\begin{pmatrix} 0.002358 & 0.025174 & 0.010821 \\ 0.011943 & 0.001715 & -0.013994 \\ 0.013743 & -0.023965 & 0.00657 \end{pmatrix}$$

To transform every RGB triple we map the transformation to our input image as a pure function at the second listlevel:

```
observedimage = Map[Dot[colorRF, #] &, im, {2}];
```

The three 'layers' of this observed image **obs** represent resp. e, e_λ and $e_{\lambda\lambda}$. We 'slice' the dataset smartly by a reordering **Transpose** whereby the e-, e_λ- and $e_{\lambda\lambda}$-values each form a plane:

```
obs = Transpose[observedimage, {2, 3, 1}];
Dimensions[obs]
```

```
{3, 228, 179}
```

Let us inspect what the color receptive fields see:

```
DisplayTogetherArray[
  Prepend[ListDensityPlot /@ obs, Show[image]], ImageSize -> 470];
```

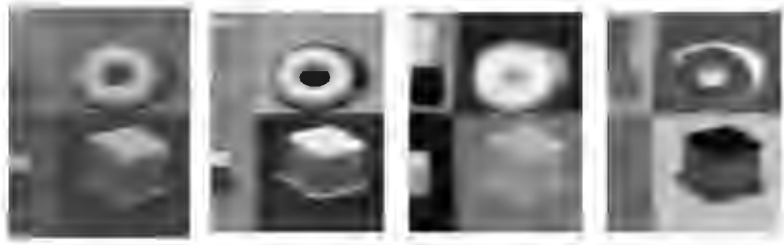


Figure 18.11 The input image (left) and the observed images with the color differential receptive fields. Image resolution 228x179 pixels.

We now develop the differential properties of the invariant color-edge detector $\mathcal{E} = \frac{1}{e}\frac{\partial e}{\partial \lambda}$, where the spectral intensity $e = e(x, y, \lambda)$. The derivatives to the spatial and spectral coordinates are easily found with the chainrule. Here are the explicit forms:

```
ℰ := D[e[x, y, λ], λ] / e[x, y, λ]; ∂x ℰ
```

$$-\frac{e^{(0,0,1)}[x, y, \lambda]\, e^{(1,0,0)}[x, y, \lambda]}{e[x, y, \lambda]^2} + \frac{e^{(1,0,1)}[x, y, \lambda]}{e[x, y, \lambda]}$$

To make this more readable, we apply *pattern matching* to make the expression shorter (for the code of **shortnotation** see FEV.nb)

```
∂x ℰ /. e[x, y, λ] -> e // shortnotation
```

$$\frac{e\, e_{xz}[x, y, \lambda] - e_x[x, y, \lambda]\, e_z[x, y, \lambda]}{e^2}$$

```
∂y ℰ /. e[x, y, λ] -> e // shortnotation
```

$$\frac{e\, e_{yz}[x, y, \lambda] - e_y[x, y, \lambda]\, e_z[x, y, \lambda]}{e^2}$$

```
∂λ ℰ /. e[x, y, λ] -> e // shortnotation
```

$$\frac{-e_z[x, y, \lambda]^2 + e\, e_{zz}[x, y, \lambda]}{e^2}$$

```
∂x,λ ℰ /. e[x, y, λ] -> e // shortnotation
```

$$\frac{1}{e^3}\left(e^2\, e_{xzz}[x, y, \lambda] - 2\, e\, e_{xz}[x, y, \lambda]\, e_z[x, y, \lambda] + e_x[x, y, \lambda]\left(2\, e_z[x, y, \lambda]^2 - e\, e_{zz}[x, y, \lambda]\right)\right)$$

```
∂y,λ ℰ /. e[x, y, λ] -> e // shortnotation
```

$$\frac{1}{e^3}\left(e^2\, e_{yzz}[x, y, \lambda] - 2\, e\, e_{yz}[x, y, \lambda]\, e_z[x, y, \lambda] + e_y[x, y, \lambda]\left(2\, e_z[x, y, \lambda]^2 - e\, e_{zz}[x, y, \lambda]\right)\right)$$

The gradient magnitude (detecting yellow-blue transitions) becomes:

```
G = Simplify[Sqrt[(∂x ℰ)^2 + (∂y ℰ)^2]] // shortnotation
```

$$\sqrt{\frac{1}{e[x,y,\lambda]^4}\left((e[x,y,\lambda]\,e_{xz}[x,y,\lambda]-e_x[x,y,\lambda]\,e_z[x,y,\lambda])^2+(e[x,y,\lambda]\,e_{yz}[x,y,\lambda]-e_y[x,y,\lambda]\,e_z[x,y,\lambda])^2\right)}$$

The second spectral order gradient (detecting purple-green transitions) becomes:

```
W = Simplify[Sqrt[(∂x,λ ℰ)^2 + (∂y,λ ℰ)^2]] // shortnotation
```

$$\sqrt{\frac{1}{e[x,y,\lambda]^6}\left((e[x,y,\lambda]^2\,e_{xzz}[x,y,\lambda]+2\,e_x[x,y,\lambda]\,e_z[x,y,\lambda]^2-e[x,y,\lambda]\,(2\,e_{xz}[x,y,\lambda]\,e_z[x,y,\lambda]+e_x[x,y,\lambda]\,e_{zz}[x,y,\lambda]))^2+(e[x,y,\lambda]^2\,e_{yzz}[x,y,\lambda]+2\,e_y[x,y,\lambda]\,e_z[x,y,\lambda]^2-e[x,y,\lambda]\,(2\,e_{yz}[x,y,\lambda]\,e_z[x,y,\lambda]+e_y[x,y,\lambda]\,e_{zz}[x,y,\lambda]))^2\right)}$$

Finally, the total edge strength N (for all color edges) in the spatio-spectral domain becomes:

```
N = Simplify[Sqrt[(∂x ℰ)^2 + (∂y ℰ)^2 + (∂x,λ ℰ)^2 + (∂y,λ ℰ)^2]]; N // shortnotation
```

$$\sqrt{\frac{1}{e[x,y,\lambda]^6}\left(e[x,y,\lambda]^2\,(e[x,y,\lambda]\,e_{xz}[x,y,\lambda]-e_x[x,y,\lambda]\,e_z[x,y,\lambda])^2+e[x,y,\lambda]^2\,(e[x,y,\lambda]\,e_{yz}[x,y,\lambda]-e_y[x,y,\lambda]\,e_z[x,y,\lambda])^2+(e[x,y,\lambda]^2\,e_{xzz}[x,y,\lambda]+2\,e_x[x,y,\lambda]\,e_z[x,y,\lambda]^2-e[x,y,\lambda]\,(2\,e_{xz}[x,y,\lambda]\,e_z[x,y,\lambda]+e_x[x,y,\lambda]\,e_{zz}[x,y,\lambda]))^2+(e[x,y,\lambda]^2\,e_{yzz}[x,y,\lambda]+2\,e_y[x,y,\lambda]\,e_z[x,y,\lambda]^2-e[x,y,\lambda]\,(2\,e_{yz}[x,y,\lambda]\,e_z[x,y,\lambda]+e_y[x,y,\lambda]\,e_{zz}[x,y,\lambda]))^2\right)}$$

As an example, we implement this last expression for discrete images.

As we did in the development of the multi-scale Gaussian derivative operators, we replace each occurrence of a derivative to λ with the respective plane in the observed image **rf** (by the color receptive fields). Note that we use **rf[[nλ+1]]** because the zero-th list element is the **Head** of the list. We recall the internal representation of a derivative in *Mathematica*:

```
FullForm[e^(1,0,2)[x, y, λ]]

Derivative[1, 0, 2][e][x, y, \[Lambda]]
```

We will look for such patterns and replace them with another pattern. We do this *pattern matching* with the command **/.** (**ReplaceAll**). We call the observed image at this stage **rf**, without any assignment to data, so we can do all calculations symbolically first:

```
Clear[rf0, rf1, rf2, σ];
rf = {rf0, rf1, rf2};
```

```
N = N /. {Derivative[nx_, ny_, nλ_][e][x, y, λ] :>
    Derivative[nx, ny][rf[[nλ + 1]]][x, y],
  e[x, y, λ] :> rf[[1]]} // Simplify
```

$$\sqrt{\left(\frac{1}{rf0^6}\left(rf0^2\,(rfl[x,y]\,rf0^{(0,1)}[x,y]-rf0\,rfl^{(0,1)}[x,y])^2+\right.\right.}$$
$$(2\,rfl[x,y]^2\,rf0^{(0,1)}[x,y]-2\,rf0\,rfl[x,y]\,rfl^{(0,1)}[x,y]+$$
$$rf0\,(-rf2[x,y]\,rf0^{(0,1)}[x,y]+rf0\,rf2^{(0,1)}[x,y]))^2+$$
$$rf0^2\,(rfl[x,y]\,rf0^{(1,0)}[x,y]-rf0\,rfl^{(1,0)}[x,y])^2+$$
$$(2\,rfl[x,y]^2\,rf0^{(1,0)}[x,y]-2\,rf0\,rfl[x,y]\,rfl^{(1,0)}[x,y]+$$
$$\left.\left.rf0\,(-rf2[x,y]\,rf0^{(1,0)}[x,y]+rf0\,rf2^{(1,0)}[x,y]))^2\right)\right)$$

Note that we do a delayed rule assignment here (:→ instead of →) because we want to evaluate the right hand side only after the rule is applied. We finally replace the spatial derivatives with our familiar spatial Gaussian derivative convolution **gD** at scale σ:

```
N = N /. {Derivative[nx_, ny_][rf_][x, y] :> gD[rf, nx, ny, σ],
   rfl[x, y] :> rfl, rf2[x, y] :> rf2}
```

$$\sqrt{\left(\frac{1}{rf0^6}\left(rf0^2\,(rfl\,gD[rf0,0,1,\sigma]-rf0\,gD[rfl,0,1,\sigma])^2+\right.\right.}$$
$$rf0^2\,(rfl\,gD[rf0,1,0,\sigma]-rf0\,gD[rfl,1,0,\sigma])^2+$$
$$(2\,rfl^2\,gD[rf0,0,1,\sigma]\quad 2\,rf0\,rfl\,gD[rfl,0,1,\sigma]+$$
$$rf0\,(-rf2\,gD[rf0,0,1,\sigma]+rf0\,gD[rf2,0,1,\sigma]))^2+$$
$$(2\,rfl^2\,gD[rf0,1,0,\sigma]-2\,rf0\,rfl\,gD[rfl,1,0,\sigma]+$$
$$\left.\left.rf0\,(-rf2\,gD[rf0,1,0,\sigma]+rf0\,gD[rf2,1,0,\sigma]))^2\right)\right)$$

This expression can now safely be calculated on the discrete data:

```
{rf0, rfl, rf2} = obs; σ = 1.;
ListDensityPlot[-N, PlotRange -> {-.4, 0}, ImageSize -> 220];
```

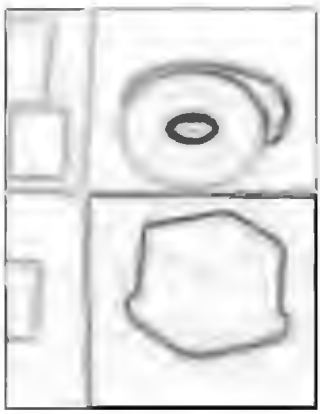


Figure 18.12 The color-invariant N calculated for our input image at spatial scale σ = 1 pixel. Primarily the total color edges are found, with little edge detection at intensity edges. Image resolution 228x179 pixels.

The following routines are implemented as standard FEV functions:

`Ɛ[im,σ]`	$\mathcal{E} = \frac{1}{e}\frac{\partial e}{\partial \lambda}$	color invariant $\frac{1}{e}\frac{\partial e}{\partial \lambda}$
`Ɛλ[im,σ]`	$\frac{\partial \mathcal{E}}{\partial \lambda}$	first wavelength derivative of $\mathcal{E}$
`Ɛλλ[im,σ]`	$\frac{\partial^2 \mathcal{E}}{\partial \lambda^2}$	second wavelength derivative of $\mathcal{E}$
`gG[im,σ]`	$\sqrt{(\frac{\partial \varepsilon}{\partial x})^2 + (\frac{\partial \varepsilon}{\partial y})^2}$	yellow-blue edges
`gW[im,σ]`	$\sqrt{(\frac{\partial^2 \varepsilon}{\partial x \partial \lambda})^2 + (\frac{\partial^2 \varepsilon}{\partial y \partial \lambda})^2}$	red-green edges
`gN[im,σ]`	$\sqrt{(\frac{\partial \varepsilon}{\partial x})^2 + (\frac{\partial \varepsilon}{\partial y})^2 + (\frac{\partial^2 \varepsilon}{\partial x \partial \lambda})^2 + (\frac{\partial^2 \varepsilon}{\partial y \partial \lambda})^2}$	total color edge strength

The prefix `g` stands for the multi-scale Gaussian derivative implementation. For the code, see appendix D.

Here is an example of finding blue-yellow edges:

```
image = Import["terre indigo.jpg"]; im = rasterpixels[image];
DisplayTogetherArray[
  {Show[image], ListDensityPlot[-gG[im, 1], PlotRange → {-.15, -.08}]},
  ImageSize → 510];
```

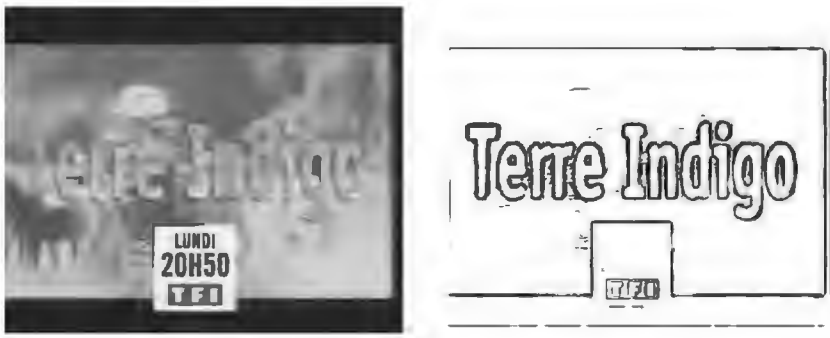


Figure 18.13 The yellow-blue edge detector $\mathcal{G} = \sqrt{(\frac{\partial \varepsilon}{\partial x})^2 + (\frac{\partial \varepsilon}{\partial y})^2}$ calculated at a spatial scale σ = 1 pixel. Image resolution 288x352. Note the absence of black-white intensity edges in the day and time indication. Example due to J. Sporring.

This example shows the color-selectivity of the spatio-color differential invariants:

```
image = Import["colortoys.jpg"]; im = rasterpixels[image];

DisplayTogetherArray[{Show[image],
   ListDensityPlot[-gG[im, σ = 1.], PlotRange → {-.2, -.06}],
   ListDensityPlot[-gW[im, σ = 1.], PlotRange → {-1.2, 0}]},
  ImageSize → 510];
```

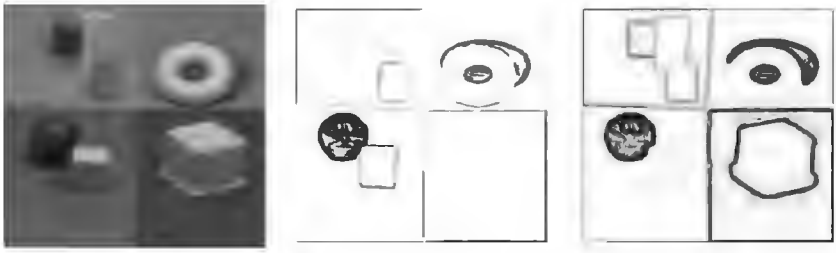


Figure 18.14 Detection of the yellow-blue edge detector $\mathcal{G} = \sqrt{\left(\frac{\partial \varepsilon}{\partial x}\right)^2 + \left(\frac{\partial \varepsilon}{\partial y}\right)^2}$ (middle) and the red-green edge detector $\mathcal{W} = \sqrt{\left(\frac{\partial^2 \varepsilon}{\partial x \partial \lambda}\right)^2 + \left(\frac{\partial^2 \varepsilon}{\partial y \partial \lambda}\right)^2}$ (right) at a scale of $\sigma = 1$ pixel.

As a histological example, spatio-color differential structure can accentuate staining patterns. The next example shows the advantage of a specific red-green detection in the localization of the edges of a stained nucleus in *paramecium caudatum*, a common freshwater inhabitant with ciliary locomotion.

```
im = rasterpixels[image = Import["Paramecium caudatum.jpg"]];
DisplayTogetherArray[
  {Show[image], ListDensityPlot[-gG[im, σ = 3.], PlotRange → {-.012, 0}],
   ListDensityPlot[-gW[im, σ = 3.], PlotRange → {-.15, 0}]},
  ImageSize → 510];
```

Figure 18.15 The color-invariant $\mathcal{G}$ (middle) and $\mathcal{W}$ (right) calculated for a histological image of a fungus cell, *paramecium caudatum* (left). The red-green color edges found by $\mathcal{W}$ form a good delineation of the cell's nucleus.

18.5 Combination with spatial constraints

Interesting combinations can be made when we combine the color differential operators with the spatial differential operators. E.g. when we want to detect specific blobs with a specific size and color, we can apply feature detectors that are best matching the shape to be found. We end the chapter with two examples: color detection of blobs in a regular pattern, and locating stained nuclei in a histological preparation.

Blobs are detected by calculating those locations (pixels) where the Gaussian curvature **lgc** = $L_{xx}L_{yy} - L_{xy}^2$ on the black-and-white version (**imbw**) of the image is greater then zero. This indicates a convex 'hilltop'. Pixels on the boundaries of the 'hilltop' are detected by requiring the second order directional derivative in the direction of the gradient L_{ww} to be positive. Interestingly, by using these invariant shape detectors we are largely independent of image intensity. For the color scheme we rely on $\mathcal{E}$ and its first and second order derivative to λ. The *Mathematica* code gives some examples of specific color detection:

```
imbw = colorToBW[im = rasterpixels[image = Import["colorblobs.gif"]]];
lww = gauge2DN[imbw, 0, 2, 2];
lgc = gD[imbw, 2, 0, 2] gD[imbw, 0, 2, 2] - gD[imbw, 1, 1, 2]^2;
el = Eλ[im, 2]; ell = Eλλ[im, 4];
t1 = Map[If[# > 0, 1, 0] &, lww, {2}];
t2 = Map[If[# > 0, 1, 0] &, lgc, {2}];
t3 = Map[If[# > 0, 1, 0] &, el, {2}];
t4 = Map[If[# > 0.001, 1, 0] &, ell, {2}];
t5 = Map[If[# > 0, 1, 0] &, el + ell, {2}];

Block[{$DisplayFunction = Identity},
  p1 = Show[image];
blue = ListDensityPlot[(1 - t1) t2 (1 - t5) (1 - (1 - t5) t3)];
yellow = ListDensityPlot[(1 - t1) t2 (1 - t4) t5];
redandmagenta = ListDensityPlot[(1 - t1) t2 t4]];
Show[
  GraphicsArray[{p1, blue, yellow, redandmagenta}], ImageSize → 420];
```

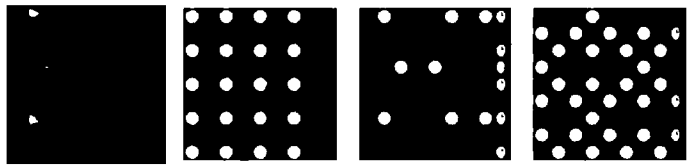


Figure 18.16 Detection of color blobs with spatial and color differential operators. Blobs are spatially described as locations with positive Gaussian curvature and positive second order directional derivative in the gradient direction of the image intensity, color is a boolean combination of the color differential features. Example due to P. van Osta.

▲ Task 18.1 Find the total detection scheme for all individual colors in the example above, and for combinations of two and three colors.

A last example detects stained carbohydrate deposits in a histological application.

```
imbw = colorToBW[im = rasterpixels[image = Import["pas.jpg"]]];
lww = gauge2DN[imbw, 0, 2, 4];
lgc = gD[imbw, 2, 0, 4] gD[imbw, 0, 2, 4] - gD[imbw, 1, 1, 4]^2;
el = Eλ[im, 4]; ell = Eλλ[im, 4]; t1 = Map[If[# > 0, 1, 0] &, lww, {2}];
t2 = Map[If[# > 0.2, 1, 0] &, lgc, {2}];
t6 = Map[If[# > 0, 1, 0] &, ell - el, {2}];
```

```
DisplayTogetherArray[
  {Show[image], ListDensityPlot[t1 t2 t6]}, ImageSize → 500];
```

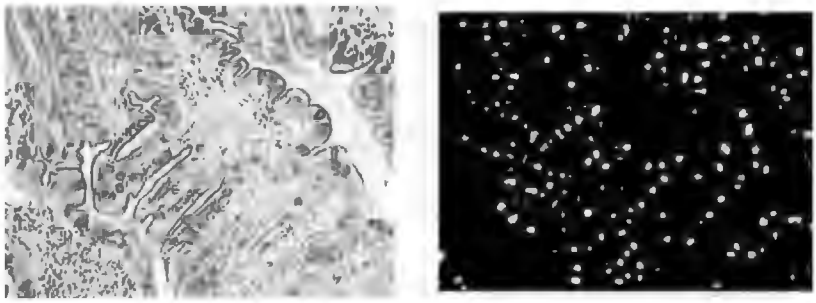


Figure 18.17 Detection of carbohydrate stacking in cells, that are specifically stained for carbohydrates with periodic acid Schiff (P.A.S.). The carbohydrate deposits are in magenta, cell nuclei in blue. The blob-like areas are detected with positive Gaussian curvature and positive L_{ww}, the magenta with a boolean combination of the color invariant $\mathcal{E}$ and its derivatives to λ. Example due to P. Van Osta. Image taken from http://www.bris.ac.uk/Depts/PathAndMicro/CPL/pas.html

18.6 Summary of this chapter

The scale-space model for color differential structure, as developed by Koenderink, states that the wavelength differential structure is measured at a single scale ($\sigma_\lambda = 55$ nm) for zeroth order (intensity), first order (blue-yellow) and second order (red-green). There is a good analogy with the color receptive field structures found in the mammalian visual cortex.

The chapter shows an actual implementation of this color differential structure. The wavelength derivatives can be extracted from the transformed RGB triples in the data.

In a similar way as for the spatial differential structure analysis of gray valued data, invariants can be defined, with combined color and spatial coordinate transformation invariance. A number of these color-spatial differential invariants are discussed and implemented on examples.

19. Steerable kernels

19.1 Introduction

Clearly, orientation of structure is a multi-scale concept. On a small scale, the local orientation of structural elements, such as edges and ridges, may be different from the orientations of the same elements at a larger scale. Figure 19.1 illustrates this.

```
<< FrontEndVision`FEV`
im = Import["fabric.gif"][[1, 1]];
DisplayTogetherArray[Prepend[
   (ListDensityPlot[gauge2DN[im0, 2, 0, #] /. im0 → im]) & /@ {1.5, 5},
   ListDensityPlot[im]], ImageSize -> 350];
```

Figure 19.1 The multi-scale nature of local orientation. Left: original, basket texture. Middle: at a small scale ($\sigma = 1.5$ pixels), orientation is governed by the fibers as is shown by the 'ridgeness' L_{vv}. Right: at a larger scale ($\sigma = 5$ pixels), an other orientation can be seen by calculating the L_{vv} ridgeness (bright areas, more or less horizontal in the picture).

Orientation plays an important role as parameter in establishing similarity relations between neighboring points. As such, it is an essential ingredient of methods for *perceptual grouping*. E.g. the grouping of edge pixels in a group that defines them as belonging to the same contour, could be done using similarity in orientation of the edges, i.e. of their respective gradient vectors. In chapter 12 we have seen that the visual system is particularly well equipped to take measurements of differential properties at a continuum of orientations:

the typical spokewheel structure of orientation columns that make up each hypercolumn (the wetware for the analysis of a single binocular visual field 'pixel') in the visual primary cortex, where the receptive fields were found at all orientations. In this chapter, we will study in detail the orientation properties of Gaussian derivative filters, and a concept named 'steerability'. We come to a proper definition of a directional derivative for all orders of differentiation, and how this can best be computed in a Cartesian framework. We then study as an application of orientation analysis the detection of stellate tumors in mammography (example taken from [Karssemeier1995a, Karssemeier1996a]), which is based on a global analysis of orientation. This is an example of computer-aided diagnosis (CAD).

Other examples of context dependent perception of curvature are some of the well known obscured parallel lines illusions, shown in figure 19.2.

```
square[x_, y_, s_] :=
  Line[{{x, y}, {x + s, y}, {x + s, y + s}, {x, y + s}, {x, y}}];
DisplayTogetherArray[
  {Show[Graphics[{Thickness[.01], Line[{{#, 0}, {0, #}}], If[EvenQ[#],
        Table[Line[{{# - n, n - 1}, {# - n, n + 1}}], {n, .1, 10, .5}],
        Table[Line[{{# - n - 1, n}, {# - n + 1, n}}], {n, .1, 10, .5}]]} & /@
      Range[1, 20, 3]], PlotRange -> {{0, 10}, {0, 10}}],
Show[Graphics[{Table[Line[{{-Cos[ϕ], -Sin[ϕ]}, {Cos[ϕ], Sin[ϕ]}}],
       {ϕ, 0, π, π / 16}], square[-.25, .25, .5], square[-.25, -.75, .5],
      Circle[{-.5, 0}, .25], Circle[{.5, 0}, .25]}],
    PlotRange -> {{-1, 1}, {-1, 1}}, AspectRatio -> Automatic]},
  ImageSize -> 400];
```

Figure 19.2 Obscured parallel lines illusions. Left: We perceive the lines as not parallel, in reality they are parallel. Right: the squares seem parallelograms, the circles seem to have an egg shape. The context around the contours of the figures determines the perceived orientation.

19.2 Multi-scale orientation

We define the *orientation* of a vector relative to a coordinate frame as the set of angles between the vector and with the frame vectors of the coordinate frame. Because we deal with orthogonal coordinates in this book, we need a single angle for a 2D vector, and two angles for a 3D vector.

Formal definition of angulation (2D), tilt and spin (3D). Figures.

The *direction* of a vector is the absolute value of the orientation. E.g. the direction of the x-axis is horizontal, and a identical direction is created when we rotate the x-axis over π radians (180 degrees).

In chapter 6,section 5, we studied the orientation of first order differential structure: the gradient vector field, and in chapter 6, section 7.3 we encountered the orientations of the principal curvatures, as the orientations of the Eigenvectors of the Hessian second order matrix.

19.3 Orientation analysis with Gaussian derivatives

In order to build the machinery for the extraction of orientation information, we need to fully understand the behavior of the Gaussian derivative operator at different orientations, and how to control this behavior. We will limit the analysis to 2D. The zeroth order Gaussian is isotropic by definition, and has no orientation. The action of this operator on an image is rotational invariant. All non-zero partial derivatives are equipped with a direction.

The first order Gaussian derivative kernel $\frac{\partial G}{\partial x}$ is shown in figure 19.3, and we define the orientation of this kernel to be zero, i.e. the angle ϕ of its axis along which the differentiation is done is zero radians with the x-axis. The orientation ϕ of the Gaussian derivative kernel to y, $\frac{\partial G}{\partial y}$, is $\pi/2$, pointing upwards along the positive y-axis. So increments in ϕ are positive in a counterclockwise fashion.

The first order Gaussian derivative kernel in another orientation can readily be made from its basic substituents: it is well known that a kernel with orientation ϕ can be constructed from $\mathrm{Cos}(\phi)\,\frac{\partial G}{\partial x} + \mathrm{Sin}(\phi)\,\frac{\partial G}{\partial y}$.

Figure 19.3 illustrates this on the convolution of the kernel on a Dirac-delta function, i.e. a single spike, which gives us the figure of the kernel, as we have seen in chapter 2:

```
im = Table[0, {128}, {128}]; im[[64, 64]] = 100;
ϕ = π / 6; Block[{$DisplayFunction = Identity},
 imx = gD[im, 1, 0, 15]; imy = gD[im, 0, 1, 15];
 imϕ = Cos[ϕ] gD[im, 1, 0, 15] + Sin[ϕ] gD[im, 0, 1, 15];
 pl = ListDensityPlot[#] & /@ {imx, imy, imϕ}];
Show[GraphicsArray[pl]];
```

Figure 19.3 A first order Gaussian derivative at any orientation can be constructed with the partial derivatives $\frac{\partial G}{\partial x}$ (left) and $\frac{\partial G}{\partial y}$ (middle) as a basis. Right: $\mathrm{Cos}(\phi)\,\frac{\partial G}{\partial x} + \mathrm{Sin}(\phi)\,\frac{\partial G}{\partial y}$, for $\phi = \pi/6$.

A class of filters where a filter of any orientation can be constructed from a linear combination of other functions is called a *steerable filter* [Freeman1991a]. The rotational components form a *basis*. A basis will contain more elements when we go to higher order. The question is now: what are the basis functions? How many basis functions do we need for the construction of a rotated Gaussian derivative of a particular order? We will discuss two important classes of basis functions (see figure 19.6):

a. basis functions that are rotated copies of the Gaussian derivative itself;
b. basis functions taken from the set of all partial derivatives in the Cartesian framework;

In particular, we will derive later in the text general formulas for e.g. the rotation of $\frac{\partial^3 G(x,y)}{\partial x^3}$ over 30 degrees clockwise ($-\pi/6$ radians). In the two bases the results are the same ($\frac{\partial^3 G(x,y)}{\partial x^3}\big|_{45^o}$ denotes the 45^o rotated version of the third derivative kernel):

$$\frac{\partial^3 G(x,y)}{\partial x^3}\Big|_{-30^o} = \frac{\sqrt{3}}{4}\frac{\partial^3 G(x,y)}{\partial x^3}\Big|_{0^o} + \frac{-3+\sqrt{3}}{4\sqrt{2}}\frac{\partial^3 G(x,y)}{\partial x^3}\Big|_{45^o} + \frac{1}{4}\frac{\partial^3 G(x,y)}{\partial x^3}\Big|_{90^o} - \frac{3+\sqrt{3}}{4\sqrt{2}}\frac{\partial^3 G(x,y)}{\partial x^3}\Big|_{135^o}$$

$$\frac{\partial^3 G(x,y)}{\partial x^3}\Big|_{-30^o} = -\frac{1}{8}\frac{\partial^3 G(x,y)}{\partial y^3} + \frac{3\sqrt{3}}{8}\frac{\partial^3 G(x,y)}{\partial x\,\partial y^2} + \frac{9}{8}\frac{\partial^3 G(x,y)}{\partial x^2\,\partial y} - \frac{3\sqrt{3}}{8}\frac{\partial^3 G(x,y)}{\partial x^3}$$

The first class will be derived in of basis functions may have interesting similarities in the control of directional derivatives in the visual system because of its self-similarity, symmetry and a reduced set of receptive field sensitivity profiles, the second is more apt for computer implementation. The derivation for the first class will be given in section 19.4, the second class will be derived in section 19.5. The two basis sets are different, and an illustrating example of multi-scale steerable filters.

19.4 Steering with self-similar functions

We need to know what is the necessary condition for a function to be steerable. We look at the first order Gaussian derivative as a first example. We express the kernel in polar coordinates $\{x, y\} = \{r\,\mathrm{Cos}(\phi),\ r\,\mathrm{Sin}(\phi)\}$ where r is the distance to the origin, and ϕ the angle with the real x-axis. With the function **TrigToExp** we express the result in complex exponentials $e^{i\phi}$:

```
Clear[ϕ]; {TrigToExp[Cos[ϕ] + I Sin[ϕ]], Exp[I ϕ] // ExpToTrig}
```

```
{e^(i ϕ), Cos[ϕ] + i Sin[ϕ]}
```

We transform our Gaussian kernel and some of its partial derivatives, and display the result, after division by the Gaussian kernel, in **MatrixForm** for notational clarity:

```
g := 1/(2 π σ^2) E^(-(x^2+y^2)/(2 σ^2)); ϕ = .;
t = {{g, ∂x g}, {∂y g, ∂x,y g}} /. {x → r Cos[ϕ], y → r Sin[ϕ]} // TrigToExp // Simplify;
t // MatrixForm
```

$$\begin{pmatrix} \frac{e^{-\frac{r^2}{2\sigma^2}}}{2\pi\sigma^2} & -\frac{e^{-\frac{r^2}{2\sigma^2}-i\phi}(1+e^{2i\phi})r}{4\pi\sigma^4} \\ \frac{ie^{-\frac{r^2}{2\sigma^2}-i\phi}(-1+e^{2i\phi})r}{4\pi\sigma^4\sigma^2} & -\frac{ie^{-\frac{r^2}{2\sigma^2}-2i\phi}(-1+e^{4i\phi})r^2}{8\pi\sigma^4\sigma^4} \end{pmatrix}$$

To see the structure more clearly, we divide the Gaussian itself out and collect the complex exponentials:

```
Collect[t/t[[1, 1]], E^(I ϕ)] // MatrixForm
```

$$\begin{pmatrix} 1 & -\frac{e^{-i\phi} r}{2\sigma^2} - \frac{e^{i\phi} r}{2\sigma^2} \\ -\frac{i e^{-i\phi} r}{2\sigma^2} + \frac{i e^{i\phi} r}{2\sigma^2} & \frac{e^{-2i\phi} r^2}{4\sigma^4} - \frac{e^{2i\phi} r^2}{4\sigma^4} \end{pmatrix}$$

The resulting function is separable in a radial part $a(r)$ and an angular part $e^{in\phi}$ where n is equal to the differential order: $f(r, \phi) = \sum_{j=1}^{M} a_n(r)\, e^{in\phi}$. A function with these properties *steers* when it can be written as rotated versions of itself.

The so-called *steerability constraint* is $f^{\theta}(x, y) = \sum_{j=1}^{M} k_j(\theta)\, f^{\theta_j}(x, y)$, where M is the number of filters, $f^{\theta}(x, y)$ is the rotated kernel, and the $f^{\theta_j}(x, y)$ are the rotated basis kernels. The weighting factors $k_j(\theta)$ are to be determined, as follows. When we fill in the polar representation of the kernel in the steerability constraint, we get

$$a_n(r)\, e^{in\phi} = \sum_{j=1}^{M} k_j(\theta)\, a_n(r)\, e^{in\phi}. \qquad (1)$$

We can divide by $a_n(r)$, and when $a_n(r) = 0$ for some n we just remove this constraint from the set of equations. The equations are equal for $-n$ and n, so we have to consider only angular frequencies in the range $0 < n < M$. Because n is equal to the order of differentiation, we have here as first result that the number of necessary basis filters for steerability in $n + 1$.

To steer the first order Gaussian derivative, we needed two basis functions, for steering a second order derivative kernel we need three basis functions. Equation (1) is a set of equations from which we can solve the weighting constants $k_j(\theta)$. Written more explicitly, they look like

$$\begin{pmatrix} 1 \\ e^{i\theta} \\ \cdot \\ e^{in\theta} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ e^{i\theta_1} & e^{i\theta_2} & \dots & e^{i\theta_M} \\ \cdot & \cdot & & \cdot \\ e^{in\theta_1} & e^{in\theta_2} & \dots & e^{in\theta_M} \end{pmatrix} \begin{pmatrix} k_1(\theta) \\ k_2(\theta) \\ \cdot \\ k_M(\theta) \end{pmatrix}.$$

The 1's in the top row are for $e^{i0\theta}$. Let us solve this set of equations.

We need to choose the orientations θ_j of the basis functions such that the columns in the matrix above are linearly independent. For symmetry reasons we choose them equally spaced over the space of directions, i.e. over the range $0 - \pi$. So, for the first order Gaussian derivative the basis orientations are 0 and $\pi/2$, for the second order they are 0, $\pi/3$ and $2\pi/3$, etc. Starting from an arbitrary angle ϕ we get $\{\phi, \phi + \pi/2\}$ and $\{\phi, \phi + \pi/3, \phi + 2\pi/3\}$ etc.

```
n = 2; θn = Table[ϕ + i π/(n + 1), {i, 0, n}]
```

$$\left\{\phi, \frac{\pi}{3} + \phi, \frac{2\pi}{3} + \phi\right\}$$

```
Exp[I 2 θn]
```

$$\left\{e^{2\,i\,\phi},\ e^{2\,i\,\left(\frac{\pi}{3}+\phi\right)},\ e^{2\,i\,\left(\frac{2\pi}{3}+\phi\right)}\right\}$$

Because the complex exponentials are complex, we have to solve them separately for both the real and the imaginary part.

```
n = 2; kj = Array[k, n + 1];
{ComplexExpand[Re[Exp[I n θ]] == Re[kj.Exp[I n θn]]],
 ComplexExpand[Im[Exp[I n θ]] == Im[kj.Exp[I n θn]]]}
```

$$\left\{\text{Cos}[2\,\theta] == \text{Cos}[2\,\phi]\,k[1] + \text{Cos}\left[2\left(\frac{\pi}{3}+\phi\right)\right]k[2] + \text{Cos}\left[2\left(\frac{2\,\pi}{3}+\phi\right)\right]k[3],\right.$$
$$\left.\text{Sin}[2\,\theta] == k[1]\,\text{Sin}[2\,\phi] + k[2]\,\text{Sin}\left[2\left(\frac{\pi}{3}+\phi\right)\right] + k[3]\,\text{Sin}\left[2\left(\frac{2\,\pi}{3}+\phi\right)\right]\right\}$$

For the even orders of differentiation we need to solve the equations specified by the top row and the even rows (if n is even, we start in row 0), for the odd orders we need the odd rows (if n is odd we start in row 1). The equation set is easily solved with *Mathematica:*

```
angularweights[n_] := Module[{k, i, ni}, Clear[θ, φ, k];
  θn = Table[φ + i π/(n + 1), {i, 0, n}]; kj = Array[k, n + 1];
sol = Solve[Flatten[
     Table[{ComplexExpand[Re[Exp[I ni θ]] == Re[kj.Exp[I ni θn]]],
       ComplexExpand[Im[Exp[I ni θ]] == Im[kj.Exp[I ni θn]]]},
      {ni, If[EvenQ[n], 0, 1], n, 2}]
], kj]; Flatten[kj /. sol] // Simplify // TrigReduce]
```

For the first order we find for the angular weights $k_j(\theta)$:

```
angularweights[1]
```

```
{Cos[θ - φ], Sin[θ - φ]}
```

Indeed, a correct result, found from these equations:

```
n = 1; Clear[θ, φ, k]; θn = Table[φ + i π/(n + 1), {i, 0, n}]; kj = Array[k, n + 1];
Table[{
  ComplexExpand[Re[Exp[I ni θ]] == Re[kj.Exp[I ni θn]]],
  ComplexExpand[Im[Exp[I ni θ]] == Im[kj.Exp[I ni θn]]]},
 {ni, If[EvenQ[n], 0, 1], n, 2}]
```

```
{{Cos[θ] == Cos[φ] k[1] - k[2] Sin[φ], Sin[θ] == Cos[φ] k[2] + k[1] Sin[φ]}}
```

For the second and third order Gaussian derivatives things get more complicated:

```
angularweights[2]
```

$$\left\{\frac{1}{3}\,(1+2\,\text{Cos}[2\,\theta-2\,\phi]),\ \frac{1}{3}\,(1-\text{Cos}[2\,\theta-2\,\phi]+\sqrt{3}\,\text{Sin}[2\,\theta-2\,\phi]),\right.$$
$$\frac{1}{9}\,((-1)^{1/6}\sqrt{3}-(-1)^{5/6}\sqrt{3}-(-1)^{1/6}\sqrt{3}\,\text{Cos}[2\,\theta-2\,\phi]+$$
$$(-1)^{5/6}\sqrt{3}\,\text{Cos}[2\,\theta-2\,\phi]-2\sqrt{3}\,\text{Sin}[2\,\theta-2\,\phi]-$$
$$\left.(-1)^{1/3}\sqrt{3}\,\text{Sin}[2\,\theta-2\,\phi]+(-1)^{2/3}\sqrt{3}\,\text{Sin}[2\,\theta-2\,\phi])\right\}$$

```
angularweights[3]
```

$$\left\{\frac{1}{2}\,(\text{Cos}[3\,\theta-3\,\phi]+\text{Cos}[\theta-\phi]),\right.$$
$$\frac{1}{4}\,(-\sqrt{2}\,\text{Cos}[3\,\theta-3\,\phi]+\sqrt{2}\,\text{Cos}[\theta-\phi]+\sqrt{2}\,\text{Sin}[3\,\theta-3\,\phi]+\sqrt{2}\,\text{Sin}[\theta-\phi]),$$
$$\frac{1}{2}\,(-\text{Sin}[3\,\theta-3\,\phi]+\text{Sin}[\theta-\phi]),$$
$$\left.\frac{1}{4}\,(\sqrt{2}\,\text{Cos}[3\,\theta-3\,\phi]-\sqrt{2}\,\text{Cos}[\theta-\phi]+\sqrt{2}\,\text{Sin}[3\,\theta-3\,\phi]+\sqrt{2}\,\text{Sin}[\theta-\phi])\right\}$$

We have found the general result for steerability of Gaussian derivative kernels.
A Gaussian derivative kernel can be steered, i.e. made in any orientation, by a linearly weighted sum of rotated versions of itself, the basis functions. There are $n+1$ functions required, equally spaced over an angle range of $0-\pi$.

For $n = 1$ the basis includes $\theta = 0°$ and $90°$;
For $n = 2$ the basis includes $\theta = 0°, 60°$ and $120°$;
For $n = 3$ the basis includes $\theta = 0°, 45°, 90°$ and $135°$;
For $n = 3$ the basis includes $\theta = 0°, 36°, 72°, 108°$ and $144°$; etc.
In other words: for each value of n we have a different set of basis functions.

```
ang2 = angularweights[2]; ang3 = angularweights[3];
ϕ = 0; Block[{$DisplayFunction = Identity},
 p1 = PolarPlot[#, {θ, 0, 2π},
      PlotRange→ {{-1, 1}, {-1, 1}}, Frame→ True] & /@ ang2;
 p2 = PolarPlot[#, {θ, 0, 2π}, PlotRange→ {{-1, 1}, {-1, 1}},
      Frame→ True] & /@ ang3];
Show[GraphicsArray[{p1, p2}], ImageSize→ 500];
```

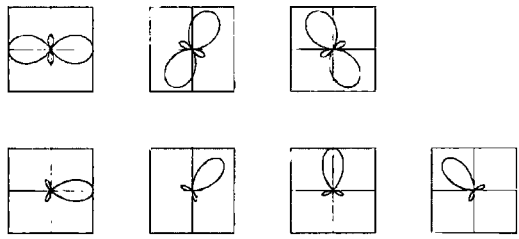


Figure 19.4 Top row: polar plot of the three coefficients to construct the rotated second order Gaussian derivative kernel. Bottom row: Polar plots of the 4 coefficients for the third order rotated derivative operator. A polar plot $f(\theta)$ is the radial plot of the radius f versus the angle θ.

The weights are found from a set of linear equations originating from the steeribility constraint. So the third order derivative $G_{xxx}|_{\phi}$ under an arbitrary angle of ϕ with the x-axis can be constructed from four basis kernels, each 45 degrees rotated, i.e.

$$\begin{aligned} G_{xxx}|_{\phi} = & \tfrac{1}{2}(\mathrm{Cos}[3\phi] + \mathrm{Cos}[\phi])\, G_{xxx}|_{0} \\ & + \tfrac{1}{4}\left(-\sqrt{2}\,\mathrm{Cos}[3\phi] + \sqrt{2}\,\mathrm{Cos}[\phi] + \sqrt{2}\,\mathrm{Sin}[3\phi] + \sqrt{2}\,\mathrm{Sin}[\phi]\right) G_{xxx}|_{\frac{\pi}{4}} \\ & + \tfrac{1}{2}(-\mathrm{Sin}[3\phi] + \mathrm{Sin}[\phi])\, G_{xxx}|_{\frac{\pi}{2}} \\ & + \tfrac{1}{4}\left(\sqrt{2}\,\mathrm{Cos}[3\phi] - \sqrt{2}\,\mathrm{Cos}[\phi] + \sqrt{2}\,\mathrm{Sin}[3\phi] + \sqrt{2}\,\mathrm{Sin}[\phi]\right) G_{xxx}|_{\frac{3\pi}{4}} \end{aligned}$$

When we fill in $\phi = -\pi/6$ we get the formula from the beginning of this chapter. It is instructive to look at the polar plots of these coefficients. This is the plot of the function as a function of the angle, with the amplitude of the function as distance to the origin. We quickly see then how the angular weights are distributed over the orientations, and we recognize the orientations of the basis functions, see figure 19.4.

19.5 Steering with Cartesian partial derivatives

When we just rotate our coordinates, we get a particular convenient representation for computer implementations. We use the same strategy as developed for the gauge coordinates in chapter 6. Instead of a locally adaptive set of directions for our frame to the orientation of the gradient vectorfield, we now choose a fixed rotation of our frame vectors.

We use the same formulas, and notice that the orientation of the {Lx, Ly} unit vector frame is given by {Sin(ϕ), Cos(ϕ)} where ϕ is our required angle of rotation:

```
Unprotect[gDϕ];
gDϕ[im_, nv_, nw_, σ_, ϕ_] :=
 Module[{Lx, Ly, v, w, im0}, v = {-Ly, Lx}; w = {Lx, Ly};
  Simplify[Nest[(v.{∂x #, ∂y #} &), Nest[(w.{∂x #, ∂y #} &), L[x, y], nw],
     nv] /. {Lx → Sin[ϕ], Ly → -Cos[ϕ]}]]
```

We denote `gDϕ[im_,nv_,nw_,σ_,ϕ_]` our rotated Gaussian derivative operator, and define this function in the appropriate way by repeated action of the differential operators (with **Nest**) and *thereafter* replacing the fixed direction {Lx, Ly} with the particular choice of {Sin(ϕ), $-$Cos(ϕ)}. A final step is the replacement of any appearance of a derivative into our regular (and now familiar) multi-scale Gaussian derivative operator `gD`.

Some examples: Here is the third derivative G_{xxx} rotated over $-\pi/6$ (30 degrees clockwise), the example of the beginning of this chapter, both in explicit formula and plotted at a scale of $\sigma = 15$ pixels (128^2 image):

```
im=.; gDϕ[im, 3, 0, 15, -π/6] // shortnotation
```

$$\frac{1}{8}\left(3\sqrt{3}\,L_{xxx} - 9\,L_{xxy} + 3\sqrt{3}\,L_{xyy} - L_{yyy}\right)$$

And a 4th order example:

```
gDϕ[im, 3, 1, 15, π / 5] // shortnotation
```

$$\frac{1}{32}\left(\left(\sqrt{50-10\sqrt{5}}+2\sqrt{10-2\sqrt{5}}\right)L_{xxxx}+8\,L_{xxxy}-6\sqrt{10-2\sqrt{5}}\,L_{xxyy}-8\sqrt{5}\,L_{xyyy}-\sqrt{50-10\sqrt{5}}\,L_{yyyy}\right)$$

Task 19.1 Show that L_{xxxy}, when rotated over $\pi/2$, gives L_{xyyy}. Explain this.

The numerical version for discrete images is **gDϕN**:

```
Unprotect[gDϕN];
gDϕN[im_, nv_, nw_, σ_, ϕ_] := gDϕ[im, nv, nw, σ, ϕ] /.
   Derivative[n_, m_][L][x, y] → gD[im0, n, m, σ] /. im0 -> im
im = Table[0, {128}, {128}]; im[[64, 64]] = 100;
DisplayTogetherArray[ListDensityPlot /@
   {gD[im, 3, 0, 15], gDϕN[im, 3, 0, 15, -π / 6]}, ImageSize -> 380];
```

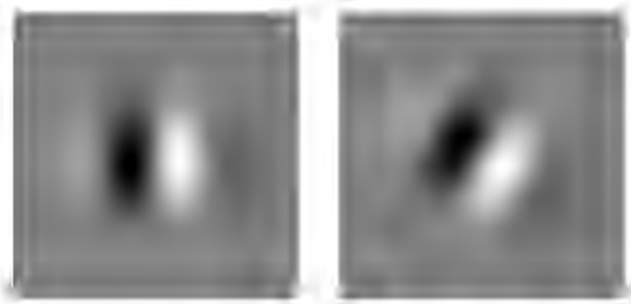


Figure 19.5 Left: the third order Gaussian derivative kernel to x. Right: the same kernel rotated 30 degrees clockwise, calculated from the expression in partial Cartesian derivatives above.

This is the proper multi-scale directional derivative. *Any* angle can now readily be constructed, no more need for 'only in 8 directions'.

To summarize this section we show the figures of the two different bases discussed. The first basis is a good starting point for models of oriented simple cells in the primary visual cortex. In chapter 9 we have seen that the orientation columns in the mammalian visual cortex contain the simple cells in a strikingly regular arrangement of ordered orientation. All orientations are present to analyze the local visual field in the hypercolumns, the arrangement in a pinwheel fashion. This representation is the topic of the last section of this chapter. The second basis, made up of Cartesian separable functions, is the basis of choice for computer implementations as these functions are just our familiar Gaussian derivative convolution kernels (**gD**). As we now have all tools for making the oriented kernels, we show both multi-scale steering bases for Gaussian derivative kernels below for the second order derivative.

```
im = Table[0, {128}, {128}]; im[[64, 64]] = 100;
Block[{$DisplayFunction = Identity},
n = 3; θn = Append[Table[iπ/(n+1), {i, 0, n}], -π/6];
p1 = ListDensityPlot[gDφN[im, 2, 0, 15, #]] & /@ θn;
p2 = Apply[ListDensityPlot[gD[im, #1, #2, 15]] &,
   {{3, 0}, {1, 2}, {2, 1}, {0, 3}}, 2];
t1 = Graphics[Text["\!\(\*
StyleBox[\"⇒\",\nFontFamily->\"Courier New
      \",\nFontSize->24,\nFontWeight->\"Plain\"]\)", {0, 0}]];
ttl = Insert[p1, t1, 5]]; tt2 = {p2, t1, p1[[5]]} // Flatten;
Show[GraphicsArray[{ttl, tt2}], ImageSize -> 350];
```

Figure 19.6 Two different sets of basis functions can be used for the construction of a steered Gaussian derivative kernel. Upper row: the basis set is formed from rotated versions of the kernel itself; bottom row: the basis set is formed from Cartesian x, y-separable Gaussian derivative kernels. The weights are calculated in the section above. Kernel as in figure 19.5.

19.6 Detection of stellate tumors

Computer-aided diagnosis (CAD) is the branch of computer vision in medical imaging concerned with the assistance of the computer in the diagnostic process. This is a rapidly growing field. There are two major reasons for its growth: a increasing array of methodologies outperform the human diagnosis in specificity, and the sheer volume of medical diagnostic data necessitates support. The system is always used as a 'second opinion' system, the final responsibility of the diagnosis is always laid on the medal specialist.

Recently, a number of commercial products have acquired FDA approval to put the system on the market and into clinical use. See e.g. the web pages of R2 Technology (www.r2tech.com), Deus Technologies (www.deustech.com) and Fujifilm (www.fujifilm.com). It is expected that more products soon will follow.

CAD up till now is mainly applied in fields where large scale screening of patients is performed. Two classical areas are screening for microcalcifications and stellate (stella = Latin: star; stellate = star-shaped) mammographic tumors (also called *spiculated* lesions), and screening for X-thorax deviations (tuberculosis screening, lung cancer etc.). We discuss the detection of stellate tumors in mammography. The following procedure has first been presented by Karssemeijer [Karssemeier1995a, Karssemeier1996a]. This is now the basis of one of the methods employed by R2 Technology.

A mammogram is a 2D X-ray photograph of the female breast, taken at high resolution (typically 2000^2 or higher) and with a low X-ray tube voltage (typically 28-35 kiloVolt), in

order to enhance the contrast between the soft tissue structures. The structure of the tissue is highly tubular: many channels are present, all converging in a tree-like structure to the nipple, so when a tumor expands, it is likely its outgrowth follows the channels. This gives often rise to a particular stellate pattern seen around the shadow of the lesion.

The *geometric reasoning:* when we investigate a pixel for the presence of a stellate structure, we actually have to investigate its immediate surrounding for the presence of lines oriented towards the pixel. Because we study a large group of surrounding pixels, we can approach a good statistical mean. The local orientation is detected by convolution of the second order Gaussian derivative, a 'classical bar-detector'. The oriented kernel is a function of L_{xx}, L_{xy} and L_{yy}:

```
Clear[φ, σ]; gDφ[im, 2, 0, σ, φ] // shortnotation
```

```
Cos[φ]^2 Lxx + Sin[2 φ] Lxy + Sin[φ]^2 Lyy
```

For each pixel in the neigborhood of the pixel for which we inspect a stellated surround, we calculated the three 'basis' derivatives L_{xx}, L_{xy} and L_{yy}. We create a kernel for the same area where each pixel indicates under which angle this pixel is located with respect to the central pixel. In this kernel we calculate again a triplet, now of the trigonometric coefficients of the second order oriented Gaussian derivative.

We use the speed of **ListCorrelate** to multiply all triplets in the kernel with the triplets of basis derivatives in the same area in the mammogram. The resulting area is somewhat smaller than the original image. It is not useful to take information into account from a periodic or mirrored boundary. The input image is **mammo**, at scale σ of the differential operator and **size** is the size of the search region:

```
stellatedetection[mammo_, σ_, size_] :=
  Module[{derivs, kernel}, derivs = Transpose[{-gD[mammo, 2, 0, σ],
      gD[mammo, 1, 1, σ], gD[mammo, 0, 2, σ]}, {3, 1, 2}];
kernel = Table[With[{φ = ArcTan[x, y]}, {Cos[φ]^2, Sin[2 φ], Sin[φ]^2}],
     {x, -size - 0.5, size + 0.5}, {y, -size - 0.5, size + 0.5}];
Flatten /@ ListCorrelate[kernel, derivs]];
```

We start with an artificial mammogram of 200x200 pixels, containing two stellate patterns with diameters of 30 and 20 pixels resp. (see figure 19.8, left), and uniformly distributed noise:

```
insert[mammo_, stellate_, x_, y_] :=
  Module[{ydim, xdim, x1, y1}, {ydim, xdim} = Dimensions[stellate];
   locsm = Flatten[Table[{y1, x1},
      {y1, y, y + ydim - 1}, {x1, x, x + xdim - 1}], 1];
   locss = Flatten[Table[{y1, x1}, {y1, 1, ydim}, {x1, 1, xdim}], 1];
   ReplacePart[mammo, stellate, locsm, locss]];

stellate[diam_] := Table[
   If[x == y || x == Round[diam / 2] || y == Round[diam / 2] || x == diam - y, 1, 0],
   {x, diam}, {y, diam}];
mammo = Table[0, {y, 200}, {x, 200}];
```

```
noise = Table[Random[], {200}, {200}];
mammo = insert[mammo, stellate[30], 50, 70];
mammo = insert[mammo, stellate[20], 120, 140] + noise;
```

We extract the n largest maxima with the following function, which was first defined in chapter 13:

```
nMaxima[im_, n_] := Module[{p, d = Depth[im] - 1},
   p = Times @@ Table[(Sign[im - Map[RotateLeft, im, {i}]] + 1)
        (Sign[im - Map[RotateRight, im, {i}]] + 1), {i, 0, d - 1}] / 4^d;
   maxs = Take[Reverse[Union[{10 Extract[im, #], Reverse[#]} & /@
        Position[p, 1]]], n];
   Apply[{10 #1 / maxs[[1, 1]], #2} &, maxs, {1}]];
```

The resulting detection is indicated with circles. The radii of the circles indicate the likelihood of being the center of a stellate region. The radii are normalized to the highest likelihood with a radius of 10 pixels.

```
{ydim, xdim} = Dimensions[mammo]; size = 20;
n = 2;
smammo = Take[mammo, {1 + size, ydim - size - 1}, {1 + size, xdim - size - 1}];
DisplayTogetherArray[{ListDensityPlot[smammo],
   ListDensityPlot[pl = stellatedetection[mammo, 4, 20], Epilog →
     {Red, Circle @@@ Reverse /@ nMaxima[pl, n]}]}, ImageSize → 450];
```

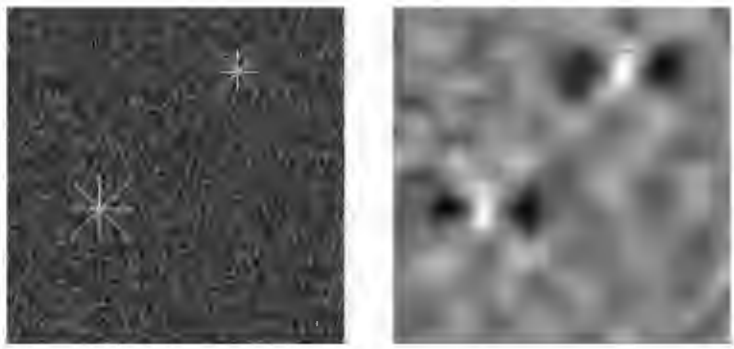


Figure 19.7 Computer-aided diagnosis in mammography: detection of two artificial stellate tumors in a noisy mammogram. Left: the input artificial mammogram (size 200x200 pixels) with 2 stellate regions. Right: the detected cumulative response of an oriented second order Gaussian derivative kernel, integrated over an area of 20x20 pixels.

▲ Task 19.2 Experiment with different values for the following parameters:
- the scale σ of the differential operator;
- the signal to noise ratio of the input image;
- the area of the search around each pixel;
- the distance between two stellate lesions;

- the size of the stellate lesion;
- the weight over the search area as a function of distance to the central pixel.

Now for a digital mammogram:

```
mammo = Import["mammogram04.jpg"][[1, 1]];
{ydim, xdim} = Dimensions[mammo];
detected = stellatedetection[mammo, σ = 3, size = 50]; n = 3;
smammo = Take[mammo, {size + 10, -size - 11}, {size + 10, -size - 11}];
sdetected = Take[detected, {10, -10}, {10, -10}];
circles = {Red, Circle @@@ Reverse /@ nMaxima[sdetected, n]};
DisplayTogetherArray[ListDensityPlot[#, Epilog → circles] & /@
   {smammo, sdetected}, ImageSize → 480];
```

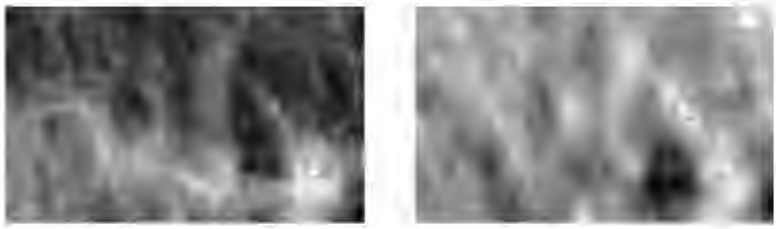


Figure 19.8 Left: Selection from a digital mammogram (size 403x291 pixels). Right: the detected cumulative response of an oriented second order Gaussian derivative kernel at $\sigma = 3$ pixels, integrated over an area of 50x50 pixels. The number of maxima to be reported is 3.

▲ Task 19.3 Experiment with other line detection kernels, such as a strongly elliptical oriented Gaussian kernel of zeroth order.

▲ Task 19.4 Find images with stellated tumors on the internet. See e.g. marathon.csee.usf.edu/Mammography/Database.html.

Note that this section only introduces the notion of using orientation sensitive responses in a *geometric reasoning* scheme, which might be part of a computer-aided diagnosis procedure. The method described above should never be used in any diagnostic judgement.

19.7 Classical papers and student tasks

The paper by Freeman and Adelson [Freeman1991a] formed the basis of this section. The references therein give a good overview of the literature on steerable filters. Other instructive (and historical) papers are by Jan Koenderink in his classical paper on receptive field models [Koenderink1988b], by Pietro Perona [Perona1991a, Perona1992, Perona1995], by Wolfgang Beil [Beil1994], by Per-Erik Danielson [Danielson1990] and Eduard Simoncelli and coworkers [Simoncelli1995, Simoncelli1996a, Farid1997a]. The construction of rotated partial derivatives can also be set up with Lie group theory, finding the Lie 'infinitesimal generators'. The papers by Michaelis and Sommer [Michaelis1995a, Michaelis1995b] and by Teo and Hel-Or [Teo1998] give a fine introduction, many examples and references of this powerful technique. One of the early theoretical studies of orientation tuning in the context of the front-end visual system is by Daughman [Daughman1983, Daughman1985]. See for an invertible orientation 'bundle' the paper by Kalitzin [Kalitzin1997a, Kalitzin1998a]. Multi-scale orientation analysis, based on models inspired by the 'spokewheel' structure as observed in the columns in the visual primary cortex, is a promising terrain for perceptual grouping research.

Task 19.5 Find the expressions for the weighting functions for a mixed Gaussian derivative kernel, e.g. $\frac{\partial^3 G}{\partial x^2 \partial y}$, for both bases.

Task 19.6 Pentland [Pentland1990] has suggested that shape-from-shading analysis can be performed by a linear filtering operation in many situations, e.g. when the reflectance function is approximately linear. Freeman and Adelson [Freeman1991a] give suggestions how to implement this with steerable functions. Make a *Mathematica* scale-space implementation for linear shape from shading.

Task 19.7 Make a *Mathematica* implementation, based on the results developed in this chapter, for 3D steerable filters. See again Freeman and Adelson [Freeman1991a] for theoretical support.

Task 19.8 Trabecular bone (the sponge-like interior of most of our bones) has an intricate multi-scale orientation structure. Extract from 2D and 3D datasets the local orientation structure at multiple scales, and come up with sensible definitions for the local structure of the bone. For inspiration, see [TerHaarRomeny1996f, Niessen1997b, Lopéz200a].

19.8 Summary of this chapter

The local vectorfield specified by the gradient and its clockwise rotated perpendicular vector form the first order orientation structure. This vectorfield is also specified by the Eigenvectors of the local *structure matrix*.
The local vectorfield specified by the unit vectors of the principal curvature vectors in each point form the second order orientation structure. This vectorfield is also specified by the Eigenvectors of the local *Hessian matrix*.
Gaussian derivative kernels are steerable kernels. They can be constructed in any direction (as directional derivative operators) in two ways: as a polynomial expressed in rotated versions of the Gaussian derivative kernel itself, or as a polynomial combination of Cartesian partial derivatives.
The geometric reasoning for the detection of structures can now be expanded to the inclusion of responses to oriented structures, such as lines. An example is given for the detection of stellate tumors in mammography.

20. Scale-time

"The dilemma is complete" -Jan Koenderink, [Koenderink1988a]
"It is later then you think" -Chinese proverb
"You are young and life is long and there is time to kill today" -Pink Floyd

20.1 Introduction

In the time domain we encounter sampled data just as in the spatial domain. E.g. a movie is a series of *frames*, samples taken at regular intervals. In the spatial domain we needed an integration over a spatial area to catch the information. Likewise, we need to have an aperture in time integrating for some time to perform the measurement. This is the *integration time*. Systems with a short resp. long integration time are said to have a fast resp. slow response. Because of the necessity of this integration time, which need to have a finite *duration* (temporal width) in time, a scale-space construct is a physical necessity again.

```
<< FrontEndVision`FEV`;
Show[Import["DaVinci watch 512x540.jpg"], ImageSize -> 160];
```

Figure 20.1 This watch Da Vinci Rattrapante (IWC Schaffhausen, Switzerland), named after Leonardo da Vinci who did many inventions in the area of clocks, indicates time at many different time scales, i.e. seconds, minutes, hours, weekdays, day of months, months, years and lunar cycle.

Furthermore, time and space are *incommensurable* dimensions (measurements along these dimensions have different units), so we need a scale-space for space and a scale-space for time.

Time measurements can essentially be processed in two ways: as pre-recorded frames or *instances*, or real-time. Temporal measurements stored for later replay or analysis, on whatever medium, fall in the first category. Humans perform continuously a temporal analysis with their senses, they measure *real-time* and are part of the second category. The scale-space treatment of these two categories will turn out to be essentially different.

Prerecorded sequences can be analyzed in a manner completely analogous with the spatial treatment of scaled operators, we just interchange space with time. The notion of *temporal scale* σ_τ then naturally emerges, which is the *temporal resolution*, a device property when we look at the recorded data (it is the inner scale of the data), and a free parameter *temporal scale* when we do the multi-scale analysis.

In the real-time measurement and analysis of temporal data we have a serious problem: the time axis is only a half axis: the past. There is a sharp and unavoidable boundary on the time axis: the present moment. This means that we can no longer apply our standard Gaussian kernels, because they have an (in theory) infinite extent in *both* directions. There is no way to include the future in our kernel, it would be a strong violation of causality.

But there may be a way out when we derive from first principles a new kernel that fulfils the constraint of causality: a kernel defined on a logarithmically remapped time axis. From this new causal kernel we might again derive the temporal and spatio-temporal family of scaled derivative operators. Jan Koenderink [Koenderink1988a] has presented the reasoning to derive the theory, and we will discuss it in detail below.

We will now treat both the pre-recorded and real-time situation in more detail, and give the operational details of the scale-space operators. The best source for reference is Jan Koenderink's original contribution on scale-time [Koenderink1988a].

There have appeared some fine papers discussing the real-time causal scale-space in detail by Luc Florack [Florack1997a, chapter 4.3] and Lindeberg, Fagerström and Bretzner [Lindeberg1996b, Lindeberg1997a, Bretzner1996a, Bretzner1997a]. Lindeberg also discusses the automatic selection of temporal scale [Lindeberg1997b].

20.2 Analysis of prerecorded time-sequences

Prerecorded temporal sample-sequences can be treated just as spatial sample-sequences. The causality constraint is satisfied because we include the whole available axis in our analysis. Boundary effects at both sides of the finite series in a practical situation are dealt with in a proper manner: we *choose* a way to extend the data (recall the discussion in chapter 5). This choice is essentially arbitrary, and the results can be predicted from the choice taken. The most common choice, also the choice taken throughout his book, is the *cyclic* representation: an infinite repetition of the data in all dimensions. Other feasible choices are mirroring, extending with zero's, extending with the mean etc.

The Gaussian derivative kernels with respect to time form the complete family of the *temporal differential operators*. They take the derivatives with respect to time, just as we have seen for the spatial derivative kernels. The temporal scale σ_τ is a free parameter, and the set of results of the operator for a range of scales is called a *temporal scale-space*.

When we combine the class of spatial differential operators with the class of temporal differential operators, we get the complete family of *spatio-temporal operators*. They take simultaneously derivatives to space and time, and can be constructed to any order through the construct of the familiar convolution with the Gaussian kernel.

For each dimension we can define a scale: the temporal scale σ_τ, the spatial scales $\{\sigma_x, \sigma_y, \sigma_z\}$. Typically, the spatial scales are identical (isotropy).

The Gaussian kernels can extend both into the past and the future in this case, because we *know both*.

The full signal is available, the complete recording *duration* is the available time axis, in analogy with the spatial domain of e.g. images. So the kernels extent over both the positive (the 'future') and the negative time axis (the 'past'). There is no real notion of future or past here, such as in the real-time case. The future resp. past here merely refers to data measured later resp. earlier than the datapoint (moment) we are currently analyzing, but which we have already in memory. Here are the graphs of some temporal derivative kernels of low order:

```
Block[{$DisplayFunction = Identity},
  pl = Table[Plot[Evaluate[D[gauss[t, σ = 1], {t, n}]],
     {t, -4, 4}, AxesLabel -> {"time", ""},
     PlotLabel -> "order: " <> ToString[n]], {n, 0, 3}]];
Show[GraphicsArray[pl], ImageSize -> 500];
```

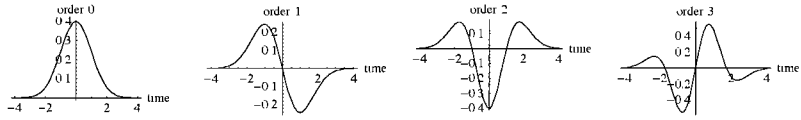


Figure 20.2 Proper temporal scale-space kernels for the temporal domain for prerecorded sequences. From left to right: Gaussian temporal derivatives for order 0 to 3. The 0th order kernel is the appropriate sampling kernel without differentiation, the temporal 'point-operator'.

Mixed partial spatio-temporal operators are spatial Gaussian derivative kernels concatenated with temporal Gaussian derivative kernels. This concatenation is a multiplication due to the separability of the dimensions involved.

```
spike = Table[0, {128}, {128}]; spike[[64, 64]] = 10^8;
gx := gD[spike, 1, 0, 20]; lapl := gD[spike, 2, 0, 20] + gD[spike, 0, 2, 20];
gxt = Table[Evaluate[gx * D[gauss[t, 1.], t]], {t, -4, 4}]; maxgxt = Max[gxt];
laplt = Table[Evaluate[lapl * D[gauss[t, 1.], t]], {t, -4, 4}];
maxlaplt = Max[laplt];
Block[{$DisplayFunction = Identity},
  p1 = ListDensityPlot[#, PlotRange -> {-maxgxt, maxgxt}] & /@ gxt;
  p2 = ListDensityPlot[#, PlotRange -> {-maxlaplt, maxlaplt}] & /@ laplt];


Show[GraphicsArray[{p1, p2}], ImageSize -> 400];
Plot[Evaluate[D[gauss[t, 1.], t]], {t, -4, 4},
  AxesLabel -> {"time →", ""}, AspectRatio -> .1, ImageSize -> 430];
```

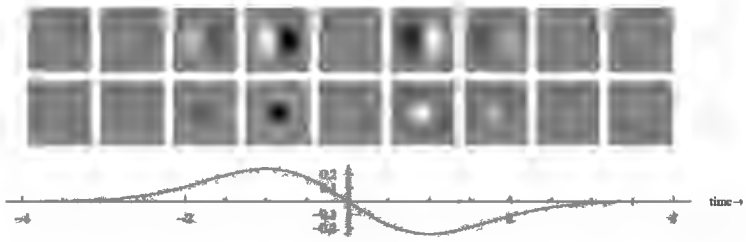


Figure 20.3 Spatio-temporal Gaussian derivative kernels. Top row: $\frac{\partial^2 G}{\partial t \partial x}$, the first order derivative in x and t, second row: $\frac{\partial}{\partial t}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)G$, the first order time derivative of the spatial Laplacian operator. Bottom plot: First order Gaussian temporal derivative operator, showing the temporal modulation of the spatial kernels. In the second half of the time domain the spatial kernels are reversed in polarity. The horizontal axis is the time axis.

So the appearance of a spatiotemporal operator is as a spatial operator *changing over time*, with a speed indicated ('tuned') by the temporal scale parameter. We illustrate this with an example in 2D-t. The mathematical expression for the mixed partial derivative operator, first order in the x-direction and time, is $\frac{\partial}{\partial t}\frac{\partial}{\partial x}$. This translates in scale-space theory into a convolution with a concatenation of the Gaussian derivative kernels $\frac{\partial G}{\partial t}$ and $\frac{\partial G}{\partial x}$, leading to a convolution operator $\frac{\partial^2 G}{\partial t \partial x}$ due to the separability of the kernels.

The following commands generate the sequence as an animation (electronic version only). Doubleclick the image to start the animation. Controls appear in the bottom of the notebook window.

```
spike = Table[0, {64}, {64}]; spike[[32, 32]] = 10^3;
gx := gD[spike, 1, 0, 10];
gxt = Table[Evaluate[gx * D[gauss[t, 1.], t]], {t, -4, 4, .5}];
maxgxt = Max[gxt]; p1 =
 ListDensityPlot[#, PlotRange -> {-maxgxt, maxgxt}, ImageSize -> 100] & /@
  gxt;
```

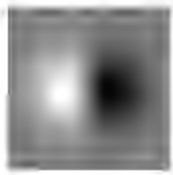


Figure 20.4 Animated sequence of the spatio-temporal Gaussian derivative kernel $\frac{\partial^2 G}{\partial t \partial x}$. In the second half of the time domain the spatial kernels are reversed in polarity. Similar sequence as the top row in figure 20.3.

Note that these temporal derivatives are *point operators*, they merely measure the change of the parameter (e.g. luminance) over time at the operational point, so per pixel. This is essentially different from the detection of motion, the subject of chapter 17, where relations between neighboring pixels are established to measure the *optic flow*.

20.3 Causal time-scale is logarithmic

For real-time systems the situation is completely different. We noted in the introduction that we can only deal with the past, i.e. we only have the half time-axis. This is incompatible with the infinite extent of the Gaussian kernel to both sides.

With Koenderink's words: "Because the diffusion spreads influences with infinite speed any blurring will immediately spread into the remote future thereby violating the principle of temporal causality. It is clear that the scale-space method can only lead to acceptable results over the complete axis, but never over a mere semi-axis. On the other hand the diffusion equation is the unique solution that respects causality in the resolution domain. Thus there can be no hope of finding an alternative. The dilemma is complete" ([Koenderink1988a]).

The solution, proposed by Koenderink, is to *remap* (reparametrize) the half t-axis into a full axis. The question is then how this should be done. We follow here Koenderink's original reasoning to come to the mapping function, and to derive the Gaussian derivative kernels on the new time axis.

We call the remapping $s(t)$. We define t_0 the present moment, which can never be reached, for as soon as we try to measure it, it is already further in time. It is our reference point, our only point in time absolutely defined, our *fiducial moment*. Every real-time measurement is relative to this point in time. Then s should be a function of $\mu = t_0 - t$, so $s(\mu) = s(t_0 - t)$. We choose the parameter μ to be dimensionless, and $\mu = 0$ for the present moment, and $\mu = -\infty$ for the infinite past. So we get $s(\mu) = s(\frac{t_0 - t}{\tau})$. The parameter τ is some time constant and is essentially arbitrary. It is the *scale* of our measurement, and we should be able to give it any value, so we want the diffusion to be scale-invariant on the μ-domain.

We also want shift invariance on this time axis, and the application of different clocks, so we require that a transformation $t' = a\,t + b$ leaves $s(t)$ invariant. μ is invariant if we change clocks.

```
Plot[0, {t, -10, 0},
  Ticks -> {{{-6, "μ2"}, {-2, "μ1"}}, None}, PlotRange -> {-1, 5},
  AspectRatio -> .1, Epilog -> {Text["← time", {-10, .5}],
    Text["0 (= present)", {.6, .5}]}, ImageSize -> 440];
```

Figure 20.5 The time-axis has only the negative half. The right end is the present moment. The moments in the past μ_1 and μ_2 are observed with a resolution that is proportional with their 'past time', i.e. μ.

On our new time-axis $s(t)$ the diffusion should be a normal, causal diffusion. On every point of the s-axis we have the same amount of diffusion, i.e. the diffusion is *homogeneous* on the s-domain.
The 'inner scale' or resolution of our measurement has to become smaller and smaller when we want to approach the present moment. But even if we use femtosecond measuring

devices, we will never catch the present moment. On the other side of the s-axis, a long time ago, we don't want that high resolution. An event some centuries ago is placed with a resolution of say a year, and the moment that the dinosaurs disappeared from earth, say some 65 million years ago, is referred to with an accuracy of a million years or so.

This intuitive reasoning is an expression of the requirement that we want our time-resolution τ on the s-axis to be proportional to μ, i.e. $\tau \simeq \mu$ or $\frac{\tau}{\mu}$ = constant. So for small μ we have a small resolution, for large μ a large one.

```
t0 = 1; τ = 1;
inset = LogLinearPlot[0, {x, 0, 1}, GridLines -> Automatic, Frame -> True,
  FrameTicks -> False, DisplayFunction -> Identity, AspectRatio -> .25];
Plot[-Log[(t0 - t)/τ], {t, -1, t0}, PlotRange -> {-1.5, 4},
 AxesLabel -> {"t", "s"},
 Epilog -> {Rectangle[{-1, -Log[t0 + 1]}, {.75, -Log[t0 - .75]}, inset],
   Dashing[{.01, .01}], Line[{{1, -1.5}, {1, 4}}]}, ImageSize -> 300];
```

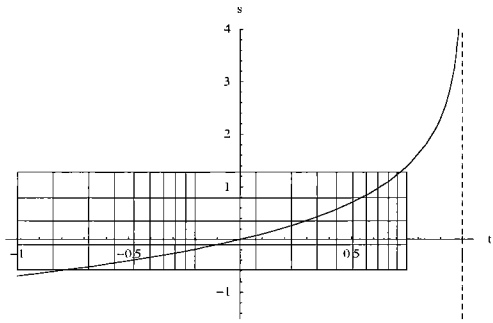


Figure 20.6 The logarithmic mapping of the horizontal t-time half-axis onto the vertical s-time full axis. The present moment t_0 (at $t = 1$ in this example, indicated by the vertical dashed line) can never be reached. The s-axis is now a full axis, and fully available for diffusion. The rectangular inset box with gridlines shows a typical working area for a real-time system. The response time delimits the area at the right, the lifetime (history) at the left. Figure adapted from [Florack1997a].

On the s-axis we should have the possibility for normal, causal diffusion. This means that the 'magnification' $\left|\frac{ds}{d\mu}\right|$ should be proportional to $\frac{\tau}{\mu}$. Then the s-axis is 'stretched' for every μ in such a way that the scale (or 'diffusion length' as Koenderink calls it) in the s-domain is a *constant relative* diffusion length in the μ-domain.

Uniform sampling in the s-domain gives a *graded resolution history* in the t- or μ-domain. In formula : $\left|\frac{ds}{d\mu}\right| \simeq \frac{\tau}{\mu}$ or $\left|\frac{ds}{d\mu}\right| = \frac{\alpha}{\mu}$. From this partial differential equation we derive that the mapping $s(\mu)$ must be logarithmic:

```
DSolve[∂μ s[μ] == α/μ, s[μ], μ]
```

```
{{s[μ] → C[1] + α Log[μ]}}
```

So our mapping for s is now: $s = \alpha \ln(\frac{t_0 - t}{\tau}) + \text{constant}$. The constant is an arbitrary translation, for which we defined to be invariant, so we choose this constant to be zero. We choose the arbitrary scaling parameter α to be unity, so we get

$$s = \ln(\frac{t_0 - t}{\tau}).$$

This is a fundamental result. For a causal interpretation of the time axis we need to sample time in a logarithmic fashion. It means that the present moment is mapped to infinity, which conforms to our notion that we can never reach it. We can now freely diffuse on the s-axis, as we have a well defined scale at all moments on our transformed time axis. This mapping also conforms to our notion of resolution of events in the past: our memory seems to do the same weighting of the aperture over the past as the transformation we introduced above.

In the s-domain we can now run the diffusion equation without violation of temporal causality. The diffusion equation is now $\frac{\partial^2 L}{\partial s^2} = \frac{\partial L}{\partial v_s}$, where $v_s = \frac{1}{2}\tau^2$ is the temporal variance. We recall from the diffusion equation in the spatial domain that the Laplacian (of the luminance in our case) along the diffusion axis is equal to the rate of change (of the luminance) with the *variance* of the scale. The check of dimensions on both sides of the equation is always a good help. The temporal blurring kernels (the scale-space measurement apertures) are now given by

$$K(s, s'; \sigma_s) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{(s-s')^2}{2\tau^2}}, \text{ or } K(s, s'; v_s) = \frac{1}{\sqrt{4\pi v_s}} e^{-\frac{(s-s')^2}{4v_s}}.$$

20.4 Other derivations of logarithmic scale-time

Florack [Florack1997a] came to the same result from a different perspective, from abstract mathematics. He used a method from *group theory*. Essentially, a group is a mathematical set of operations, which (in popular words) all do the same operation, but with a different parameter. Examples of a transformation group are the group of rotations, the group of additions, the group of translations etc. A group is formally defined as a set of similar transformations, with a member that does the unity operation (projects on itself, i.e. does nothing, e.g. rotation over zero degrees, an enlargement of 1, a translation of zero etc.), it must have an inverse (e.g. rotation clockwise, but also anti-clockwise) and one must be able to concatenate its members (e.g. a total rotation which consists of two separate rotations after each other).

Florack studied the group properties of whole and half axes of real numbers. The group of summations is a group on the whole axis, which includes the positive and the negative numbers. This group however is *not* a group on the half axis. For we might be able to do a summation which has a result outside the allowed domain. The group of multiplications however is a group on the positive half axis.

Two numbers multiplied from the half axis give a result on the same half axis. If we could make all sums into multiplications, we would have an operation that makes it a group again. The formal transformation from sums into multiplications is the logarithmic function: $e^{a+b} = e^a * e^b$ and its inverse $\ln(a * b) = \ln(a) + \ln(b)$. The zero element is addition of zero,

or multiplication with one. So the result is the same logarithmic function as the function of choice for the causal parametrization of the half axis.

Lindeberg and Fagerström [Lindeberg1996b] derived the causal temporal differential operator from the non-creation of local extrema (zero-crossings) with increasing scale. Jaynes' method of 'transformation groups' to construct prior probabilities has a strong similarity to the reasoning in this chapter. For details see [Jaynes1968].

Interestingly, we encounter more often a logarithmic parametrization of a half axis when the physics of observations is involved:

- Light intensities are only defined for positive values, and form a half axis. It is well known e.g that the eye performs a logarithmic transformation on the intensity measured on the retina.
- Sound intensities are measured in decibels (dB), i.e. on a logarithmic scale.
- Scale is only defined for positive values, and form a half axis (scale-space). The natural scalestep τ on the scale-axis in scale-space is the logarithm of the diffusion scale σ: $\tau = \ln(\sigma) - \ln(\sigma_0)$ (recall chapter 1).

Another example of the causal logarithmic mapping of the time axis is the striking Law of Benford, which says that in a physical *measurement* that involves a duration, the occurrence of the first digit has a logarithmic distribution. I.e. a "1" as the first digit occurs roughly 6.5 times more often than a "9"! This law follows immediately from the assumption that random intervals are *uniformly* distributed on the logarithmic s-axis [Florack1997a, pp. 112].

▲ Task 20.1 The decay times of a large set of radioactive isotopes is available on internet: isotopes.lbl.gov and ie.lbl.gov/tori.html. Show that the first digits of these decay times indeed show Benford's Law, with a much pronounced occurrence of "1"s. [Buck1993].

Our perception of time also seems to be non-linear. We have more trouble remembering accurate events that took place a long time ago, as in our childhood, than events that just recently happened. Why has last year passed by so quickly, and did a year at highschool seem to last so much longer? Our perception is that time seems to go quicker when we get older. We have a increasingly longer reference to the past with which we can compare, and our notion is that time seemed to go slower a longer time ago. These observations suggest some logarithmic scaling: stretched out at the low end and compressed at the high end. It is known in psychological literature that our perceived time 'units' may be related to our age: a 10% of age for a 5-year old is half a year, for a 50-year old it is 5 years. A half year seems to pass just as quick for the 5-year old as 5 years for the 50-year old. This also leads to a logarithmic notion of time. See also [Gibbon18981] and [Pöppel1978a].

20.5 Real-time receptive fields

We have now all information to study the shape of the causal temporal derivative operators. The kernel in the transformed s-domain was given above. The kernel in the original temporal domain t becomes

$$K(t, t_0; \tau) = \frac{1}{\sqrt{2\pi}\,\tau} e^{-\frac{1}{2\tau^2}\ln(\frac{t_0-t}{\tau})^2}.$$

In figure 20.7 we see that the Gaussian kernel and its temporal derivatives are *skewed*, due to the logarithmic time axis remapping. It is clear that the present moment t_0 can never be reached.

```
Clear[gt]; Block[{$DisplayFunction = Identity},
 pl = Table[τ = .2; t0 = 0; Plot[Evaluate[gt[n] = D[
        1/Sqrt[2 π τ] Exp[-1/(2 τ^2) Log[(t0 - t)/τ]^2], {t, n}]],
    {t, -.4, 0}, PlotRange -> All,
    PlotLabel → "temporal order = " <> ToString[n],
    Epilog -> Text["time →", {-.45, 0}]], {n, 0, 2}]];
Show[GraphicsArray[pl], ImageSize -> 400];
```

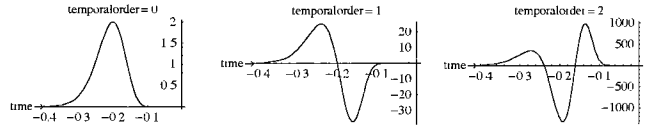


Figure 20.7 Left, middle, right: the zeroth, first and second Gaussian temporal derivative operator in causal time. The timescale in each plot runs from the past on the left, to the present on the right. The temporal scale $\tau = 200$ ms, the right boundary is the present, $t_0 = 0$. Note the pronounced skewness of the kernels.

The zerocrossing of the first order derivative (and thus the peak of the zeroth order kernel) is just at $t = -\tau$. The extrema of the first derivative kernel are found when we set the second derivative of the time kernel to zero. Here are both solutions:

```
Clear[t, τ, timekernel]; t0 = 0; Off[InverseFunction::ifun, Solve::ifun];
timekernel[t_] = 1/Sqrt[2 π τ^2] Exp[-1/(2 τ^2) Log[(t0 - t)/τ]^2];
zerot = Solve[∂t timekernel[t] == 0, t]
zerott = Solve[∂t,t timekernel[t] == 0, t]
```

$$\{\{t \to -\tau\}, \{t \to -\tau\}\}$$

$$\left\{\left\{t \to -e^{\frac{1}{2}\left(-\tau^2 - \tau\sqrt{4+\tau^2}\right)}\,\tau\right\}, \left\{t \to -e^{\frac{1}{2}\left(-\tau^2 + \tau\sqrt{4+\tau^2}\right)}\,\tau\right\}\right\}$$

It is now easy to combine the spatial and temporal kernels. Time and space are separable incommensurable dimensions, so we may apply the operators in any order: first a spatial kernel and then a temporal kernel is the same as first a temporal and then a spatial kernel.

The most logical choice is the simultaneous action. When we make a physical realization of such an operator, we have to plot it in the spatial and time domain. An example is given below for a 2D first order spatial derivative, and first order temporal derivative. We get a 2D-time sequence:

```
spike = Table[0, {128}, {128}]; spike[[64, 64]] = 10^8;
τ = .3; gt = ∂t timekernel[-t]; gx = gD[spike, 1, 0, 20];
max = Max[gx ∂t timekernel[t] /. First[zerott]];
pl = Table[ListDensityPlot[gt gx, PlotRange -> {-max, max},
    DisplayFunction -> Identity], {t, .1, .6, .05}];
Show[GraphicsArray[pl], ImageSize -> 500];
```

Figure 20.8 Eleven frames of a time series of a 2D-time spatio-temporal time-causal Gaussian derivative kernel, which takes the first order derivative to time and the first order to space in the x-dimension. The present moment is on the left. Note the inversion that takes place in the spatial pattern during the time course.

This has been measured in cortical simple cells by single cell recordings (see figure 20.9). See for the methodology to map receptive field sensitivity profiles chap. 9 and figure 11.12.

```
Show[GraphicsArray[Partition[
    Import /@ ("separablext " <> ToString[#] <> ".gif" & /@ Range[60]),
    10]], ImageSize -> 500];
```

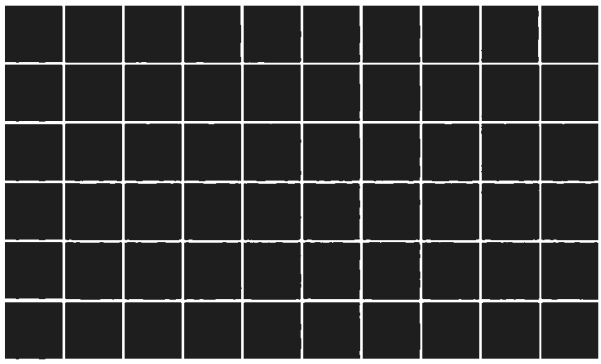


Figure 20.9 Time sequence of a cortical simple cell. Frames row-wise, upper left frame is first. Time domain: 0 - 300 msec in 5 msec steps. Data of the cell: on-center X cell, non-lagged; X-Y domain size: 3 x 3 degs; Bar size: 0.5 x 0.5 degs; Eye: left; Total time for receptive field measurement: 19 min. Number of repetitions: 50; Duration of stimuli: 26.3ms. From: neurovision.berkeley.edu/Demonstrations/VSOC/teaching/RF/LGN.html.

20.6 A scale-space model for time-causal receptive fields

The temporal behaviour of cortical receptive fields has been measured extensively. The method most often used is the method of 'reverse correlation' (explained in section 9.6).

Figure 20.9 shows a recording of a cortical simple cell from Freemans' lab in Berkeley, that shows clearly the spatial similarity to the first Gaussian derivative kernel, and the modulated amplitude over time. The spatio-temporal relations become more clear when they are plotted in the spatio-temporal domain. Figure 20.10 shows a plot where the horizontal axis displays the spatial modulation, the vertical axis displays the temporal modulation.

```
Show[Import["Simple cell V1 XT.gif"],
  Frame -> True, FrameLabel -> {"space →", "time →"},
  FrameTicks -> None, ImageSize -> 100];
```

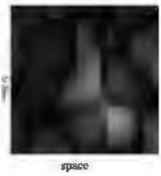


Figure 20.10 Spatio-temporal response of a cortical simple cell. From [DeAngelis1995a]. The polarity of the spatial response reverses sign during the time course of the response of the cell. Horizontal axis is space (6 degs), vertical axis is time (0-300 msec). Time zero is at the bottom. Cell data: X-Y domain size: 6 x 6 degs; X-Y grid: 20 x 20 points; time domain: 0 - 300 msec in 5 msec steps; orientation: 15 degs; bar size: 2.5 x 0.5 degs; duration of individual stimuli: 52.8 msec (4 frames).From [neurovision.berkeley.edu]. Copyright Izumi Ohzawa, UC Berkeley.

Here is another cell measured by Dario Ringach and coworkers [Ringach1997]. Data from manuelita.psych.ucla.edu/~dario/research.htm:

```
Show[GraphicsArray[Partition[
    Import /@ ("Ringach V1 " <> ToString[#] <> ".gif" & /@ Range[12]),
    6]], ImageSize -> 500];
```

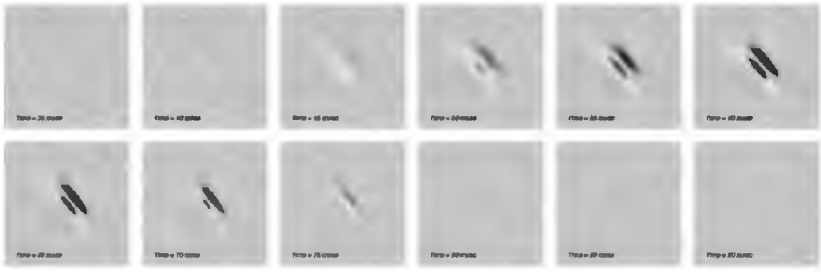


Figure 20.11 Timescale 35 ms (first frame) to 90 ms (last frame) in 5 ms intervals. Upper left frame is the start of the sequence.

Recent more precise measurements of the spatio-temporal properties of macaque monkey and cat LGN and cortical receptive fields give support for the scale-time theory for causal time sampling. De Valois, Cottaris, Mahon, Elfar and Wilson [DeValois2000] applied the method of reverse correlation and multiple receptive field mapping stimuli (m - sequence,

maximum length white noise stimuli) to map numerous receptive fields with high spatial and temporal resolution. Fig 20.12 shows some resulting receptive field maps:

```
Show[GraphicsArray[Import /@
    ("Valois RF " <> ToString[#] <> ".jpg" & /@ {a, b, c, d, e, f})],
  ImageSize -> 500];
```

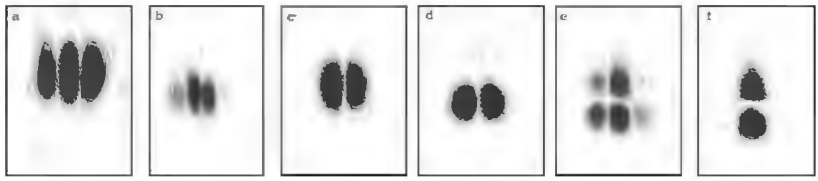


Figure 20.12 Examples of spatio-temporal receptive field maps of a sample of V1 simple cells of macaque monkey. Vertical axis in each plot: time axis from 0 ms (bottom) to 200 ms (top). Horizontal axis per plot: space (in degrees), a: 0-0.9, b: 0-1.2, c,d: 0-0.6, e: 0,1.9, f: 0-1.6 degrees. Note the clearly skewed sensitivity profiles in the time direction. Every 'island' has opposite polarity to its neighboring 'island' in each plot. Due to black and white reproduction the sign of the response could not be reproduced. The scale-space models for the plots are respectively: a: $\frac{\partial^2 L}{\partial x^2}$, b: $\frac{\partial^3 L}{\partial x^3}$, c: $\frac{\partial L}{\partial x}$, d: $\frac{\partial L}{\partial x}$, e: $\frac{\partial^3 L}{\partial x^2 \partial t}$, f: $\frac{\partial^2 L}{\partial x \partial t}$. Adapted from [DeValois2000].

If we plot the predicted sensitivity profiles according to Gaussian scale-space theory we get remarkably similar results. In figure 20.13 the space-time plots are shown for zeroth to second spatial and temporal differential order. Note the skewness in the temporal direction.

Important support for especially the Gaussian scale-time derivative model comes from another observation by De Valois et al. [DeValois2000]. They state: 'Note that the response time course of these two non-directional cell populations are approximately 90 degrees shifted in phase relative to each other, that is, they are on average close to temporal quadrature.

That is, the population of biphasic cells is shifting over from one phase to the reverse at the time that the monophasic cell population reaches its peak response' (a quadrature filter can be defined, independently of the dimensionality of the signal space, as a filter that is zero over one half of the Fourier space. In the spatial domain, the filter is complex: an even real part and an odd imaginary part).

```
Clear[gt, gs, n]; τ = 0.3; t0 = 0; σ = 2;
gt[n_] = D[1/Sqrt[2 π τ^2] Exp[-1/(2 τ^2) Log[-(t0 - t)/τ]^2], {t, n}];

gs[n_] = D[1/Sqrt[2 π σ^2] Exp[-1/(2 σ^2) x^2], {x, n}];
Block[{$DisplayFunction = Identity},
  p = Table[ContourPlot[Evaluate[gt[i] gs[j]],
     {x, -15, 15}, {t, .01, .8}, PlotPoints -> 30,
     ContourShading -> True, FrameLabel → {"space", "time"},
     PlotLabel → "n_space =" <> ToString[j] <> ",n_time =" <> ToString[i] ],
    {j, 0, 2}, {i, 0, 2}]];
```

```
Show[GraphicsArray[p], ImageSize → 390];
```

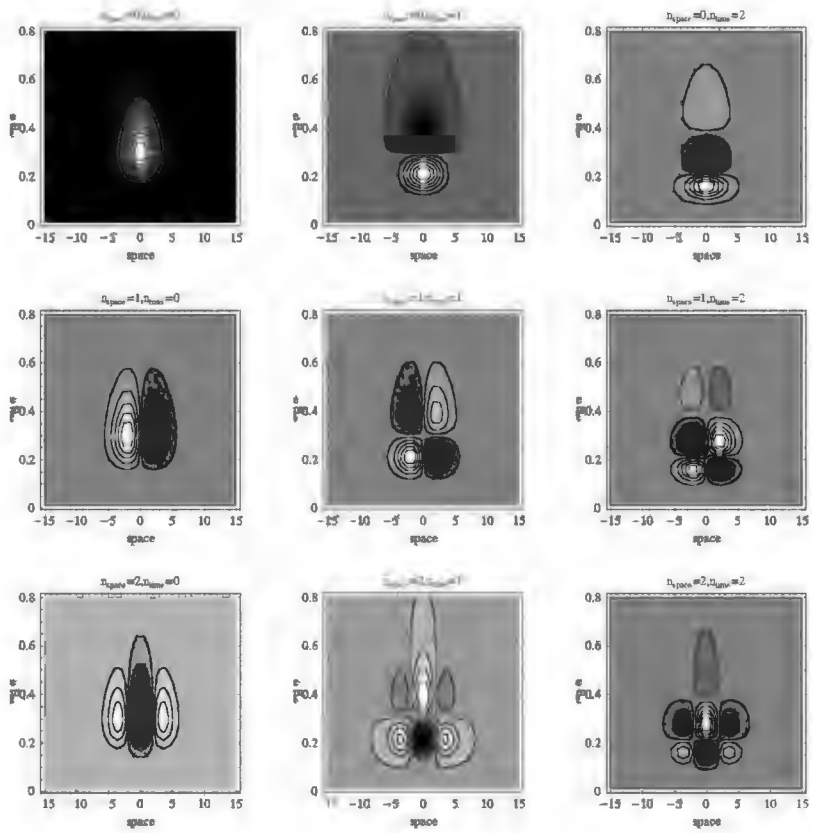


Figure 20.13 Model for time-causal spatio-temporal receptive field sensitivity profiles from Gaussian scale-space theory. All combinations from zeroth to second order partial derivative operators with respect to space and time are shown. Vertical axis: time. Horizontal axis: space.

This fits very well to the model: The receptive fields are the zeroth and first order temporal Gaussian derivative, and for these functions the zerocrossing of the the first order derivative coincides with the maximum of the zeroth order.

```
Show[GraphicsArray[Import /@ {"Valois time to peak01.jpg",
    "Valois time to peak02.jpg"}], ImageSize -> 360];
```

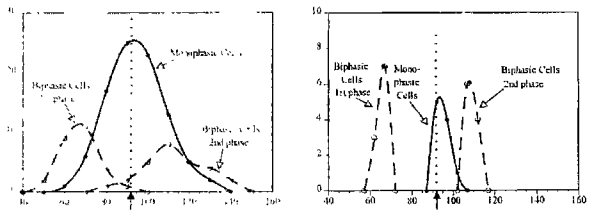


Figure 20.14 Distribution of time-to-peak responses, calculated from the spatio-temporal receptive field mappings of the population of monophasic cells, and of both the initial and later reversed phase of the biphasic cells. Note that the time of peak response for the monophasic cells almost exactly coincides with the time of polarity reversal of the biphasic cells, in perfect agreement with the Gaussian temporal derivative model. Horizontal axis: time-to-peak response time, in ms. Vertical axis: number of cells. Right figure: Time-to-peak response of a random sample of just 5 cells, showing that even a small sample set has the properties of the larger collection. Adapted from [DeValois2000].

De Valois et al. define the *biphasic index* to be the ratio of the amplitude of the 2nd temporal peak and the amplitude of the first peak of the temporal response function. The maximum and minimum value of the first order temporal derivative of the kernel (∂_t `temprf`) are where the second order temporal derivative ($\partial_{t,t}$ `temprf`) is zero. We simplify the expression knowing that $\sigma_t > 0$, and get two values, one for the minimum amplitude and one for the maximum amplitude:

```
Clear[τ, t0]; temprf = 1/(Sqrt[2 π] τ^2) Exp[-1/(2 τ^2) Log[-t/t0]^2];
peaks = Simplify[∂_t temprf /. Solve[∂_{t,t} temprf == 0, t], τ > 0]
```

$$\left\{-\frac{e^{\frac{1}{4}\left(-2+\tau^2+\tau\sqrt{4+\tau^2}\right)}\left(\tau+\sqrt{4+\tau^2}\right)}{2\sqrt{2\pi}\,t0\,\tau^3},\ \frac{e^{\frac{1}{4}\left(-2+\tau^2-\tau\sqrt{4+\tau^2}\right)}\left(-\tau+\sqrt{4+\tau^2}\right)}{2\sqrt{2\pi}\,t0\,\tau^3}\right\}$$

The biphasic index is no longer a function to t_0, and only depends on σ_t:

```
biphasicindex = -peaks[[2]]/peaks[[1]] // Simplify
```

$$\frac{e^{-\frac{1}{2}\tau\sqrt{4+\tau^2}}\left(-\tau+\sqrt{4+\tau^2}\right)}{\tau+\sqrt{4+\tau^2}}$$

So we may expect a range of values for the biphasic index. De Valois et al. found values for the biphasic index having a distinct peak between 0.4 and 0.8, which corresponds well with a temporal scale range of 100 - 500 milliseconds (see graph in figure 20.15).

```
Plot[biphasicindex, {τ, .1, .5}, ImageSize -> 140,
  AxesLabel → {"timescale\nτ", "Biphasic\nindex"}];
```

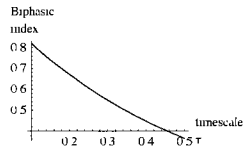


Figure 20.15 Predicted biphasic index according to the Gaussian time-causal temporal differential operator model. The biphasic index is defined [DeValois2000] as the ratio between the amplitude magnitudes of the first and second peak in the response in the time domain.

▲ Task 20.2 Give a measure for the skewness of the time-causal Gaussian kernel.

20.7 Conclusion

The causal time-scale, multi-scale temporal differential operator model from Gaussian scale-space theory has not yet been tested against the wealth of currently available receptive field measurement data.

It may be an interesting experiment, to test the quantitative similarity, and to find the statistics of the applied spatial and temporal scales, as well as the distribution of the differential order.

The Gaussian scale-space model is especially attractive because of its robust physical underpinning by the principal of temporal causality, leading to the natural notion of the logarithmic mapping of the time axis in a real-time measurement.

The distributions of the locations of the different scales and the differential orders have not yet been mapped on the detailed cortical orientation column with the pinwheel structure. Orientation has been clearly charted due to spectacular developments in optical dye high resolution recording techniques in awake animals. Many interesting questions come up: Is the scale of the operator mapped along the spokes of the pinwheel? Is the central singularity in the repetitive pinwheel structure the largest scale? Is differential order coded in depth in the columns?

These are all new questions arising from a new model. The answer to these questions can be expected within a reasonable time, given the fast developments, both in high resolution recording techniques, and the increase in resolution of non-invasive mapping techniques as high-field functional magnetic resonance imaging (fMRI) [Logothetis1999].

20.8 Summary of this chapter

When a time sequence of data is available in stored form, we can apply the regular symmetric Gaussian derivative kernels as causal multi-scale differential operators for temporal analysis, in complete analogy with the spatial case. When the measurement and analysis is real-time, we need a reparametrization of the time axis in a logarithmic fashion. The resulting kernels are skewed towards the past. The present can be never reached, the new logarithmic axis guarantees full causality. The derivation is performed by the first principle of a scale of observation on the new time axis which is proportional to the time the event happened. This seems to fit well in the intuitive perception of time by humans.

Recent physiological measurements of LGN cell receptive fields and cortical V1 simple cell receptive fields reveal that the biological system seems to employ the temporal and spatiotemporal differential operators. Especially striking is the skewness in the temporal domain, giving strong support for the working of the biological cells as time-causal temporal differential operators.

21. Geometry-driven diffusion

To teach is to learn twice. (Joseph Joubert, 1754-1824)

21.1 Adaptive Smoothing and Image Evolution

So far we calculated edges and other differential invariants at a range of scales. The task determined whether to select a fine or a coarse scale. The advantage of selecting a larger scale was the improved reduction of noise, and the appearance of more prominent structure, but the price to pay for this is reduced localization accuracy. Linear, isotropic diffusion cannot preserve the position of the differential invariant features over scale.
A solution is to make the diffusion, i.e. the amount of blurring, locally *adaptive* to the structure of the image. E.g. in order to preserve edges while reducing the noise by area-averaging such as blurring, one may try to prevent blurring at the location of the edges, but do a good noise-reducing job at pixels (voxels) in a homogeneous area, i.e. where there are no edges. E.g. the well-known Sobel and Prewitt edge operators do an averaging in the direction perpendicular to its differential operation (as opposed to the Roberts operator).
This adaptive filtering process is possible by three classes of (all nonlinear) mathematical approaches, which are in essence equivalent:

1. Nonlinear partial differential equations (PDE's), i.e. nonlinear diffusion equations which evolve the luminance function as some function of a *flow*. This general approach is known as the 'nonlinear PDE approach';
2. Curve evolution of the isophotes (curves in 2D, surfaces in 3D) in the image. This is known as the 'curve evolution approach'.
3. Variational methods that minimize some energy functional on the image. This is known as the 'energy minimization approach' or 'variational approach'.

The word 'nonlinear' implies the inclusion of a nonlinearity in the algorithm.
This can be done in an infinite variety, and it takes *geometric reasoning* to come up with the right nonlinearity for the task.

This explains the commonly used term 'geometry-driven diffusion' for this field. Otherwise stated, this gives us a tool to include *knowledge* in the way we want to modify the images by some evolutionary process. We can include knowledge about a preferred direction of diffusion, or that we like the diffusion to be reduced at edges or at points of high curvature in order to preserve edges and corners, etc. As we study the evolution of images over time, we also call this field *evolutionary computing* of image structure, or the application of *evolutionary operations*.

The applicability of these mathematically challenging and nonrigid (but 'only requiring a lot of clever well-understood reasoning') approaches to image processing sparked the attention of both pure and applied mathematicians. In each of the general approaches mentioned below

a wealth of methods have been proposed. From 1993-1996 a consortium of laboratories in the US and Europe (sponsored by the US National Science Foundation and the European Community) was active in this field. The result of their work is a rather complete recording of this field at the time [ter Haar Romeny 1994f].

This chapter is an introduction to this rapidly expanding field. Excellent reviews have appeared for each of the above general approaches. A good point for further introduction into nonlinear diffusion equations are the review papers by Weickert [Weickert 1997d, 1998a, 1999b]. Curve evolution was given much momentum by the important discovery by Osher and Sethian that the process of curve evolution became much more stable and flexible when the curve was considered the zero level set of some characteristic function. The nonlinear evolution is done on the characteristic function.

Many, if not most, aspects of this *level set* approach are treated in the book by Sethian [Sethian 1999]. The variational approaches were pioneered by Mumford and Shah [Mumford and Shah 1985a]. A detailed overview of these methods is given in the book by Morel and Solemini [Morel and Solemini 1995].
In the early visual pathway we see an abundance of feedback. A striking finding is that the majority of fibers (roughly 75% !) in the *optic radiation* (the fibers between the LGN and the primary visual cortex) are projecting in a *retrograde* (backwards) fashion, from cortex to LGN [Sherman and Kock 1990, Sherman 1996a], recall chapter 11.

These cortico-thalamic projections may well tune the receptive fields with the differential geometric information extracted with the receptive fields in the visual cortex.

It is long known that the receptive field sensitivity functions of LGN receptive fields are not static. They may be temporally modulated as temporal differentiating operators, they may also be some function of the inconing image under control of the cortical input. Unfortunately little is still known about the exact wiring. It is clear that any LGN cell receives input from the cortical cell it projects to [Mumford 1991a], but the cortico-fugal (Latin: fugare = to escape) connections seem to project all over the LGN [Logothetis 1999], making long-range interactions possible.

Nonlinear diffusion techniques have become an important and extensive branch in computer vision theory. It is impossible to explain all approaches, theories and techniques in this chapter. This chapter is a starting point for interactive study and a basic understanding of the mathematics involved, such as the underlying geometric reasoning leading to the particular PDE and numerical stability of approximations.

21.2 Nonlinear Diffusion Equations

The introduction of a *conductivity coefficient* (c) in the diffusion equation makes it possible to make the diffusion adaptive to local image structure:

$$\frac{\partial L}{\partial s} = \vec{\nabla}.c\,\vec{\nabla}L$$

where the function $c = c(L, \frac{\partial L}{\partial x}, \frac{\partial^2 L}{\partial x^2}, ...)$ is a function of local image differential structure, i.e. depends on local partial derivatives. The general structure of a diffusion PDE is shown in the formula above: The change of luminance with increasing scale is a *divergence* ($\vec{\nabla}.$) of some *flow* ($c\vec{\nabla}L$). We also call $c\vec{\nabla}L$ the *flux function*. With $c = 1$ we have normal linear, isotropic diffusion: the divergence of the gradient is the Laplacian. The most famous case, introduced by Perona and Malik in their seminal paper [Perona and Malik 1990] which sparked the field, is where c is a *decreasing* monotonic function of the gradient magnitude, $c = c(|\vec{\nabla}L|)$. The diffusion is reduced at the location of edges because c is small at strong edges, and vice versa.

The term conductivity is understood when we consider the analogon of diffusion of heat in a metal plate. The local flow of heat can be influenced by the insertion of local heat isolators in the plate, which act as barriers for the heatflow, leading to a non-isotropic diffusion of heat and thus to a non-isotropic temperature distribution over the plate over time. The conductivity c is different for every pixel location, and is a function of the local differential structure. We will discuss several possible classical functions proposed in the literature.

The nonlinear diffusion paradigm enables *geometric reasoning*, we may put knowledge in the task of the evolution of the image. Examples of such reasoning statements are:
- 'reduce the diffusion at locations where edges (or other local features such as corners, T-junctions, etc.) occur', or
- 'adapt the diffusion so it is maximized along edges and minimized across edges', or
- 'enhance the diffusion in the direction of ridges and reduce the diffusion perpendicular to them' etc.

The naming of nonlinear diffusion equation is sometimes not consistent in the literature. We list some names and their meaning:
- *linear diffusion*, equivalent to *isotropic diffusion*: the diffusion is the same in all directions, for all dimensions; the conductivity function c is a constant;
- *geometry-driven diffusion*, the most general naming for the use of geometric reasoning; it includes (invariant) local differential geometric properties in the diffusion equations, curve evolution schemes and variational, energy minimizing expressions;
- *variable conductance diffusion*, the 'Perona and Malik' type of gradient magnitude controlled diffusion;
- *inhomogeneous diffusion*: the diffusion is different for different locations in the image; this is the most general naming;
- *anisotropic diffusion*: the diffusion is different for different *directions*; the Perona and Malik nonlinear diffusion scheme is *not* an anisotropic diffusion (despite the title of their original paper); it says nothing about the direction, only about the magnitude of the diffusion; it is an example of inhomogeneous diffusion, or variable conductance diffusion;

- *coherence enhancing diffusion*: a particular case of anisotropic diffusion where the direction of the diffusion is governed by the direction of local image structure, for example the eigenvectors of the *structure matrix* or *structure tensor* (the outer product of the gradient with itself, explained in chapter 6), or the local ridgeness.
- *tensor driven diffusion*: the diffusion coefficient is a tensor, not a scalar. Mathematically,

this is the most general form. The tensor operates as a Euclidean operator on the gradient flow vector, and can modify its magnitude and direction accordingly. The tensor can be calculated on a different scale than the gradient.

21.3 The Perona & Malik Equation

Perona and Malik [Perona and Malik 1990] proposed to make c a function of the gradient magnitude in order to reduce the diffusion at the location of edges:

$$\frac{\partial L}{\partial s} = \vec{\nabla}.c\left(\left|\vec{\nabla}L\right|\right)\vec{\nabla}L \qquad (1)$$

with two possible choices for c: $c_1 = e^{-\frac{|\nabla L|^2}{k^2}}$, and $c_2 = 1\Big/\left(1+\frac{|\nabla L|^2}{k^2}\right)$. Note that these expressions are equal up to a first order approximation:

```
<< FrontEndVision`FEV`;
c1 = Series[Exp[-(∇L)^2/k^2], {(∇L), 0, 4}]
c2 = Series[1/(1 + (∇L)^2/k^2), {(∇L), 0, 4}]
```

$$1 - \frac{(\nabla L)^2}{k^2} + \frac{(\nabla L)^4}{2\,k^4} + O[\nabla L]^5$$

$$1 - \frac{(\nabla L)^2}{k^2} + \frac{(\nabla L)^4}{k^4} + O[\nabla L]^5$$

```
im = Import["mr256.gif"][[1, 1]]; σ = 2;
DisplayTogetherArray[ListDensityPlot[Exp[-(gD[im, 1, 0, σ]^2 + gD[im, 0, 1, σ]^2)/#^2],
    PlotLabel -> "k = " <> ToString[#]] & /@ {5, 10, 20}, ImageSize -> 400];
```

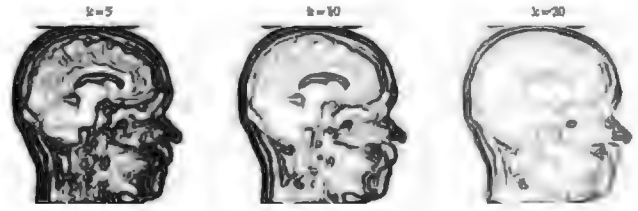


Figure 21.1 The conductivity coefficient c_1 in the Perona & Malik equation as a function of the parameter k. Gradient scale: σ = 2 pixels, image resolution 256^2. For higher k, larger gradients are taken into account only.

The geometric reasoning here is to let intra-region smoothing occur preferentially over inter-region smoothing. In this particular choice the conductivity function $c(\vec{\nabla}L)$ is small for large edge-strength and vice versa, i.e. the image is diffused most where the gradient is smallest.

Figure 21.1 shows $c_1 = e^{-\frac{|\nabla L|^2}{k^2}}$ for a sagittal MR image for $\sigma_{\vec{\nabla}L} = 2$ and three values for k.

We see clearly the reduced conductivity at the edges (darker in the pictures), and we see that we can control the relative influence of the effect with the free parameter k. This parameter has the dimension of the gradient (meter^{-1}) as we want the exponent to be dimensionless.

In the rest of the examples we take the first choice for the conductivity coefficient: c_1. So, the Perona and Malik (P&M) equation becomes

$$\frac{\partial L}{\partial s} = \vec{\nabla}.\left(e^{-\frac{|\vec{\nabla}L|^2}{k^2}}\,\vec{\nabla}L\right) \tag{2}$$

Expanding the differential operators for the right hand side, we get in 1D:

```
∂x (Exp[- (∂x L[x])^2 / k^2] ∂x L[x]) // Simplify
```

$$\frac{e^{-\frac{L'[x]^2}{k^2}}\,(k^2 - 2\,L'[x]^2)\,L''[x]}{k^2}$$

and in 2D:

```
k = .;
PM = ∂x (E^-((∂x L[x,y])^2 + (∂y L[x,y])^2)/k^2 ∂x L[x, y]) + ∂y (E^-((∂x L[x,y])^2 + (∂y L[x,y])^2)/k^2 ∂y L[x, y]) //
  FullSimplify; PM // shortnotation
```

$$\frac{e^{-\frac{L_x^2+L_y^2}{k^2}}\,((k^2 - 2\,L_x^2)\,L_{xx} - 4\,L_x\,L_{xy}\,L_y + (k^2 - 2\,L_y^2)\,L_{yy})}{k^2}$$

We recognize the strongly nonlinear character of this equation. Unfortunately, there is no analytical solution for this PDE. So we have to rely on numerical methods to approximate the solution. Fortunately there are many efficient and stable numerical schemes for the time-evolution of an image governed by this type of *divergence of a flow*-type PDE's. We will discuss some of them in the course of this chapter.

The most straightforward numerical approximation of $\frac{\partial L}{\partial s} = \nabla.c\,\nabla L$ is the *forward-Euler* approximation $\delta L = \delta s\,(\nabla.c\,\nabla L)$ where δL is the increment in L and δs is the (typically small) stepsize in scale: the *evolution stepsize*. Through iteration (calculation of many small consecutive increments) we can calculate the image at the required level of evolution, i.e. at the required level of adaptive blurring.

The images in between form a scale-space again, or alternatively, a series of images over evolution-time. For the limit $k \to \infty$, we get the linear diffusion equation again:

```
Limit[PM, k -> ∞] // shortnotation
```

$$L_{xx} + L_{yy}$$

21.4 Scale-space implementation of the P&M equation

To work on discrete images, we replace (with the **Replace** operator, short notation **/.**) every occurrence of a spatial derivative in the right-hand side of the P&M equation (**pmc1** resp. **pmc2**) with the scaled Gaussian derivative operator **gD**:

For c_1:

```
Clear[im, σ, k]; c1 = E^-((∂x L[x,y])^2 + (∂y L[x,y])^2)/k^2;
pmc1[im_, σ_, k_] = ∂x (c1 ∂x L[x, y]) + ∂y (c1 ∂y L[x, y]) /.
   Derivative[n_, m_][L][x_, y_] → gD[im, n, m, σ] // Simplify
```

$$\frac{1}{k^2}\left(e^{-\frac{gD[im,0,1,\sigma]^2+gD[im,1,0,\sigma]^2}{k^2}}\left((k^2-2\,gD[im,0,1,\sigma]^2)\,gD[im,0,2,\sigma]-4\,gD[im,0,1,\sigma]\,gD[im,1,0,\sigma]\,gD[im,1,1,\sigma]+(k^2-2\,gD[im,1,0,\sigma]^2)\,gD[im,2,0,\sigma]\right)\right)$$

For c_2:

```
Clear[im, σ, k]; c2 = 1 / (1 + ((∂x L[x, y])^2 + (∂y L[x, y])^2)/k^2);
pmc2[im_, σ_, k_] = ∂x (c2 ∂x L[x, y]) + ∂y (c2 ∂y L[x, y]) /.
   Derivative[n_, m_][L][x_, y_] → gD[im, n, m, σ] // Simplify
```

$$(k^2\,(gD[im,0,2,\sigma]\,(k^2-gD[im,0,1,\sigma]^2+gD[im,1,0,\sigma]^2)-4\,gD[im,0,1,\sigma]\,gD[im,1,0,\sigma]\,gD[im,1,1,\sigma]+gD[im,0,1,\sigma]^2\,gD[im,2,0,\sigma]+(k^2-gD[im,1,0,\sigma]^2)\,gD[im,2,0,\sigma]))/(k^2+gD[im,0,1,\sigma]^2+gD[im,1,0,\sigma]^2)^2$$

We calculate the variable conductance diffusion first on a simple small (64x64) noisy test image of a black disk (minimum: 0, maximum: 255):

```
imdisk = Table[If[(x - 32)^2 + (y - 32)^2 < 300, 0, 255], {y, 64}, {x, 64}];
noise = Table[100 Random[], {64}, {64}]; im = imdisk + noise;
ListDensityPlot[im, ImageSize -> 120];
```

Figure 21.2 Simple 64x64 test image of a black disk on a white background (intensities 0 resp. 255) with additive uniform noise (amplitude=100).

A rule for the choice of k is difficult to give. It depends on the choice of which edges have to be enhanced, and which have to be canceled. The histogram of gradient values (at $\sigma = 1$) may give some clue to how much 'relative edge strength' is present in the image:

```
Histogram[Flatten[grad = √(gD[im, 1, 0, 1]^2 + gD[im, 0, 1, 1]^2)],
  HistogramCategories → Range[0, 200, 5], ImageSize -> 280];
```

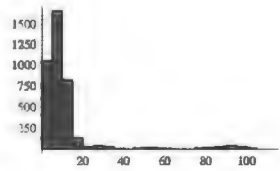


Figure 21.3 Histogram of the (scalar) magnitude of the gradient values at $\sigma = 1$ of the disk image of figure 21.2.

In section 21.5 we derive that k determines the 'turnover' point of edge reduction versus enhancement. A forward-Euler approximation scheme now becomes particularly simple:

```
peronamalikc1[im_, δs_, σ_, k_, niter_] := Module[{}, evolved = im;
Do[evolved += δs (pmc1[evolved, σ, k]), {niter}]; evolved];
```

where **im** is the input image, **δs** is the time step, **σ** is the scale of the differential operator, **k** is the conductivity control parameter and **niter** is the number of iterations. Here is an example of its performance:

```
line = {Red, Line[{{0, 32}, {64, 32}}]}; DisplayTogetherArray[
 {ListDensityPlot[#, Epilog -> line] & /@
   {im, imp = peronamalikc1[im, .1, .7, 25, 20]},
  ListPlot /@ {im[[32]], imp[[32]]}}, ImageSize -> 380];
```

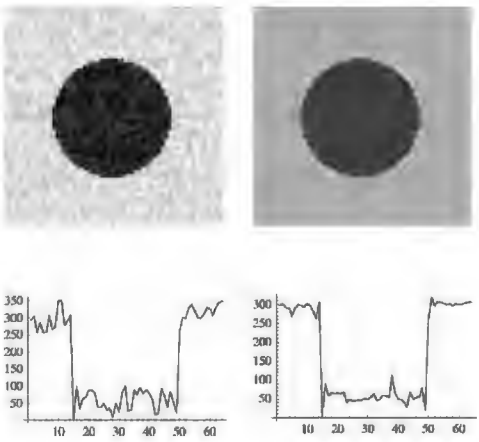


Figure 21.4 Top left: input image; top right: same image after variable conductance diffusion, with operator scale $\sigma = .8$ pixels, timestep $\delta s = 0.1$, $k = 100$, nr. of iterations = 10. Bottom row: intensity profile of middle row of pixels for both images. The edge steepness is well preserved, while the noise is substantially reduced. A little overshoot is seen at the edges.

We can define a signal-to-noise ratio (SNR) for this particular image by taking two square (16x16) areas, one in the middle of the black disk and one in the lower left corner in the background. The signal is defined as the difference of the means, the noise as the sum of the variances of the intensity values in the areas:

```
<< Statistics`DescriptiveStatistics`;
snr[im_] :=
  Module[{}, m1 = SubMatrix[im, {24, 24}, {16, 16}] // Flatten;
m2 = SubMatrix[im, {3, 3}, {16, 16}] // Flatten;
      Mean[m2] - Mean[m1]
    ------------------------- ];
   Variance[m1] + Variance[m2]

ListDensityPlot[im, Epilog -> {Hue[1], Thickness[.01],
    Line[{{3, 3}, {3, 19}, {19, 19}, {19, 3}, {3, 3}}],
    Line[{{24, 24}, {24, 40}, {40, 40}, {40, 24}, {24, 24}}]},
  ImageSize -> 150];
```

Figure 21.5 Areas through which the signal-to-noise ratio (SNR) is defined.

Clearly, the signal-to-noise ratio increases substantially during the evolution until $t = \text{niter} \times \delta s = 1$:

```
evolved = im; out = {}; σ = .8; δs = .1; k = 100; niter = 10;
Do[evolved += δs (pmc1[im, σ, k]);
  out = Append[out, snr[evolved]], {niter}];
ListPlot[out, PlotJoined -> True, AxesLabel ->
   {"evolution\ntime\n(in iterations)", "SNR"}, ImageSize -> 250];
```

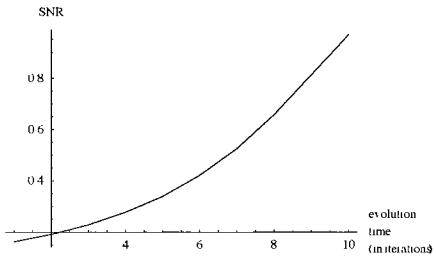


Figure 21.6 The signal-to-noise ratio (SNR) increases substantially with increasing evolution time.

But this cannot continue, of course, for physical reasons. When we continue the evolution until $t = 20$ (in units of iterations), we see that the gain is lost again:

```
evolved = im; out = {}; σ = .8; δs = .1; k = 100; niter = 20;
Do[evolved += δs (pmcl[im, σ, k]);
  out = Append[out, snr[evolved]], {niter}];
ListPlot[out, PlotJoined -> True, AxesLabel ->
   {"evolution\ntime\n(in iterations)", "SNR"}, ImageSize -> 190];
```

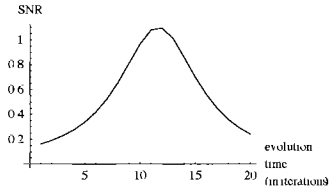


Figure 21.7 There is a maximum in the signal-to-noise ratio (SNR) for variable conductance diffusion with increasing evolution time.

▲ Task 21.1 Play with this maximum in SNR as a function of timestep, σ and k.

▲ Task 21.2 Show that this maximum in SNR occurs for $t = 4\sigma - 2$.

▲ Task 21.3 Investigate the Perona & Malik equation and the SNR for 3D.

A good empirical value for k is the 80% or 90% percentile of the cumulative frequency distribution of the gradient magnitude values in the image. Of course, this is a 'rule of thumb', the best choice may depend on the image *and the task* at hand and may need some experimentation. The function **kpercentile[im, σ, perc, nbins]** calculates the percentile value k_{perc} for the image **im**, gradient scale **σ**, percentage **perc** ([0-1]) in the number of bins **nbins**:

```
Needs["Statistics`DataManipulation`"];

kpercentile[im_, σ_, perc_, nbins_] := Module[{max, cummax},
   grad2 = √(gD[im, 1, 0, σ]^2 + gD[im, 0, 1, σ]^2); max = Max[grad2];
counts = BinCounts[Flatten[grad2], {0, max, max / nbins}];
cumcounts = Rest[FoldList[Plus, 0, counts]];
   cummax = Max[cumcounts];
Length[Select[cumcounts, (# < perc cummax) &]] max / nbins];
```

Here is the result for an image of man-made structures. We first find an estimate for k at $\sigma = 1$:

```
im = Import["Utrecht256.gif"][[1, 1]];
k90 = kpercentile[im, 1, .9, 100]
```

```
77.1295
```

We choose deliberately a smaller $k = 25$ to get more blurring overall:

```
δs = .02; σ = 1; k = 25; evolved = im;
Do[evolved += δs (pmcl[im, σ, k]), {60}]
Show[GraphicsArray[ListDensityPlot[#, DisplayFunction → Identity] & /@
    {im, evolved}], ImageSize -> 400];
```

Figure 21.8 Left: original image, resolution 256x256. Right: Variable conductance diffusion with $k = 25$, $\sigma = 1$, $\delta s = 0.02$ and 60 iterations. Note the good keeping of the gradient steepness at the edges and the reduction of the brick and rooftile texture.

21.5 The P&M equation is ill-posed.

It is instructive to study the P&M equation in somewhat more detail. Let us look how the diffusion process depends on the gradient strength, so we consider (in 1D for simplicity) $c = c(L_x)$. So the P&M equation is $L_s = \frac{\partial}{\partial x}(c(\frac{\partial L}{\partial x})\frac{\partial L}{\partial x}) = c' L_x + c L_{xx}$. Suppose that the flow (or flux function) $c L_x$ is decreasing with respect to L_x at some point x_0, then we have $\frac{\partial}{\partial L_x}(c L_x) = c + c' L_x = -a$ with $a > 0$. Now $c' = \frac{\partial c}{\partial L_x}$. So the nonlinear diffusion equation reads in this situation $L_s + a L_{xx} = 0$, from which we get $L_s = -a L_{xx}$. Locally we have an *inverse heat equation* which is well known to be ill-posed. This heat equation locally blurs or *de*blurs, dependent on the condition of c. We study this condition in 1D. For c_1 and c_2 we find:

```
Clear[k, Lx, c1, c2]; c1[Lx_] := Exp[-Lx^2 / k^2];
c2[Lx_] := 1 / (1 + Lx^2 / k^2); ∂Lx (c1[Lx] Lx) // Simplify
```

$$\frac{e^{-\frac{Lx^2}{k^2}}\,(k^2 - 2\,Lx^2)}{k^2}$$

```
∂Lx (c2[Lx] Lx) // Simplify
```

$$\frac{k^4 - k^2\,Lx^2}{(k^2 + Lx^2)^2}$$

The function $c_1 L_x$ decreases for $L_x > \frac{1}{2}\sqrt{2}\,k$ and $c_2 L_x$ decreases for $L_x > k$. This implies that with k we can adjust the *turnover point* in the gradient strength, below which we have blurring, and above which we have deblurring. This is a favorable property: small edges disappear through blurring during the evolution, while the stronger edges are not only kept but even made steeper through the deblurring. Here are the graphs of the flux $c\,L_x$ and of $\frac{\partial(c\,L_x)}{\partial L_x}$ for both c's with $k = 2$:

```
k = 2; DisplayTogetherArray[
 {{Plot[c1[Lx] Lx, {Lx, 0, 5}, AxesLabel → {"Lx", "flow c1 Lx"}],
   FilledPlot[Evaluate[∂Lx (c1[Lx] Lx)],
    {Lx, 0, 5}, AxesLabel → {"Lx", "∂Lx (c1 Lx)"},
    epilog = Epilog -> {Text["blurring", {1.5, .8}], Arrow[{1.5, .7}, {1.1, .5}],
       Text["deblurring", {3, .4}], Arrow[{3, .3}, {2.8, .1}]}]},
  {Plot[c2[Lx] Lx, {Lx, 0, 5}, AxesLabel → {"Lx", "flow c2 Lx"}],
   FilledPlot[Evaluate[∂Lx (c2[Lx] Lx)], {Lx, 0, 5},
    AxesLabel → {"Lx", "∂Lx (c2 Lx)"}, epilog]}, ImageSize -> 310];
```

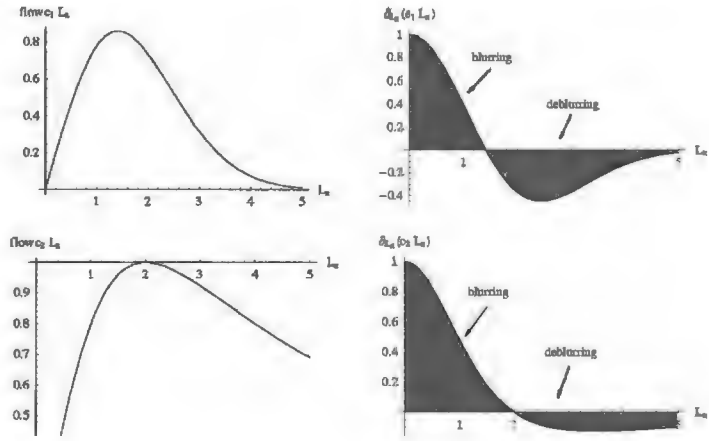


Figure 21.9 The value of k determines the turnover point of the direction of diffusion. Top row: Flow function and its derivative with respect to the gradient for conductivity function c_1. Bottom row: idem for c_2. For both c's: $k = 2$. For positive slope of the flux as a function of the gradient we get local blurring, for negative slope we get local deblurring.

We saw before that a reasonable value for k is the 90% percentile, i.e. all edges with strength below this value will be smoothened out, and all edges stronger than this value will be enhanced.

The original formulation by Perona and Malik employed nearest neighbor differences in 4 directions to calculate the local gradient strength. This introduces artefacts because there is a bias for direction. We now understand that the Gaussian derivative kernel is the appropriate regularized differential operator, which does not introduce a bias for direction. This was introduced first by Catté, Lions, Morel and Coll [Catté et al. 1992].

21.6 Von Neumann stability of numerical PDE's

When we approximate a partial differential equation with finite differences in the forward Euler scheme, we want to make large steps in evolution time (or scale) to reach the final evolution in the fastest way, with as little iterations as possible. How large steps are we allowed to make? In other words, can we find a criterion for which the equation remains stable? A famous answer to the question of stability was derived by Von Neumann [Ames 1977a], and is called the *Von Neumann stability criterion*. We explain the concept with a simple 1D evolution equation, the 1D linear diffusion equation: $\frac{\partial L}{\partial t} = \frac{\partial^2 L}{\partial x^2}$.

This equation can be approximated with finite differences as $\frac{L_j^{n+1} - L_j^n}{\Delta t} = \frac{L_{j+1}^n - 2L_j^n + L_{j-1}^n}{\Delta x^2}$, where we use a forward derivative for the time derivative in the left-hand side of the equation and centered differences for the second order spatial derivative in the right-hand side. The upper index n denotes the moment in time, the lower index j denotes the spatial pixel position. We define $R = \frac{\Delta t}{\Delta x^2}$, so we rewrite $L_j^{n+1} - L_j^n - R(L_{j+1}^n - 2L_j^n + L_{j-1}^n) = 0$, in *Mathematica* (we define the finite difference function `f[j,n]`):

```
Clear[L, f];
f[j_, n_] := L[j, n + 1] - L[j, n] - R (L[j + 1, n] - 2 L[j, n] + L[j - 1, n])
```

Let the solution L_j^n of our PDE be a generalized exponential function, with k a general (spatial) wavenumber:

```
Clear[ξ, j, n, k, Δx];
L[j_, n_] := ξ^n Exp[I j k Δx]
```

When we insert this solution in our discretized PDE, we get

```
f[j, n]
```

$$-e^{i\,j\,k\,\Delta x}\,\xi^n + e^{i\,j\,k\,\Delta x}\,\xi^{1+n} - R\,(e^{i\,(-1+j)\,k\,\Delta x}\,\xi^n - 2\,e^{i\,j\,k\,\Delta x}\,\xi^n + e^{i\,(1+j)\,k\,\Delta x}\,\xi^n)$$

We want the increment function $f(j, n)$ to be maximal on the domain j, so we get the condition $\frac{\partial f(j,n)}{\partial j} = 0$. The *Mathematica* function **`ExpToTrig`** rewrites complex exponentials into sinusoidal functions, using Euler's formula $e^{i\phi} = \cos(\phi) + i\sin(\phi)$:

```
ExpToTrig[E^(I ϕ)]
```

```
Cos[ϕ] + i Sin[ϕ]
```

The maximum criterion $\frac{\partial f(j,n)}{\partial j} = 0$ can be solved for ξ:

```
Solve[ExpToTrig[∂j f[j, n]] == 0, ξ] // Simplify
```

```
{{ξ → 1 - 2 R + 2 R Cos[k Δx]}}
```

The amplitude ξ^n of the solution $\xi^n e^{i\,j\,k\,\Delta x}$ should not explode for large n, so in order to get a stable solution we need the criterion $|\xi| \leq 1$. This means, because $\mathrm{Cos}(k\,\Delta x) - 1$ is always non-positive, that

$$R = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{3}$$

This is the Von Neumann criterion. For the 2D case we can prove in a similar way (assuming $\Delta x = \Delta y$): $R_{2D} = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{4}$.

▲ Task 21.4 Show that the Von Neumann criterion for N-dimensional isotropic diffusion equals $\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2N}$.

This is an essential result. When we take a too large step size for Δt in order to reach the final time faster, we may find that the result gets unstable. We show an example in section 21.8. The Von Neumann criterion gives us the fastest way we can get to the iterative result. It is safe to stay well under the maximum value, to not compromise this stability close to the criterion.

The pixelstep Δx is mostly unity, so the maximum evolution stepsize should be $\Delta t < \frac{1}{4}$ (pixel^2). This is indeed a strong limitation, making many iteration steps necessary. Gaussian derivative kernels improve this situation considerably, as we will see in the next section.

21.7 Stability of Gaussian linear diffusion

The numerical stability criterion for Gaussian derivative implementations was derived by Niessen et al.[Niessen et al. 1997]. The difference is that we now do not take the nearest neighbor differences for an approximation of the derivatives, but a regular convolution with the Gaussian aperture function, as scale-space theory prescribes. We do the analysis in 1D first, then in 2D.

We start again with a general possible solution for the luminance function $L(x, j, n)$, where x is the spatial coordinate, j is the discrete spatial grid position, and n is the discrete moment in evolution time of the PDE.

```
Clear[g, gxx, s, ξ, j, k, n, Δx];
L[x_, j_, n_] := ξ^n Exp[I j x];
```

We define the Gaussian kernel and the Laplacian (in 1D it is just the second order spatial derivative):

```
g[x_] := 1/Sqrt[4 π s] Exp[-x^2/(4 s)];
gxx[x_] = Simplify[∂_{x,x} g[x], s > 0]
```

$$\frac{e^{-\frac{x^2}{4s}}\,(-2s + x^2)}{8\sqrt{\pi}\,s^{5/2}}$$

We recall that the convolution of a function f with a kernel g is defined as $f \otimes g = \int_{-\infty}^{\infty} f(y)\, g(y - x)\, dx$. The dummy variable y shifts the kernel over the full domain of

f, and the result $f \otimes g$ is a function of x. So for discrete location j at timestep n we get for the blurred intensity:

```
convolved[x_] = Simplify[∫_-∞^∞ L[y, j, n] gxx[y - x] d y, s > 0]
```

$$-e^{j\,(-j\,s+i\,x)}\,j^2\,\xi^n$$

If we compare this with the original intensity function, we find a multiplication factor $-e^{-j^2 s} j^2$:

```
factor = convolved[x] / L[x, j, n] // Simplify
```

$$-e^{-j^2\,s}\,j^2$$

```
s = 1; Plot[factor, {j, 0, 3},
 AxesLabel -> {"j (time)", "factor"}, ImageSize -> 230];
```

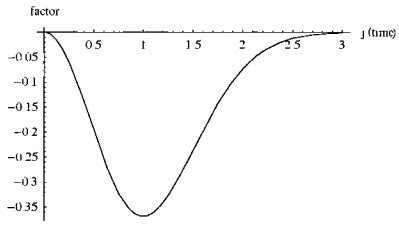


Figure 21.10 There is a clear minimum of the multiplication factor $-e^{-j^2 s} j^2$ as a function of j. With this value the largest steps in the evolution can be made, giving the fastest way to the result.

We are looking for the largest absolute value of the factor, because then we take the largest evolution steps. Because the factor is negative everywhere we need to find the minimum of the factor with respect to j, i.e. $\frac{\partial \text{factor}}{\partial j} = 0$:

```
Clear[s]; solution = Solve[∂j factor == 0, j]
```

$$\left\{\{j \to 0\}, \left\{j \to -\frac{1}{\sqrt{s}}\right\}, \left\{j \to \frac{1}{\sqrt{s}}\right\}\right\}$$

Only positive times j make sense, so we take $j = \frac{1}{\sqrt{s}}$. We then find for the maximum size of the timestep factor $\frac{-1}{e\,s}$:

```
factor /. j -> 1/√s
```

$$-\frac{1}{e\,s}$$

So we find for the Gaussian derivative implementation $\xi = 1 - \frac{\Delta t}{e\,s}$, so $|\xi| \leq 1$ implies $\frac{\Delta t}{e\,s} \leq 2$, thus $\Delta t \leq 2\,e\,s$. Introducing this in the time-space ratio $R = \frac{\Delta t}{(\Delta x)^2}$ we finally get the limiting stepsize for a stable solution under Gaussian blurring:

$$R = \frac{\Delta t}{\Delta x^2} \leq 2\,e\,s = e\,\sigma^2 \tag{4}$$

Note that this enables substantially larger stepsizes then in the nearest neighbor case. E.g. when $\sigma = 2$, we get (for $\Delta x = 1$) a timestep of $\Delta t \leq 10.87$. We give the reasoning for the 2D case for completeness. We define a 2D image $L(x, y, j, k, n)$ on our discrete grid with (x, y) the spatial coordinates, (j, k) the spatial integer grid positions, and n the discrete evolution time:

```
Clear[x, y, j, k, n]; L[x_, y_, j_, k_, n_] := ξ^n E^(I j x) E^(I k y);
g = 1/(4 π s) Exp[-(x^2 + y^2)/(4 s)]; laplacian[x_, y_] = Simplify[∂_{x,x} g + ∂_{y,y} g, s > 0]
```

$$\frac{e^{-\frac{x^2+y^2}{4s}}\,(-4s + x^2 + y^2)}{16\,\pi\,s^3}$$

First we integrate with respect to the shift α in x, then to the shift β in y (we may do this due to the separability of the Gaussian kernel, and thus of the Laplacian function):

```
convolution1 = Simplify[∫_{-∞}^{∞} L[α, β, j, k, n] laplacian[α - x, β - y] dα, s > 0]
```

$$\frac{e^{-\frac{4j^2s^2 - 4ijsx + y^2 - 4iksβ - 2yβ + β^2}{4s}}\,(-2s - 4j^2s^2 + (y-\beta)^2)\,\xi^n}{8\sqrt{\pi}\,s^{5/2}}$$

```
convolution = Simplify[∫_{-∞}^{∞} convolution1 dβ, s > 0]
```

$$-e^{-j^2 s - k^2 s + i j x + i k y}\,(j^2 + k^2)\,\xi^n$$

The factor relative to the original luminance function is now:

```
factor = convolution / L[x, y, j, k, n] // Simplify
```

$$-e^{-(j^2+k^2)s}\,(j^2 + k^2)$$

This function has a minimum at $\left\{\frac{\partial \text{factor}}{\partial j} = 0,\ \frac{\partial \text{factor}}{\partial k} = 0\right\}$:

```
solution = Solve[{∂_j factor == 0, ∂_k factor == 0}, s]
```

$$\left\{\left\{s \to \frac{1}{j^2 + k^2}\right\}\right\}$$

From the graph of the factor we appreciate that the solution space is on a circle in the spatial (j, k) domain:

```
s = 1; Plot3D[factor, {j, 0, 3}, {k, 0, 3}, Axes -> True,
 AxesLabel -> {"j", "k", "factor"}, ImageSize -> 320]; Clear[s];
```

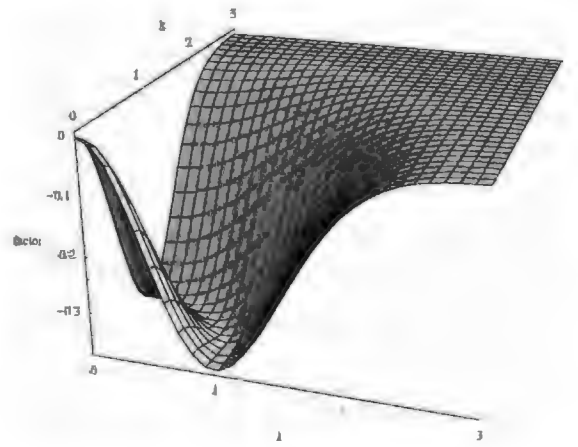


Figure 21.11 There is a clear circular line of minima of the multiplication factor $-e^{-(j^2+k^2)s}(j^2+k^2)$ as a function of j and k. With these values the largest steps in the evolution can be made, giving the fastest way to the result.

So $j^2 + k^2 = \frac{1}{s}$. As in the 1D case, we find for the minimum factor $\frac{-1}{e\,s}$:

```
factor /. {j^2 + k^2 -> 1/s}
```

$$-\frac{1}{e\,s}$$

So in 2D we find for the Gaussian derivative implementation the same result:

$$R = \frac{\Delta t}{\Delta x \Delta y} \leq 2\,e\,s = e\,\sigma^2 \qquad (5)$$

21.8 A practical example of numerical stability

The following example is from [Niessen 1997a]. The stability criterion for Gaussian implementation is $\frac{\Delta s}{(\Delta x)^2} < 2\,e\,s = e\,\sigma^2$ (because $s = \frac{1}{2}\sigma^2$). For the Gaussian blurring of an image with $\sigma = 0.8$ pixels for the Laplacian operator, we get $\Delta s < e\,0.8^2 = 1.74$. Let us study the effect of taking a range of evolution steps from somewhat smaller to somewhat larger than the critical evolution step $\Delta s = 1.74$. As test image we take a white circle on a black background. We blur this image to $\sigma = \sqrt{128}$ pixels (which is to $s = 64$ pixels^2) in two ways: a) with normal Gaussian convolution and b) with the numerical implementation of the diffusion equation and Gaussian derivative calculation of the Laplacian.

```
disk = Table[If[(x - 64)^2 + (y - 64)^2 < 1000, 1, 0], {y, 128}, {x, 128}];
diskblurred = gD[disk, 0, 0, Sqrt[128]];
```

```
DisplayTogetherArray[{ListDensityPlot[disk],
   ListDensityPlot[diskblurred]}, ImageSize → 280];
```

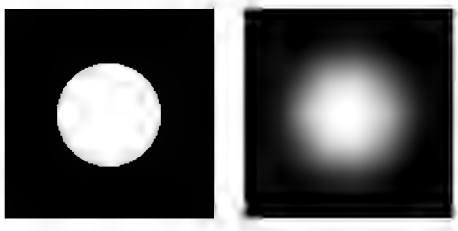


Figure 21.12 Test image and its Gaussian blurred version at $\sigma = \sqrt{128}$ pixels. Resolution image 128 x 128.

This function implements the numerical approximation of the linear, isotropic blurring of the image **im**:

```
num[im_, nrsteps_, σ_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im;
   Do[imt += δs (gD[imt, 2, 0, σ] + gD[imt, 0, 2, σ]), {nrsteps}]; imt];
```

To show the critical effect of the Von Neumann stability criterion, we evolve the image to a scale $s = 64$ and plot the numerical result for a narrow range of timestep sizes (timestep = **64 / nsteps**) around the critical value 1.74:

```
DisplayTogetherArray[
  Table[ListDensityPlot[num[disk, nsteps, .8, 64],
    PlotLabel → "timestep = " <> ToString[N[64 / nsteps]]],
   {nsteps, 35, 39}], ImageSize -> 400];
```

Figure 21.13 Study of the influence of time stepsize on the accuracy of the result in the numerical approximation of the linear diffusion equation. Image resolution 128^2, evolution to $s = 64$ pixel^2, scale Laplacian operator $\sigma = 0.8$ pixel. The critical timestep $\Delta s = e\,\sigma^2 = 2.1718 \times 0.8^2 = 1.74$.

▲ Task 21.5 Why are the artefacts at left/right, top/bottom?

It is clear from the results that timesteps larger than the critical timestep (two left cases) lead to erroneous results. It is safe to take the timestep at least 10% smaller than the critical timestep. Note the amazingly sharp transition around the critical timestep.

21.9 Euclidean shortening flow

Alvarez, Guichard, Lions and Morel [Alvarez 1992a] realized that the P&M variable conductance diffusion was complicated by the choice of the parameter k.

They reasoned that the principal influence on the local conductivity should be to direct the flow in the *direction* of the gradient only: we want a lot of diffusion along the edges, but virtually no diffusion across the edges. This led to the proposal to make the intensity flow a function of the unit gradient vector $\frac{\vec{\nabla}L}{|\nabla L|} = (\cos(\phi), \sin(\phi))$, so we get the (nonlinear) diffusion equation

$$\frac{\partial L}{\partial s} = \left|\vec{\nabla}L\right| \vec{\nabla}.\left(\frac{\vec{\nabla}L}{|\nabla L|}\right) \tag{6}$$

In *Mathematica*:

```
Clear[L]; << Calculus`VectorAnalysis`;
SetCoordinates[Cartesian[x, y, z]];
∇L = Grad[L[x, y, 0]]
```

$$\{L^{(1,0,0)}[x, y, 0], L^{(0,1,0)}[x, y, 0], 0\}$$

```
√(∇L .∇L) Div[∇L / √(∇L.∇L)] // Simplify // shortnotation
```

$$\frac{-2\,L_x\,L_{xy}\,L_y + L_{xx}\,L_y^2 + L_x^2\,L_{yy}}{L_x^2 + L_y^2}$$

This is exactly $\kappa \left|\vec{\nabla}L\right|$, i.e. the isophote curvature κ (see chapter 6) times the gradient magnitude $\left|\vec{\nabla}L\right|$. From the discussion of the *gauge coordinates* we recall that this is equal to the second order derivative in the direction tangential to the isophote: $\kappa \left|\vec{\nabla}L\right| = L_{vv}$.

The nonlinear diffusion according to this paradigm becomes particularly concise:

$$\frac{\partial L}{\partial s} = L_{vv} \tag{7}$$

Because the Laplacian $\Delta L = L_{xx} + L_{yy} = L_{vv} + L_{ww}$, we get $\frac{\partial L}{\partial s} = \Delta L - L_{ww}$. We see that we have corrected the normal diffusion with a factor proportional to the second order derivative in the gradient direction (in gauge coordinates: L_{ww}). This subtractive term cancels the diffusion in the direction of the gradient. This gives us also a recipe for 3D: $\frac{\partial L}{\partial s} = \Delta L - L_{ww}$ where $\Delta L = L_{uu} + L_{vv} + L_{ww} = L_{xx} + L_{yy} + L_{zz}$ is the 3D Laplacian and u and v are the gauge directions tangential to the isophote surface.

The enthusiasm of Alvarez et al. about this equation led them to the name 'fundamental equation of image processing' (!). This PDE is better known as the *Euclidean shortening flow* equation, for reasons we will discuss in the next section.
There are a number of differences between this equation and the Perona & Malik equation: - the flow (of flux) is independent of the magnitude of the gradient; - There is *no* extra free parameter, like the edge-strength turnover parameter k; - in the P&M equation the diffusion

decreases when the gradient is large, resulting in contrast dependent smoothing; - this equation is gray-scale invariant.

21.10 Grayscale invariance

A function is grayscale invariant, if the function does not change value when the grayscale function L is modified by a monotonically increasing or decreasing function $f(L)$, $f \neq 0$. Grayscale invariance is an attractive property: image analysis is the same when we change e.g. the contrast or brightness on a monitor, or if we put up sunglasses. The gradient is dependent on f:

```
Clear[f, L]; u = f[L[x, y, s]];
gradu = √((∂x u)^2 + (∂y u)^2) ; gradu // shortnotation
```

$$\sqrt{(L_x[x, y, s]^2 + L_y[x, y, s]^2)\, f'[L[x, y, s]]^2}$$

The Euclidean shortening flow equation $\frac{\partial L}{\partial s} = \left|\overline{\nabla} L\right| \overline{\nabla}.\left(\frac{\overline{\nabla} L}{\left|\overline{\nabla} L\right|}\right)$ is *independent* of f:

```
FullSimplify[∂s u - gradu (∂x (∂x u)/gradu + ∂y (∂y u)/gradu) == 0, f'[L[x, y, s]] ≠ 0] //
 shortnotation
```

$$(2\, L_x[x, y, s]\, L_{xy}[x, y, s]\, L_y[x, y, s] + L_y[x, y, s]^2\, (-L_{xx}[x, y, s] + L_z[x, y, s]) + L_x[x, y, s]^2\, (-L_{yy}[x, y, s] + L_z[x, y, s]))\,/\,(L_x[x, y, s]^2 + L_y[x, y, s]^2) == 0$$

The function f does not show up anymore: the equation is greyscale invariant.

21.11 Numerical examples shortening flow

Let us study some numerical examples of this PDE on a simple image. We first define the numerical iterative approximation to the nonlinear PDE:

```
euclideanshortening[im_, nrsteps_, σ_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im;
   Do[imt += δs (gD[imt, 0, 2, σ] gD[imt, 1, 0, σ]^2 -
         2 gD[imt, 0, 1, σ] gD[imt, 1, 0, σ] gD[imt, 1, 1, σ] +
         gD[imt, 0, 1, σ]^2 gD[imt, 2, 0, σ]) /
       (gD[imt, 0, 1, σ]^2 + gD[imt, 1, 0, σ]^2), {nrsteps}]; imt];
```

In gauge coordinates the expression becomes more compact:

```
euclideanshortening[im_, nrsteps_, σ_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im;
   Do[imt += δs gauge2DN[im0, 0, 2, σ] /. im0 -> imt, {nrsteps}]; imt];
```

We study the disk image again, with additive uniform uncorrelated noise:

```
disk = Table[
   If[(x - 64)^2 + (y - 64)^2 < 1000, 1, 0] + Random[], {y, 128}, {x, 128}];
diskblurred = gD[disk, 0, 0, √128];
DisplayTogetherArray[{ListDensityPlot[disk],
   ListDensityPlot[diskblurred]}, ImageSize → 230];
```

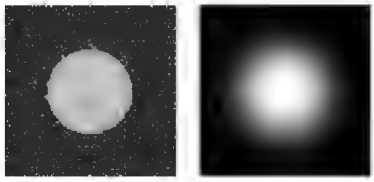


Figure 21.14 Noisy disk image. Image resolution 128^2, disk=1, background=0, noise amplitude [0-1]. Right: image blurred with $\sigma = \sqrt{128}$, or $s = 64$ with linear Gaussian diffusion. By blurring the image the noise is gone, but the edge is gone too.

The critical timestep for this numerical scheme is again $\frac{\Delta t}{(\Delta x)} < 2\,e\,s$. We check this with the same settings as above for the linear diffusion example:

```
Block[{$DisplayFunction = Identity},
  pl = Table[ListDensityPlot[euclideanshortening[disk, nsteps, .8, 64],
     PlotLabel → "timestep = " <> ToString[N[64 / nsteps]]],
    {nsteps, 32, 42}]];
Show[GraphicsArray[Partition[pl, 5]], ImageSize → 355];
```

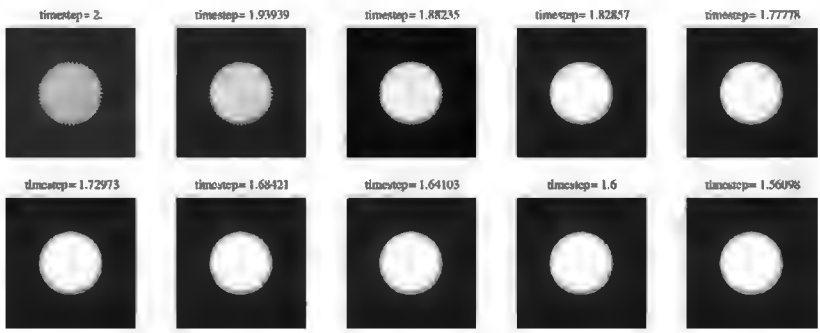


Figure 21.15 Euclidean shortening flow on the noisy disk image for a range of timestep sizes around the critical timestep $\Delta s = e\,0.8^2 = 1.74$. Clearly artefacts emerge for timestep sizes larger than the critical timestep.

And here is the evolution itself (we choose $s = 1.6$ pixels^2 as timestep) for $s = 1$ to 57 in 8 steps, with the scale of the flux operator $\sigma = 0.8$:

```
Block[{$DisplayFunction = Identity}, pl = Table[nsteps = Ceiling[s / 1.6];
    ListDensityPlot[euclideanshortening[disk, nsteps, .8, s],
     PlotLabel → "time = " <> ToString[s]], {s, 1, 57, 8}]];
```

```
Show[GraphicsArray[Partition[p1, 4]], ImageSize → 510];
```

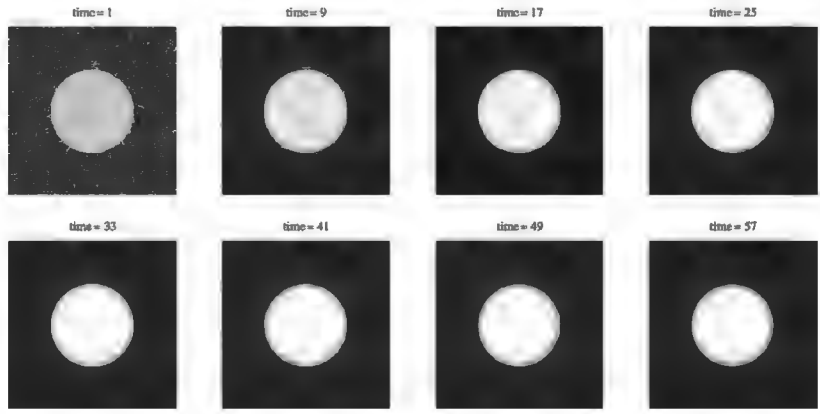


Figure 21.16 Euclidean shortening flow on the noisy disk image as a function of evolution time/scale (in pixel^2). Noise is significantly reduced, the edge is preserved.

The noise gradually disappears in this nonlinear scale-space evolution, while the edge strength is well preserved. Because the flux term, expressed in Gaussian derivatives, is rotation invariant, the edges are well preserved irrespective of their direction: this is *edge-preserving smoothing*..

```
DisplayTogetherArray[
  ListPlot[disk[[64]], PlotJoined -> True, PlotLabel → "original"],
  ListPlot[euclideanshortening[disk, 32, .8, 32][[64]],
   PlotJoined -> True, PlotLabel → "time = 31"], ImageSize -> 350];
```

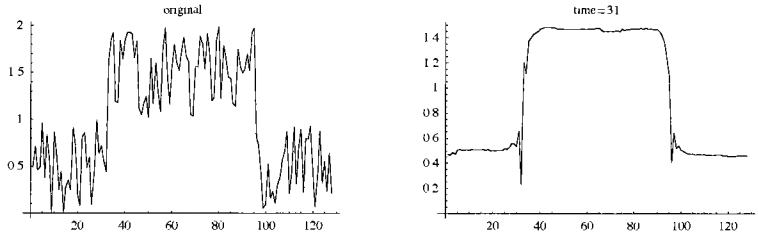


Figure 21.17 Plot of the middle horizontal row of pixel values for the original noisy disk image (left) and the image at time $t = 31$ with Euclidean shortening flow. Note the overshoot at the edges, and the good noise removal while keeping a steep edge.

This is an example for an ultrasound image with its particular speckle pattern:

```
us = Import["us.gif"][[1, 1]]; Block[{$DisplayFunction = Identity, s = 9},
 p1 = ListDensityPlot[us, PlotLabel → "Original"];
 p2 = ListDensityPlot[use = euclideanshortening[us, 6, .8, s],
   PlotLabel → "scale = " <> ToString[s]]];
```

```
Show[GraphicsArray[{p1, p2}], ImageSize → 500];
```

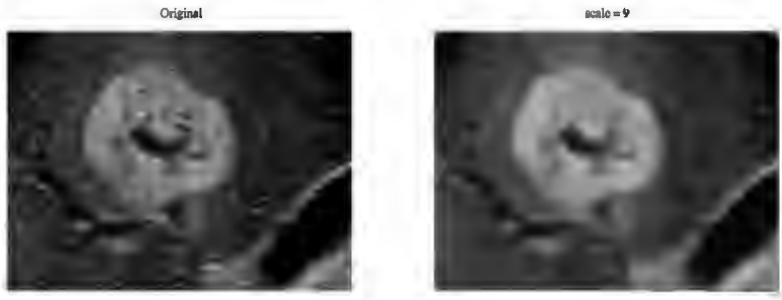


Figure 21.18 Euclidean shortening flow on a 260 x 345 pixel ultrasound image (source: www.atl.com).

Many improvements and acceleration schemes have been proposed recently. It is beyond the scope of this book to discuss these developments in detail. Good references to study are: Kacur and Mikula [Kacur2001], Weickert et al. [Weickert1997e, Weickert1998b].

21.12 Curve Evolution

We can consider an image as a collection of sampled luminance points. Alternatively, we can consider an image as a set of isophotes (or *level sets*). They too describe the image completely. When we consider the evolution of this set of isophotes under the control of a nonlinear diffusion equation, we consider the *evolution of curves*. This has become an important branch of nonlinear evolutionary image processing. Mathematically a lot was known about the evolution of curves and surfaces, among others from the studies of the propagation with time of the firefronts (iso-temperature surfaces) in flames and combustion [Sethian 1982].

Consider a parameterized closed curve without self-intersections $C(p): S^1 \to \mathbb{R}$, and let $\vec{x} = \vec{x}(p)$ be a regular parametric representation of the curve C with $\frac{\partial \vec{x}}{\partial p} \neq 0$. The evolution of the curve makes sure that the curve deforms. These deformations are functions of local geometry, so we get in general a motion of each point on the curve. This deforming motion can be decomposed in a component along the inward unit normal vector $\vec{N}$, and a component along the tangential unit vector $\vec{T}$. When the parameter t measures evolution time, we get $\frac{\partial \vec{x}}{\partial t} = \alpha(p, t)\,\vec{T} + \beta(p, t)\,\vec{N}$.

But an infinitesimal displacement along the tangential unit vector has no effect. We could as well see the curve as a rope and shift the rope while keeping the same form of the curve itself: only a displacement of points perpendicular to the curve will shift the shape of it. This was first proved by Gage [Gage 1983], who showed that the previous equation is equivalent to $\frac{\partial \vec{x}}{\partial t} = \beta'(p, t)\,\vec{N}$ with $\vec{x}(p, 0) = \vec{x}_0(p)$.

We consider planar curves, which are totally determined by their curvature as we recall from chapter 6. So, the first order expansion for β' in terms of its curvature is $\frac{\partial x}{\partial t} = (\beta_0 + \beta_1 \kappa)\vec{N}$.

This describes a deformation characterized by a constant motion term $\beta_0 \vec{N}$ and a motion due to curvature $\beta_1 \kappa \vec{N}$, both in the direction of the inward normal.

So we can discriminate three interesting cases:
- $\beta_0 = 0$, $\beta_1 \neq 0$: pure curvature motion. This flow is Alvarez's fundamental equation discussed before, and is known as the *Euclidean shortening flow*.

The reason that it is called this way we will discuss in the next section. This flow evolves concave regions to convex regions [Gage and Hamilton 1986], and shrinks convex regions to circular points [Gage 1983, Gage 1984]. Curvature motion has a smoothing effect on curves.
- $\beta_0 \neq 0$, $\beta_1 = 0$: constant motion or *normal motion flow*. The isophotes move in the direction of their normal with velocity β_0. This flow is intimately related to the *dilation* and *erosion* operators with a disk as structuring element from mathematical morphology. We will discuss mathematical morphological operators in the next section in more detail. Constant motion has a sharpening effect on curves.
- $\beta_0 \neq 0$, $\beta_1 \neq 0$: both curvature and constant motion. Both deformations are in a sense each other's opposite and could be appropriately weighted, prescribed by the task, to come to a satisfactory shape deformation. This framework was developed by Kimia et al. as the 'reaction-diffusion' (sharpening resp. smoothing) framework [Kimia 1992, Kimia 1996a]. Note that β_0 and β_1 do not have the same dimensionality, so we need to work with *natural coordinates* ($\hat{x} \rightarrow \frac{x}{\sigma}$, see chapter 6). This 2-dimensional scale-space with parameters β_0 and β_1 is also known as the *entropy scale-space*.

21.13 Duality between PDE- and curve evolution formulation

It was proven by Osher and Sethian [Osher and Sethian 1988, Sethian 1990a] that for all points on a level set (defined by $L(x) = a$ where a is a constant) for all a and where $\vec{\nabla} L \neq 0$, and with the evolution of the intensities of the image (the generation of a scale-space) specified by the evolution equation $\frac{\partial L}{\partial t} = \vec{\nabla}.\vec{F}(\partial_i L,\ \partial_{i,i} L,\ \ldots)$, then the levelsets of L evolve governed by the following curve evolution equation: $\frac{\partial C}{\partial t} = \frac{-\vec{\nabla}.\vec{F}(\partial_i L, \partial_{i,i} L, \ldots)}{|\nabla L|}\vec{N}$.

Here $\vec{F}$ is the (arbitrary) flow vector expressed in any partial spatial derivative of the image, and $\vec{N}$ is the inward unit normal.

In general, a point on a curve can move in *any* direction. This direction can be decomposed as a component in the tangential (to the curve) direction $\vec{T}$, and a component in the normal direction $\vec{N}$. So the most general curve evolution equation is $\frac{\partial C}{\partial t} = \alpha \vec{T} + \beta \vec{N}$. However, a motion tangential to the curve in unnoticeable, the exact position of the curve (isophote in our situation) is irrelevant. Compare the situation with a rubber band running over two rotating wheels: two photographs taken at different moments show the band in different positions, but this cannot be seen on the pictures of the structureless band. This irrelevance of the component $\alpha \vec{T}$ was shown mathematically by Gage [Gage 1983].

So the level sets evolve according to $\frac{\partial C}{\partial t} = \beta \vec{N}$.

This is an important formulation, as we can now relate both schemes directly together. It implies that when we consider a curve evolution scheme $\frac{\partial C}{\partial t} = g\left(\kappa, \frac{\partial \kappa}{\partial p}, \frac{\partial^2 \kappa}{\partial p^2}, \ldots\right)\vec{N}$ of a curve $C(x(p), y(p))$ where p is the curve arclength parameterizing the curve and κ the isophote curvature, we know that the luminance L evolves according to $\frac{\partial L}{\partial t} = -L_w\, g\left(-\frac{L_{vv}}{L_w}, \ldots\right)$. We recall from chapter 6 that $\kappa = -\frac{L_{vv}}{L_w}$. The PDE for the evolution of the luminance for the entropy scale-space becomes now: $\frac{\partial L}{\partial t} = \beta_0 L_w + \beta_1 L_{vv}$, belonging to the class of Hamilton-Jacobi equations.
Olver, Sapiro and Tannenbaum [Olver 1994d] give a very useful generalization to the Euclidean shortening flow equation. They show that when the evolution of the curve C is expressed in the second order derivative with respect to the intrinsic curve arclength r, i.e. $\frac{\partial C}{\partial t} = \frac{\partial^2 C}{\partial r^2}$, then this equation defines a flow which is invariant to any Lie group action (an action which can be expressed in infinitesimal form, like translations, rotations, scalings) for which the arclength is invariant. They show that these equations locally behave as geometric heat equation $\frac{\partial C}{\partial t} = \frac{1}{g^2}\Delta L$, where g is the group-invariant metric $g = \frac{\partial r}{\partial x}$. If r is the (usual) Euclidean arclength we get the Euclidean shortening flow discussed above.

Name of flow	Luminance evolution	Curve evolution	Timestep N.N.	Timestep Gauss der.
Linear	$\frac{\partial L}{\partial t} = \Delta L$	$\frac{\partial C}{\partial t} = -\frac{\Delta L}{\lvert \hat{\nabla} L \rvert}$	$\frac{(\Delta x)^2}{2D}$	2 es
Variable conductance	$\frac{\partial L}{\partial t} = \hat{\nabla} \cdot (c\, \hat{\nabla} L)$	$\frac{\partial C}{\partial t} = -\frac{\hat{\nabla} \cdot (c\, \hat{\nabla} L)}{\lvert \hat{\nabla} L \rvert}$	$\frac{(\Delta x)^2}{2D}$	2 es
Normal or constant motion	$\frac{\partial L}{\partial t} = c L_w$	$\frac{\partial C}{\partial t} = c\, \hat{N}$	$\frac{\Delta x}{c}$	-
Euclidean shortening	$\frac{\partial L}{\partial t} = L_{vv}$	$\frac{\partial C}{\partial t} = \kappa \hat{N}$	$\frac{(\Delta x)^2}{2}$	2 es
Affine shortening	$\frac{\partial L}{\partial t} = L_{vv}^{\frac{1}{3}} L_w^{\frac{2}{3}}$	$\frac{\partial C}{\partial t} = \kappa^{\frac{1}{3}} \hat{N}$	-	-
Affine shortening modified	$\frac{\partial L}{\partial t} = \frac{\left[L_{vv}^{\frac{1}{3}} L_w^{\frac{2}{3}}\right]_{\sigma_1}}{\left(\frac{L_w}{k}\right)^{\frac{2}{3}}_{\sigma_2}}$	$\frac{\partial C}{\partial t} = \kappa^{\frac{1}{3}} \left(\frac{L_w}{k}\right)^{-\frac{2}{3}} \hat{N}$	-	-
Entropy	$\frac{\partial L}{\partial t} = \beta_0 L_w + \beta_1 L_{vv}$	$\frac{\partial C}{\partial t} = (\beta_0 + \beta_1 \kappa) \hat{N}$	-, $\frac{(\Delta x)^2}{2}$	-, 2 es

Figure 21.19 Table of some popular nonlinear diffusion equations with their name, PDE formula for the luminance evolution, the PDE formula for (isophote) curve evolution, the maximum allowed timestep for nearest neighbor implementations (N.N.), and the maximum allowed timestep for Gaussian derivative implementation.

When r_a is the *affine* arclength (invariant under affine transformations) they derive the curve evolution equation $\frac{\partial C}{\partial t} = \frac{\partial^2 C}{\partial r_a^2} = \kappa^{\frac{1}{3}} \vec{N}$, or equivalently $\frac{\partial L}{\partial t} = L_{vv}^{\frac{1}{3}} L_w^{\frac{2}{3}}$.

Table 21.19 summarizes the properties of a number of important nonlinear diffusion schemes [adapted from Niessen 1997d]:

Here are the *Mathematica* forward-Euler implementations of constant-motion (normal) flow, affine shortening flow, and a modified version of affine shortening flow (proposed by

Niessen [Niessen 1997c]) to slow down the evolution at high gradients so it maintains corners better:

```
constantmotion[im_, nrsteps_, σ_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im;
    Do[imt += δs √(gD[imt, 1, 0, σ]^2 + gD[imt, 0, 1, σ]^2 , {nrsteps}]; imt];

affineshortening[im_, nrsteps_, σ_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im; Do[
    imt += δs (gD[imt, 0, 2, σ] gD[imt, 1, 0, σ]^2 - 2 gD[imt, 0, 1, σ] gD[imt, 1, 0, σ]
        gD[imt, 1, 1, σ] + gD[imt, 0, 1, σ]^2 gD[imt, 2, 0, σ]) ^ (1 / 3)
      (gD[imt, 0, 1, σ]^2 + gD[imt, 1, 0, σ]^2)^(1/3), {nrsteps}]; imt];
```

The modified affine shortening flow applies a different scale σ_{grad} for the modifying gradient, weighted by a factor k:

```
affineshorteningmodified[im_, nrsteps_, σ_, σgrad_, k_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im;
    Do[imt += δs (gD[imt, 0, 2, σ] gD[imt, 1, 0, σ]^2 - 2 gD[imt, 0, 1, σ]
        gD[imt, 1, 0, σ] gD[imt, 1, 1, σ] + gD[imt, 0, 1, σ]^2 gD[imt, 2, 0, σ]) ^
      (1 / 3) (gD[imt, 0, 1, σ]^2 + gD[imt, 1, 0, σ]^2)^(1/3)
      (1/k (gD[imt, 0, 1, σgrad]^2 + gD[imt, 1, 0, σgrad]^2))^(-2/3) , {nrsteps}]; imt];
```

We use a test image of a square (intensities between 0 and 100) with added Gaussian noise (mean = 0, σ = 50).

```
<< Statistics`ContinuousDistributions`;
noisysquare = Table[If[35 < x < 93 && 30 < y < 93, 0, 100] +
    Random[NormalDistribution[0, 50]], {y, 128}, {x, 128}];
DisplayTogetherArray[ListDensityPlot /@ {noisysquare,
euclideanshortening[noisysquare, 32, 1, 32]}, ImageSize -> 315];
```

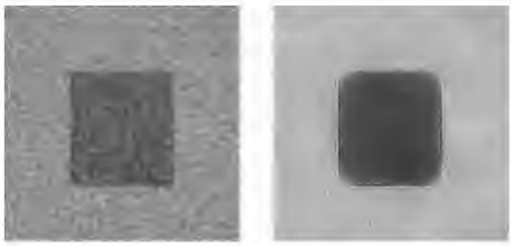


Figure 21.20 Euclidean shortening flow is a curvature-driven flow; in curve evolution terminology: curves are moved in the gradient direction with a speed proportional with the curvature. Clearly the corners with $\kappa >> 0$ are smoothed, while the straight ($\kappa = 0$) edges are preserved.

Why the name 'shortening flow'?

Sapiro and Tannenbaum [Sapiro1993e] proved that the length of the curve under a shortening flow indeed shrinks. They showed for the metric of the curve, defined as $g(p,t) = \left| \frac{\partial \vec{x}}{\partial p} \right|$, where p is an arbitrary parametrization of the curve $\vec{x}$, that the evolution of the metric is equal to: $\frac{\partial g}{\partial t} = -\kappa^2 g$.

The total length of the curve $L = \int_0^{2\pi} g(\tau, t)\, d\tau$ evolves as $\frac{\partial L}{\partial t} = \frac{\partial}{\partial t} \int_0^{2\pi} g(\tau, t)\, d\tau = -\int_0^{2\pi} \kappa^2\, g(\tau, t)\, d\tau = -\int_0^L \kappa^2\, dv$, from which we see that the length is always *decreasing* with time.

21.14 Mathematical Morphology

Mathematical morphology is one of the oldest image processing and analysis techniques. The original idea is the application of a *logical* area-operator (called *structuring element*) on areas of the image in the same way as convolution filters. The following example illustrates this. We define a simple binary image **lbin** and a structuring element **structel**:

```
lbin = PadRight[Table[1, {3}, {3}], {7, 7}, 0, 2];
structel = Table[1, {3}, {3}]; MatrixForm /@ {lbin, structel}
```

$$\left\{\begin{pmatrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{pmatrix}, \begin{pmatrix} 1&1&1\\ 1&1&1\\ 1&1&1 \end{pmatrix}\right\}$$

The structuring element is shifted over the image exactly as in a 2D convolution. The logical operation **BitOr** applied on all elements that are in both the structuring element and the underlying image patch (this is detected with **BitAnd**) give a dilation of the object in the image:

```
ListConvolve[structel, lbin, {2, 2}, 0, BitAnd, BitOr] // MatrixForm
```

$$\begin{pmatrix} 0&0&0&0&0&0&0\\ 0&1&1&1&1&1&0\\ 0&1&1&1&1&1&0\\ 0&1&1&1&1&1&0\\ 0&1&1&1&1&1&0\\ 0&1&1&1&1&1&0\\ 0&0&0&0&0&0&0 \end{pmatrix}$$

Reversing the operations gives an erosion:

```
ListCorrelate[1 - structel, lbin, {2, 2}, 1, BitOr, BitAnd] // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
Show[Import["erosions-dilations.gif"], ImageSize -> 350];
```

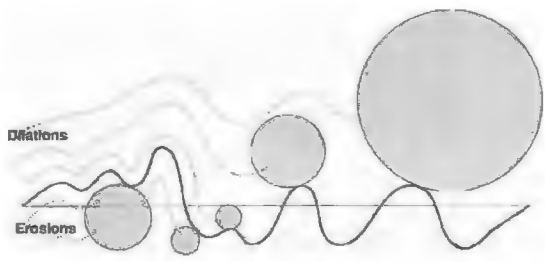


Figure 21.21 Erosion and dilation of a curve with a ball rolled over it at the outer and inner borders. The larger structuring element smoothes the curve more. The size of the ball is the scale of the smoothing process. From van den Boomgaard [vandenBoomgaard1993a].

The relation with normal flow now becomes clear: the motion of the contour of the image is governed by the structuring element in exactly the same way as the level set is moved in the direction of the normal.

This is only true for an isotropic convex (i.e. round) structuring element. One also says that the unit gradient vector $|\nabla L|$ is the infinitesimal generator for the normal motion evolution equation. Figure 21.21 shows the concept.

We define the following functions in *Mathematica* (for application on binary images only):

```
Unprotect[binarydilate, binaryerode];
binarydilate[im_, structel_] := ListConvolve[structel,
  im, Ceiling[Dimensions[structel] / 2], 0, BitAnd, BitOr];
binaryerode[im_, structel_] := ListCorrelate[1 - structel,
  im, Ceiling[Dimensions[structel] / 2], 1, BitOr, BitAnd];
```

The sequence of erosion followed by dilation removes small structures selectively from the image:

```
textim = Import["Text.gif"][[1, 1]];
im = textim; Block[{$DisplayFunction = Identity},
 p0 = ListDensityPlot[im];
 p1 = Table[ListDensityPlot[im = binaryerode[im, structel]], {i, 3}];
 p2 = Table[ListDensityPlot[im = binarydilate[im, structel]], {i, 3}]];
```

```
Show[GraphicsArray[{p0, p2[[3]]}], ImageSize -> 340];
```

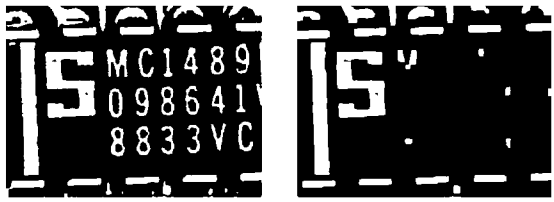


Figure 21.22 Left: original binary image. Right: result after three erosions followed by three dilations with a square 3x3 structuring element. The smaller structures have been eroded fully and did not return upon subsequent restoration of the contour by dilation.

Here are the intermediate steps:

```
Show[GraphicsArray[{p1, p2}], ImageSize -> 330];
```

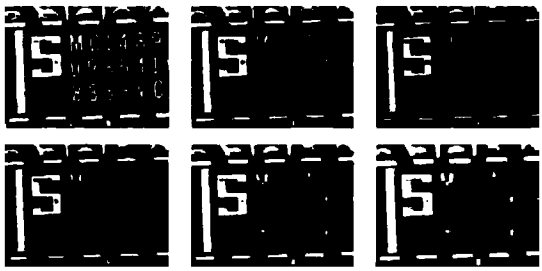


Figure 21.23 Top row: three consecutive erosions of the text image. Bottom row: Three consecutive dilations of the eroded image.

Subtracting the result of the operation of erosion (of the original image) from the result of the operation of dilation (of the original image) gives us the result of the *morphological gradient operator*:

```
ListDensityPlot[binarydilate[textim, structel] -
   binaryerode[textim, structel], ImageSize -> 180];
```

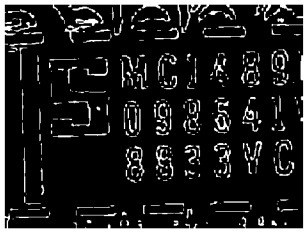


Figure 21.24 The subtraction of an eroded and dilated image gives the morphological gradient of the image. The scale of the operation is hidden in the size of the structuring element.

21.15 Mathematical morphology on grayvalued images

The classical way to change the binary operators from mathematical morphology into operators for gray-valued images, is to replace the binary operators by maximum/minimum operators.

Mathematical morphology is a mature and well documented branch of computer vision. We will limit ourselves here to the *Mathematica* implementation of the grayvalue-erosion and grayvalue dilation operators. Dilation employs the **Max** operator, erosion the **Min** operator.

```
Unprotect[dilate, erode]; dilate[im_, el_] :=
 ListConvolve[el, im, Ceiling[Dimensions[el] / 2], Min[im], Plus, Max];
erode[im_, el_] := ListCorrelate[-el, im,
  Ceiling[Dimensions[el] / 2], Max[im], Plus, Min];
```

Here is an example for a 256^2 image:

```
immr = Import["mr256.gif"][[1, 1]]; el = Table[1, {3}, {3}];
DisplayTogetherArray[
  {ListDensityPlot[immr], ListDensityPlot[imd = dilate[immr, el]],
   ListDensityPlot[ime = erode[immr, el]]}, ImageSize -> 510];
```

Figure 21.25 Left: original image. Middle: grayvalue dilated, right: grayvalue eroded version.

Structuring element = $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

And here is the morphological gradient for this image and structuring element:

```
ListDensityPlot[ime - imd, ImageSize -> 450];
```

Figure 21.26 Grayvalue morphological gradient . Structuring element as in fig. 21.25.

21.16 Mathematical morphology versus scale-space

It was shown by van den Boomgaard and Dorst that a parabolic structuring element leads to Gaussian blurring [van den Boomgaard 1997]. This establishes an elegant equivalence between mathematical morphology and Gaussian scale-space.

Florack, Maas and Niessen [Florack 1999a] related mathematical morphology and Gaussian scale-space by showing that both theories are cases from a more general formulation.

It can be shown that dilation or erosion with a ball is mathematically equivalent to *constant motion flow*, where the isophotes are considered as curves and they are moved in the gradient (or opposite) direction. Here is the *Mathematica* code and some examples:

```
constantmotion[im_, nrsteps_, σ_, evolutionrange_] :=
  Module[{δs, imt}, δs = evolutionrange / nrsteps; imt = im;
   Do[imt += δs √(gD[imt, 1, 0, σ]^2 + gD[imt, 0, 1, σ]^2), {nrsteps}]; imt];

DisplayTogetherArray[
  ListDensityPlot[constantmotion[textim, 10, 1, 10]],
  ListDensityPlot[constantmotion[textim, 10, 1, -10]], ImageSize -> 440];
```

Figure 21.27 Grayvalue morphological dilation and erosion are equivalent to non-linear diffusion with *constant motion flow* (with the PDE $\frac{\partial L}{\partial s} = \pm |\vec{\nabla} L|$). The expansion of the contour by the morphological structuring element is equlvalent to the curve motion of the isophote in the gradient direction. Left: constant motion dilation. Right: constant motion erosion.

21.17 Summary of this chapter

The diffusion can be made locally adaptive to image structure. Three mathematical approaches are discussed:

1. PDE-based nonlinear diffusion, where the luminance function evolves as the divergence of some flow. The nonlinear PDE's involve local image derivatives, and cannot be solved analytically;
2. Evolution of the isophotes as an example of curve-evolution;
3. Variational methods, minimizing an energy functional defined on the image.

Adaptive smoothing requires geometric reasoning to define the influence on the diffusivity coefficient. The simplest equation is the equation proposed by Perona & Malik, where the variable conduction is a function of the local edge strength. Strong gradient magnitudes prevent the blurring locally, the effect is edge preserving smoothing. The strong feedback connections seen from V1 to LGN may be involved (among many other possible mechanisms) in a locally adaptive scheme for image enhancement.

The numerical implementation of the nonlinear PDE's is examplified with an iterative forward-Euler scheme. This is the simplest scheme to start with, but is unstable for stepsizes larger than the Von Neumann criterion. We derive this criterion again and expand it for scaled (regularized) differential operators. The stepsize can then be made substantially larger.

The Perona & Malik equation leads to deblurring (enhancing edges) for edges larger than the turnover point k, and blurs smaller edges. This is one of the reasons why the performance of this PDE is so appreciated.

There is a strong analogy between curve evolution and PDE based schemes. They can be related directly to one another.

Euclidean shortening flow involves the diffusion to be limited to the direction perpendicular to the gradient only. The divergence of the flow in the equation is equal to the second order gauge derivative L_{vv} with respect to v, the direction tangential to the isophote. Normal motion flow is equivalent to the mathematical morphological erosion or dilation with a ball. The dilation and erosion operators are shown to be convolution operators with boolean operations on the operands.

22. Epilog

Computer vision is a huge field, and this book could only touch upon a small section of it. First of all, the emphasis has been on the introduction of the notion of *observing* the physical phenomena, which makes the incorporation of scale unavoidable. Secondly, scale-space theory nicely starts from an axiomatic basis, and incorporates the full mathematical toolbox. It has become a mature branch in modern computer vision research.

A third major notion is the *regularization* property of multi-scale operators, in particular spatio-temporal differential operators. This enables the use of powerful differential geometric methods on the discrete data of computer vision, for the many fields where differential structure is part of the analysis, such as shape, texture, motion etc.

A next major emphasis has been the inspiration from the workings of the human visual system. Here too is still much to be learned, especially in the layers following the visual front-end, where perceptual grouping and recognition is performed. The intricate feedback loops and the local orientation column linking should be among the first to be studied in detail. Modern neuro-imaging technology is about to give many clues.

The theory covered in this book, focusing on *bio-mimicking* front-end vision, is primarily applied to the *local* analysis of image structure. The extraction of global, intelligently connected structure is a widely explored area, where model-based and statistical methods prevail to arrive to good perceptual grouping of local image structure. The study of the multi-scale relations in the deep structure in scale-space, and the use of hierarchical, more topological multi-scale methods, has just only started.

Finally, computer vision is solidly based on mathematics, in any applicable field. Image processing has become a *science*.

This book showed the use of *Mathematica*, as a powerful combination of a complete high-level mathematical programming language, from which through *pattern matching* the numerical implementation can be automatically generated, enabling rapid prototyping for virtually all the concepts discussed in this book. Many of the functions are intrinsically n-dimensional.

The role of and need for robust computer vision techniques is ever increasing. In diagnostic radiology, computer-aided diagnosis will see great successes in the next decade, and the availability of huge image databanks will stimulate the study to image guided retrieval and self-organization of analysis systems.

Much has been left untreated, the main reason is that the field is so huge. A possible sequel of this book might include multi-scale methods for shape from shading, texture analysis

(locally orderless images), 3D differential geometry, wavelet based analysis, nonlinear and statistical methods, and deep structure analysis.

This book is meant to be a *interactive* tutorial in multi-scale image analysis, describing the basic functionality. It is my sincere wish that this book has invited you to actively explore this fascinating area.

Eindhoven, Summer 2002.

```
<< FrontEndVision`FEV`;
Show[Import["ScaleSpaceForest2.jpg"]];
```

Figure 22.1 Artist's impression of a 'deep structure scale-space' forest, where the geodesic paths of the extrema and saddlepoints are shown as brown branches, and the points of annihilation of extrema and saddlepoints are depicted as green balls. Scale runs vertical. The mist is a partly transparent surface where the determinant of the Hessian vanishes. Artist: ir. Frans Kanters, University of Technology Eindhoven, Department of Biomedical Engineering, Biomedical Image Analysis Group, 2003.

A. Introduction to *Mathematica*

After finding the next-to-last bug, clean up your debugging stuff. The last bug in any piece of code is invariably found by the first user of the code and never by the programmer.
Roman Maeder, Programming in Mathematica I, pp. 43.

Mathematica is an fully integrated environment for technical computing. It has been developed by prof. Stephen Wolfram and is now being developed and distributed by Wolfram Research Inc. *Mathematica* comes with an excellent on-board help facility, which includes the full text of the handbook (over 1400 pages).

Mathematica used to be slow and memory-intensive. This might be the reason why so many computer vision labs have not considered applying it to images. It is a pleasant 'discovery' that *Mathematica* is now *fast* and efficient with memory.

As a concise quick reference guide, here are the most important things to know when you want to get started with *Mathematica*:

A.1 Quick overview of using Mathematica

Mathematica consists essentially of two programs, that communicate with each other: the *kernel* for the calculations and the *front-end* for the graphical user-interface (input and presentation of data). The front-end is an advanced text processor. It creates *notebooks*, portable documents that consists of *cells*, indicated by the brackets to the right of the cell. The front-end and kernel together form an interactive interpreter.

Cells come in many styles, like 'title', 'text', 'input' etc. The front-end takes care of automatic grouping of the cells (this, as any function, can be turned off).

By double-clicking on the group bracket a group of cells can be closed or opened, for quick overview of the document.

Input style cells are sent to the kernel and executed by pressing 'shift-return'. Commands can extend over several lines in an input type cell, separated by returns. Cells can be edited, and run again. *Mathematica* remembers activities in the order of processing, not by the location of the cell in the notebook.

The format menu item contains all commands for formatting text. The style sheets set the definitions for appearances of elements for each cell type. We initialize every chapter with the following commands. The first sets paths and options for often used functions, the second contains the functions predefined for the book:

```
<< FrontEndVision`FEV`;
```

Mathematica can do both symbolic and numerical calculations. It has arbitrary precision. **N[π,100]** gives the numerical value of π in 100 significant digits, **D[Log[x],x]** gives the first derivative of the natural logarithm of x.

```
N[π, 100]
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781
 64062862089986280348253421170 68
```

```
D[Log[x], x]
```

$$\frac{1}{x}$$

Use of brackets in *Mathematica*:

{ }	List	Example: **{x,y,z}**
[ ]	Function	Example: **HermiteH[n,x]**
( )	Combine	Example: $(x+3)^2$
(*...*)	Comment	Example: **(* This function .. *)**

The internal structure of *every* expression is a **Head** in front of a list of operands. Check the internal representation with **FullForm**:

```
FullForm[{1, 2, a b, c^2, {p, q}}]
```

```
List[1, 2, Times[a, b], Power[c, 2], List[p, q]]
```

Mathematica is strongly list oriented. Lists can be nested in any order. Every expression is a list, as are our image data. Most commands are optimized for list operations.

Notation:
- Multiplication is indicated with a space or *****.
- A semicolon **;** at the end of a command means "Print no output". Useful when a lot of textual output is expected. The semicolon **;** is also the regular expression statement separator.
- Enter Greek letters with the escape key ⎋ before and after it: E.g.: ⎋ p ⎋ turns into π. Any symbol can be entered through 'palettes' (see the File menu on the title bar of your *Mathematica* session).
- Enter superscript with `control-^`, subscript with `control-_`, division line with `control-/`, a square-root sign with `control-2`.
- *Mathematica*'s internal variables and functions all begin with a capitalized letter, your own defined variables should always begin with a lower case for clear distinction: **Pi**, **Plot3D[]**, **myfunction[]**, **{x,y,z}**.
- Often we use the 'postfix' form for the application of a function: `2 ^ 100 // N` is equivalent to $N[2^{100}]$.

```
2 ^ 100 // N
```

$$1.26765 \times 10^{30}$$

Some front-end tips:
- The menu item Format - Show Toolbar gives a handy toolbar below the title bar of your window.

- Keeping the alt-key press while dragging the mouse gives smooth window scrolling.
- **Help** is available on any command.

The full 1400 page manual is online under the Help menu item.

There is a very useful 'getting started' section, and a 'tour of *Mathematica*'. Shortcut for help: Highlight the text and press F1.
- Command completion is done with control-k, the list of arguments of a function with shift-control-k.
- The notebook can be executed completely with the Kernel menu commands.
- All output can be deleted, which may save disk space considerably.
- A series of Graphics output can be animated by double-clicking one of the figures. The bottombar of the window show steering controls for the animation.
- The input menus contains many interesting features, as 3D viewpoint selection, color selection, hyperlinks, sound, tables and matrices, automatic numbering objects etc. and is worth studying the features.

A.2 Quick overview of the most useful commands

The commands below occur often in this book. Full explanation and many examples are given in the Help browser of *Mathematica*. For the sake of the readers that do not have *Mathematica* running, available or at hand while reading this book, some short examples are given of the actions of the commands.

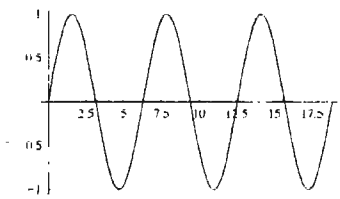
Plot commands come with many options. See available options in the Help browser or with e.g. **Options[Plot3D]**.

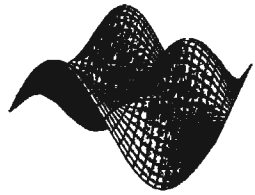
```
Plot, Plot3D, ContourPlot, ListPlot, ListDensityPlot
```

Mathematica shows every plot it creates immediately. The output of the plot commands is controlled by the option DisplayFunction. It specifies the function to apply for displaying graphics (or sound).

To prevent intermediate results, e.g. while preparing a series of plots to be shown with **GraphicsArray**, a useful construct is to create a scoping construct with **Block[{vars} ...]**. Within a block, all variables **vars** are hidden from the main global context. E.g. in the following example the setting for **$DisplayFunction** is temporarily set in the block context to **Identity**, which means: no output.

```
Block[{$DisplayFunction = Identity},
  p1 = Plot[Sin[x], {x, 0, 6 π}];
  p2 = Plot3D[Sin[x] Cos[y], {x, 0, 2 π}, {y, 0, 2 π}]];
Show[GraphicsArray[{p1, p2}]];
```





Mathematica is **List** oriented. This is a short nested **List**:

```
ma = {{a, b, c}, {d, e, f}, {g, h, i}}
```

```
{{a, b, c}, {d, e, f}, {g, h, i}}
```

FullForm gives the internal representation, i.e. a **Head** with a series of operands:

```
FullForm[ma]
```

```
List[List[a, b, c], List[d, e, f], List[g, h, i]]
```

Listable is an attribute of many functions. It means that they perform their action on all elements of a list:

```
ma + 1
```

```
{{1 + a, 1 + b, 1 + c}, {1 + d, 1 + e, 1 + f}, {1 + g, 1 + h, 1 + i}}
```

```
ma^2
```

```
{{a^2, b^2, c^2}, {d^2, e^2, f^2}, {g^2, h^2, i^2}}
```

Clear[x] or **f[x]=.** clears **f[x]**. **Remove[f]** completely removes the symbol **f**.

```
f =.; f^3
```

```
f^3
```

Nest applies a function multiple times.

```
f =.; Nest[f, x, 3]
```

```
f[f[f[x]]]
```

With repeated operations (using **Nest**) a wide variety of self-similar structure can be generated. From Stephen Wolfram's new book [Wolfram2002] the gasket fractal:

```
Nest[SubsuperscriptBox[#, #, #] &, "Ω", 5] // DisplayForm
```

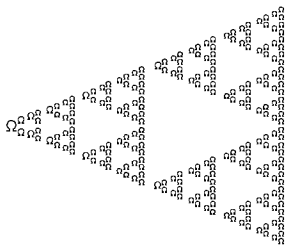


Figure A.1 The gasket fractal is created by repeated action, which is implemented with the function **Nest**. See also mathforum.org/advanced/robertd/typefrac.html.

Map maps a function on the elements of a list:

```
Map[f, ma]
```

```
{f[{a, b, c}], f[{d, e, f}], f[{g, h, i}]}
```

You can specify the level in the list where the function should be mapped:

```
Map[f, ma, 2]
```

```
{f[{f[a], f[b], f[c]}], f[{f[d], f[e], f[f]}], f[{f[g], f[h], f[i]}]}
```

```
Map[f, ma, {2}]
```

```
{{f[a], f[b], f[c]}, {f[d], f[e], f[f]}, {f[g], f[h], f[i]}}
```

Apply replaces the head of an expression with a new head. This sums the columns:

```
Apply[Plus, ma]
```

```
{a + d + g, b + e + h, c + f + i}
```

This sums the rows, i.e. **Plus** is applied at level 1 in the **List**:

```
Apply[Plus, ma, {1}]
```

```
{a + b + c, d + e + f, g + h + i}
```

Apply can even replace at level 0, i.e. the head itself. This sums all elements of a matrix:

```
Apply[Plus, ma, {0, 1}]
```

```
a + b + c + d + e + f + g + h + i
```

Some often used commands have short notations:

Apply	**@@**
Map	**/@**

Replace /.
Condition /;

Postfix //

```
Times @@ ma /. {c -> c^2}
```

```
{a d g, b e h, c^2 f i}
```

In the following lines, the first statement rotates (cyclic) the elements one position to the right, while the second rotates one position downward (rowshift down).

```
RotateRight /@ ma
```

```
{{c, a, b}, {f, d, e}, {i, g, h}}
```

```
RotateRight[ma]
```

```
{{g, h, i}, {a, b, c}, {d, e, f}}
```

```
Plus @@ ma
```

```
{a + d + g, b + e + h, c + f + i}
```

```
ma // MatrixForm
```

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

To execute a function on every dimension of a multidimensional array (e.g. 2D the columns and the rows), use the function **Map[f,data,level]** with indication of the level of mapping. Here is an example for an operation on a 3D array, for the z, y and x direction:

```
Clear[a, b, c, d, e, f, g]; m2 = {{{a, b}, {c, d}}, {{e, f}, {g, h}}};
Map[RotateRight, m2, {0}]
Map[RotateRight, m2, {1}]
Map[RotateRight, m2, {2}]
```

```
{{{e, f}, {g, h}}, {{a, b}, {c, d}}}
```

```
{{{c, d}, {a, b}}, {{g, h}, {e, f}}}
```

```
{{{b, a}, {d, c}}, {{f, e}, {h, g}}}
```

Table generates lists of any dimension, e.g. vectors, matrices, tensors:

```
rt = Table[Random[], {i, 1, 3}, {j, 1, 4}]; rt // MatrixForm
```

$$\begin{pmatrix} 0.419066 & 0.685606 & 0.696535 & 0.470892 \\ 0.898048 & 0.0157444 & 0.0700894 & 0.44903 \\ 0.0873124 & 0.818922 & 0.833205 & 0.0487981 \end{pmatrix}$$

A.3 Pure functions

An operator is a 'pure function', e.g. `(#^2) &` is a function without name, where some operation on the operand # is performed, in this case squaring the variable. So `(#^2) &` means: 'square the argument'. Multiple variables are indicated with #1, #2 etc.

```
(#^2) & [f]
```

f^2

```
p = (Sin[#1 + #2] + Sqrt[#2]) &;
p[f, g]
```

$\sqrt{g} + \text{Sin}[f + g]$

We use a pure function frequently when we have to apply a function repeatedly to different arguments:

```
f = windingnumber[#] &;
f /@ {1, 4, 6}
```

```
{windingnumber[1], windingnumber[4], windingnumber[6]}
```

```
f[4]
```

```
windingnumber[4]
```

or when we want to plot something for a particular range of scales: (the pure function is now '`ListDensityPlot` the output of `gD` at some scale for the scales 2, 3 and 6'):

```
im = Import["mr256.gif"][[1, 1]];
DisplayTogetherArray[
  ListDensityPlot[gD[im, 1, 0, #]] & /@ {2, 3, 6}, ImageSize → 410];
```

Figure A.2 Use of the function `Map (/@)`.

A.4 Pattern matching

To show the usefulness of pure functions and the technique of pattern matching we work out an example in some more detail: We do manipulations on words of a complete English dictionary, consisting of 118617 English words.

We find palindromes (words that are the same when written in reverse order) and word length statistics. The example is taken from one of the Wolfram *Mathematica* tutorials, see library.wolfram.com.

We read the data with **ReadList** and check its size with **Dimension**. Note that we put a ; (semicolon) at the end of the next cell. This prevents the full text of the dictionary to be printed as output to the screen, which is in this case a very useful feature!

```
data = ReadList["dictionary118617.txt", String];
Dimensions[data]
```

```
{118617}
```

These are the first 20 elements:

```
Take[data, 20]
```

```
{aardvark, aardvarks, aaronic, abaca, abaci, aback, abacterial, abacus,
 abacuses, abaft, abalienate, abalienated, abalienating, abalienation,
 abalone, abalones, abandon, abandoned, abandonedly, abandonee}
```

We use the commands for counting the number of letters in a string, and to reverse a string:

```
StringLength["FrontEndVision"]
```

```
14
```

```
StringReverse["FrontEndVision"]
```

```
noisiVdnEtnorF
```

The following command selects those elements which are equal to its reverse, and are longer then 2 letters. **&&** denotes the logical **And**. Note the use of the pure function:

```
Select[data, (# == StringReverse[#] && StringLength[#] > 2) &]
```

```
{adinida, aha, ama, ana, anna, bib, bob, boob, civic, dad, deed, deified,
 deled, did, dud, eke, ene, ere, ese, esse, eve, ewe, eye, gag, gig, hah,
 huh, kayak, kook, level, madam, malayalam, minim, mom, mum, nisin,
 non, noon, nun, pap, peep, pep, pip, poop, pop, pup, radar, redder,
 refer, reviver, rotator, rotor, sagas, sees, sexes, shahs, sis,
 solos, sos, stats, succus, suus, tat, tenet, tit, tnt, toot, tot, wow}
```

To calculate the length *of each* word, we *map* **StringLength** on **data**:

```
wordLengths = Map[StringLength, data];
Max[wordLengths]
```

```
45
```

Count[list, pattern] gives the number of elements in list that match pattern.

```
Count[wordLengths, 10]
```

```
14888
```

```
t = Table[Count[wordLengths, i], {i, 1, Max[wordLengths]}]
```

```
{0, 93, 754, 3027, 6110, 10083, 14424, 16624, 16551, 14888, 12008,
 8873, 6113, 3820, 2323, 1235, 707, 413, 245, 135, 84, 50, 23,
 16, 9, 4, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2}
```

```
ListPlot[t, PlotStyle → PointSize[0.02],
  AxesLabel -> {"wordlength", "occurrence"},
  PlotJoined -> False, ImageSize -> 230];
```

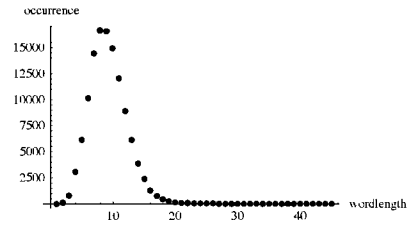


Figure A.3 Histogram of word lengths in a large English dictionary.

Pattern matching is one of the most powerful techniques in *Mathematica*. There are three symbols for a pattern: _ denotes anything which is a single element, __ denotes anything which is one or more elements, ___ denotes anything which is zero, one or more elements. **x_** denotes a pattern which is known under the name x. **Replacement** (**/.**) is by **Rule** (→). The following statement replaces every occurrence of a into $\sqrt{a}$:

```
f = .; ma /. {a -> √a}
```

```
{{√a, b, c}, {d, e, f}, {g, h, i}}
```

This returns the positions in the dictionary where words are found of more then 23 letters:

```
Position[data, x_ /; StringLength[x] > 23]
```

```
{{5028}, {17833}, {33114}, {33134}, {35841}, {49683}, {50319},
 {57204}, {57205}, {60016}, {62133}, {62598}, {62599}, {63552},
 {63723}, {63724}, {63725}, {67140}, {67141}, {70656}, {74099},
 {74101}, {74103}, {74166}, {79229}, {79273}, {79274}, {79500},
 {83241}, {104039}, {105810}, {106774}, {113411}, {114511}}
```

This returns the first pair of two consecutive 13-letter words in the dictionary:

```
Dimensions[data]
```

```
{118617}
```

```
data /. {a___, b_ /; StringLength[b] == 13,
   c_ /; StringLength[c] == 13, d___} -> {b, c}
```

```
{abstentionism, abstentionist}
```

A.5 Some special plot forms

ParametricPlot3D[{fx,fy,fz},{t,tmin,tmax},{u,umin,umax}] creates a surface, rather than a curve.

The surface is formed from a collection of quadrilaterals. The corners of the quadrilaterals have coordinates corresponding to the values of the f_i when t and u take on values in a regular grid.

```
r[u_] := -1 + E^(u/(6 π)); x = 2 r[u] Cos[u] Cos[v/2]^2;
y = 2 r[u] Cos[v/2]^2 Sin[u]; z = -1/Sqrt[2] r[2 u] + r[u] Sin[v];
shell = ParametricPlot3D[{x, y, z}, {u, 0, 6 Pi}, {v, 0, 2 Pi},
   PlotPoints → {100, 40}, PlotRange → All, Axes → False, Boxed → False,
   ViewPoint -> {2.581, 1.657, 0.713}, ImageSize -> 200];
```

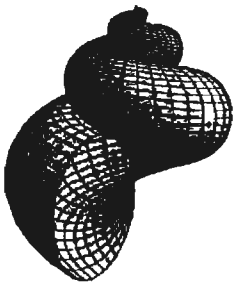


Figure A.4 An example of `ParametricPlot3D` for the plotting of more complicated 3D manifolds.

```
Clear[x, y, z]; << FrontEndVision`ImplicitPlot3D`;
torus = ImplicitPlot3D[z^2 == 1 - (2 - Sqrt[x^2 + y^2])^2,
   {x, -3, 3}, {y, -3, 3}, {z, -1, 1}, PlotPoints → {15, 15, 10},
   Passes → Automatic, ImageSize -> 200];
```

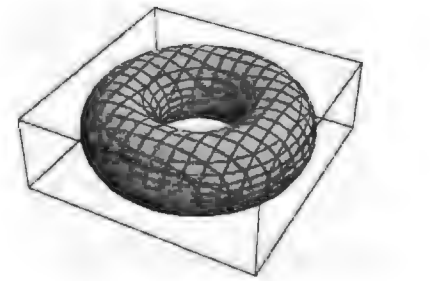


Figure A.5 An example of `ImplicitPlot3D` for the plotting of more complicated 3D manifolds.

A.6 A faster way to read binary 3D data

Mathematica has a set of utilities to read and write binary data from and to files. Examples of such commands are **`ReadBinary[...]`** and **`WriteBinary[...]`**. They are available in the package **`Utilities`BinaryFiles`** (see the help browser). These commands however are slow.

A faster way is to use an external C program to read the data, and to communicate with this program with *MathLink*. The program binary.exe (for Windows) is an executable C-program that contains all commands of the package **`BinaryFiles`** in a fast version. This executable is available from *MathSource* at the URL:

www.mathsource.com/Content/Enhancements/MathLink/0206-783.

Here also versions for other platforms are available. It is beyond the scope of this book to explain *MathLink*, but in the help browser and at the *MathSource* repository good manuals are available. The package is installed by **`Install`**:

```
Install["binary.exe"];
```

The taskbar in Windows at the bottom of the screen should now display the active program binary.exe with which we now will communicate.

Let us read a file in raw bytes dataformat with a 3D MRI dataset. It is given that the set **`mri_01.bin`** contains 166 slices with 146 rows (x-dimension) and 168 columns (y-dimensions). Each pixel is stored as an unsigned byte. To read this file we first need to open it:

```
channel = OpenReadBinary[
   $FEVDirectory <> "Images\mri02.bin", FilenameConversion → Identity];
```

It is fastest to read the binary 3D image stack slice by slice. We first define space to store the image, then we read 166 slices as bytes. Each slice is partitioned into 146 rows. At the end we close the file.

```
im = Table[{}, {166}];
Table[
  im[[i]] = Partition[ReadListBinary[channel, Byte, 168 146], 146];,
  {i, 1, 166}];
Close[channel];
```

We check for the dimensions of our 3D image, about 4 million pixels:

```
Dimensions[im]
```

```
{166, 168, 146}
```

By calculating the **Transpose** of the 3D image, we can interchange the coordinates. The second argument is the new ordering of the coordinates. In this way it is easy to plot the other perpendicular planes. Let us look at the 85th image of the original stack, and the 85th image of two transposed forms respectively, as shown in the statement below. As we see from the left figure, the original slices were acquired in the *coronal* plane. The transposed images show us the *sagittal* plane (middle, Latin 'sagitta' = arrow) and the *transversal* plane (right). This method of transposing the data so we get other perpendicular planes is called *multiplanar reformatting*.

```
DisplayTogetherArray[
  ListDensityPlot /@ {im[[85]], Transpose[im, {3, 2, 1}][[85]],
    Transpose[im, {3, 1, 2}][[85]]}, ImageSize → 500];
```

Figure A.6 *Multiplanar reformatting* is the visualization of perpendicular images in a 3D dataset by transposing the 3D dataset. The left image is one of the original acquisitions in the coronal plane. The middle image shows the 85th image in the sagittal plane. It is formed by showing all rows of pixels perpendicular to the pixels in the 85th column in the left image. The right image shows the 85th image in the transversal plane. It is formed by showing all rows of pixels perpendicular to the pixels in the 85th row in the left image.

Multiplanar reformatting is of course only of high quality if the voxels are isotropic. Note in this example that slight differences in overall intensities in the original 3D MRI acquisition show up as vertical lines.

By manipulation of the pointer StreamPosition we can read an arbitrary slice from the 3D dataset. This pointer points at the position just preceding the next bytes to read. After opening the file, the streamposition is set to zero, which is at the beginning of the data. By

setting the stream position pointer 84 images further (84*168*146 locations further from zero), the next statement reads the 85th image only:

```
channel = OpenReadBinary[
   $FEVDirectory <> "Images\mri02.bin", FilenameConversion → Identity];
SetStreamPosition[channel, 84 * 168 * 146];
im85 = Partition[ReadListBinary[channel, Byte, 168 146], 146];
ListDensityPlot[im85, ImageSize -> 150];
```

Figure A.7 Direct read of a single slice from a 3D dataset is best done by manipulation of the stream position pointer.

```
Close[channel];
```

A.7 What often goes wrong

In this section we give a random set of traps in which you may easily fall if not warned:

A7.1 Repeated definition

When a function from a package is called before the package is actually read into the kernel, *Mathematica* adds the name to its global list as soon as it appears:

```
Remove[Histogram]

Histogram

Histogram
```

The function does not work, because the package **Graphics`Graphics`** has not been read.

When subsequently the package is read, *Mathematica* complains that it may overwrite a previous definition, and does not redefine the function **Histogram**. The natural way out is to **Remove** the first definition, and to read the package again.

```
<< Graphics`Graphics`;

Histogram::shdw :
 Symbol Histogram appears in multiple contexts {Graphics`Graphics`, Global`};
   definitions in context Graphics`Graphics` may
   shadow or be shadowed by other definitions.

Remove[Histogram];
<< Graphics`Graphics`;
```

Now the definition is fine:

```
? Histogram

Histogram[{x1, x2, ...}] generates a bar graph representing a histogram of the
   univariate data {x1, x2, ...}.  The width of each bar is proportional
   to the  width of the interval defining the respective category, and
   the area of the bar  is proportional to the frequency with which the
   data fall in that category. Histogram range and categories may be
   specified using the options HistogramRange and HistogramCategories.
   Histogram[{f1, f2, ...}, FrequencyData -> True] generates a histogram
   of the univariate frequency data {f1, f2, ...}, where fi is the
   frequency with which the original data fall in category i. More...

Histogram[{x1, x2, ...}] generates a bar graph representing
   a histogram of the univariate data {x1, x2, ...}. More...
```

A7.2 Endless numerical output

Prevent accidental output to the notebook if not necessary and very long, eg. when an image is calculated.

Any output is not printed when the statement is concluded with a semicolon. The first statement generates about *a million random numbers to the screen*, which will take a very long time to generate and prevent you from continuing (luckily, we made the cell inevaluatable). The second statement with the semicolon is fine.

```
m = Table[Random[], {1000}, {1000}]

m = Table[Random[], {1000}, {1000}];
```

Use Alt-. to abort an unwanted evaluation.

A7.3 For speed: make data numerical when possible

Be careful with symbolic computations on larger datasets. You may only be interested in the numerical result. Compare the examples below:

```
mm1 = Table[Sin[x y], {y, 1, 8}, {x, 1, 8}];
Timing[symbolicInverse = Inverse[mm1];]

{36.343 Second, Null}
```

Even for this small matrix, each symbolic term is huge, and very impractical to handle. The numerical result is very fast:

```
Timing[numericalInverse = Inverse[N[mm1]];]

{0. Second, Null}
```

We look at the result:

```
Short[numericalInverse, 4]

{{0.150096, -0.00900765, 0.16249, -0.295079,
  -0.194507, -0.0870855, 0.139714, 0.336583}, <<6>>, {<<1>>}}
```

Another example: the numerical Eigenvalues of a matrix with 10,000 elements is computed fast:

```
mm2 = Table[Random[], {100}, {100}];
Timing[Eigenvalues[mm2];]

{0.032 Second, Null}
```

And use functional programming and internal functions as much as possible.

```
Timing[array = Range[10^7];]

{0.031 Second, Null}

Timing[array = Table[i, {i, 1, 10^7}];]

{5.906 Second, Null}
```

A7.4 No more loops and indexing

E.g. to multiply each 2 elements of an array, from head to tail:

```
kl = m = Table[Random[], {i, 1, 10^6}];
Timing[For[i = 0, i < 10^6, kl[[i]] = m[[i]] m[[10^6 - i + 1]], i++]] // First

18.719 Second
```

The same result is acquired much faster if we use mathematical programming with native *Mathematica* functions. They are optimized for speed, and programming becomes much more elegant.

```
Timing[k = m Reverse[m]] // First

0.406 Second
```

A7.5 Copy and paste in InputForm

There are 4 format types for cells in *Mathematica*. **InputForm**, **OutputForm**, **StandardForm** and **TraditionalForm**. See the help browser for a description of these types. Be alert when pasting a **TraditionalForm** cell as input cell, *Mathematica* may be not able to interpret this unequivocally. When you attempt this, *Mathematica* will issue a warning, and you see the wigged line in the cell bracket.

A.8 Suggested reading

A number of excellent books are available on *Mathematica*. A few of the best are listed here:

[Blachman1999] N. Blachman. *Mathematica*: A practical approach. Prentice Hall, 2nd edition, 1999. ISBN 0-13-259201-0.
This complete tutorial is the easiest, quickest way for professionals to learn Mathematica, the world's leading mathematical problem-solving software. The book introduces the basics of Mathematica and shows readers how to get around in the program. It walks readers through all of Mathematica's practical, built-in numerical functions--and covers symbolic capabilities, plotting, visualization, and analysis.

[Ruskeepää1999] H. Ruskeepää. *Mathematica* Navigator: Graphics and methods of applied mathematics. Academic press, London. 1999. ISBN 0-12-603640-3 (paperback+CD-ROM).

Mathematica Navigator gives you a general introduction to the use of Mathematica, with emphasis on graphics, methods of applied mathematics, and programming.

The book serves both as a tutorial and as a handbook. No previous experience with Mathematica is assumed, but the book contains also advanced material and material not easily found elsewhere. Valuable for both beginners and experienced users, it is a great source of examples of code.

From the author:
- I would like first to ask, what is the general nature of your book Mathematica Navigator?
- Before answering, I would like to ask you, whether you know what is the difference between an applied mathematician and a pure mathematician?
- Hm..., I seem to remember having heard some differences, but no, I don't remember any at this moment. Tell me.
- An applied mathematician has a solution for every problem while a pure mathematician has a problem for every solution.
- Yes, indeed. That is a very describing difference. But how does this maxim relate with my question about the nature of your book?
- I am an applied mathematician, and so I took the task of solving every problem - namely in using Mathematica.
- Not a very modest goal...
- Frankly, I took the task to write as useful a guide as possible so that you would have a more easy way to the wonderful world of Mathematica.
- Do you start with the basics?
- Yes, and then the book goes carefully through the main material of Mathematica.
- What are the main areas of Mathematica?
- Graphics, symbolic calculation, numerical calculation, and programming.
- And how far does your book go?
- The book contains some advanced topics and material not easily found elsewhere, such as stereographic figures, graphics for four-dimensional functions, graphics of real-life data, fractal images, constrained nonlinear optimization, boundary value problems, nonlinear

difference equations, bifurcation diagrams, partial differential equations, probability, simulating stochastic processes, statistics.

And for many subjects we also write our own programs, to practice programming.
- Do you emphasize symbolic or numerical methods?

- Both are important. For a given problem, we usually first try symbolic methods, and if they fail, then we resort to numerical methods. Thus, for each topic, the book presents first symbolic methods and then numerical methods. The book gives numerical methods a special emphasis.
- Have you excluded some topics?
- Topics of a "pure" nature such as number theory, finite fields, quaternions, or graph theory are not considered. Commands for manipulating strings, boxes, and notebooks are covered only briefly. MathLink is left out (MathLink is a part of Mathematica enabling interaction between Mathematica and external programs).
- Do you like to say something about the writing of the book?
- The writing was simply exciting. It is one of the most interesting epochs of my life thus far. By writing the book I learned a lot about Mathematica and obtained a comprehensive view of it. And the more I learned about Mathematica, the more I admired it.
- What are the fine aspects of Mathematica?
- It is consistent, reliable, and comprehensive. In addition, Mathematica has very powerful commands, produces excellent graphics, and has a wonderful interface. However, it certainly takes some time to get used to Mathematica, but that time is interesting and rewarding, and then you have a powerful tool at your disposal.
- Thank you very much for this interview.
- Thank you.

[Wolfram1999] S. Wolfram. The *Mathematica* book. Fourth edition, Wolfram Media / Cambridge University Press, 1999. 1470 pages. ISBN 0-52-164314-7.

The definite reference guide for *Mathematica*, written by the author of *Mathematica*, Stephen Wolfram. Not a tutorial, but a handbook. The full text of this book is available in the indexed help-browser of *Mathematica*, as well as a searchable document on the web: documents.wolfram.com/v4/index3.html.

[Mäder1996a] R. Mäder, Programming in *Mathematica,* 3rd ed.. Addison-Wesley Pub., 1996.

This revised and expanded edition of the standard reference on programming in *Mathematica* addresses all the new features in the latest versions 3 and 4 of *Mathematica.*

The support for developing larger applications has been improved, and the book now discusses the software engineering issues related to writing and using larger programs in Mathematica. As before, Roman Mäder, one of the original authors of the Mathematica system, explains how to take advantage of its powerful built-in programming language. It includes many new programming techniques which will be indispensable for anyone interested in high level Mathematica programming.

[Mäder1996b] R. Mäder, The Mathematica programmer II. Academic Press, 1996. 296 pages. ISBN 0-12-464992-0 (paperback). This book, which includes a CD-ROM, is a second volume to follow The Mathematica Programmer (now out of print) and includes many new programming techniques which will be indispensable for anyone interested in high level Mathematica programming.

A9. Web resources

Webpages are very dynamic, and it is impossible to give a complete overview here. Some stable pointers to a wealth of information and support are:

www.wolfram.com: The official homepage of Wolfram Inc., the maker and distributor of *Mathematica*. Here many links are available for support, add-on packages, books, the complete *Mathematica* book on-line, WebMathematica, GridMathematica, etc.

www.mathsource.com: *MathSource* is a vast electronic library of *Mathematica* materials, including immediately accessible *Mathematica* programs, documents, journals (*Mathematica* in Education, the *Mathematica* Journal) and many, many examples. Established in 1990, *MathSource* offers a convenient way for *Mathematica* developers and users to share their work with others in the *Mathematica* community. In *MathSource* you can either browse the archive or search by author, title, keyword, or item number.

There are many introductions to *Mathematica* and pages with helpful links. Here are some examples:

Tour of Mathematica (also available in the helpbrowser of *Mathematica*)
www.verbeia.com/mathematica/tips/tip_index.html Ted's Tips and Tricks
www.mathematica.ch/ (in German)
phong.informatik.uni-leipzig.de/~kuska/mview3d.html/ (MathGL3d, an OpenGL translator for Graphics3D structures)
www.unca.edu/~mcmcclur/mathematicaGraphics/ *Mathematica* graphics examples
www.wolfram.com/products/applications/parallel/ Parallel computing toolkit
www.wolfram.com/solutions/mathlink/jlink/ Java toolkit
www.math.wright.edu/Calculus/Lab/Download/ Calculus teaching material
forums.wolfram.com/mathgroup/ MathGroup newsgroup archive
mathforum.org/math.topics.html MathForum by Drexler University
integrals.wolfram.com/ The Integrator

B. The concept of convolution

B.1 Convolution

```
<< FrontEndVision`FEV`;
Show[Import["retinapsf.gif"], ImageSize -> 150];
```

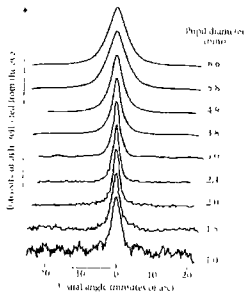


Figure B.1 Experimental measurements of light that has been reflected from a human eye looking at a fine line ('line spread function'). The reflected light has been blurred by double passage through the optics of the eye. The amount of unsharpness is a function of the pupil diameter: with wider pupil opening the line got more blurred. Source: Campbell and Gubish, 1996. Taken from [Wandell1995].

We take the example of the not-exactly-sharp projection of images on the human retina. Lens errors, cornea distortions, the retinal nerve tissue: they all contribute to the 'smearing effect'.

In fact, *every* point in the stimulus gives rise to the same blurring effect. Figure 1 shows some actual measurements at various pupil diameters. Suppose that the profile of the blur function is a Gaussian kernel. Then every point of the stimulus will become a Gaussian function when projected on the retina. We study the 1-D case, and call the blur function **g[x,σ]**. The blur function is also called the *filter* or the *kernel*, or the *operator* (sometime we encounter the name *stencil* or *template*).

$$g[x\_,\ \sigma\_] := \frac{1}{\sqrt{2\,\pi\,\sigma^2}}\ \text{Exp}\left[-\frac{x^2}{2\,\sigma^2}\right];$$

We define the input signal as **f[x]**, and the output signal as **g[x]**. We can think of the input signal as a series of points next to each other, and each of these input points gives rise to a blurred output point. We plot the input point function **fpnt[x]** (a very narrow function around the point **x=0**) at positions x=0 and x=2:

```
fpnt[x_] = If[-0.05 < x < 0.05, 1, 0];
Block[{$DisplayFunction = Identity},
pl = Plot[fpnt[x], {x, -4, 4}]; p2 = Plot[fpnt[x - 2], {x, -4, 4}]];
Show[GraphicsArray[{pl, p2}], ImageSize -> 300];
```

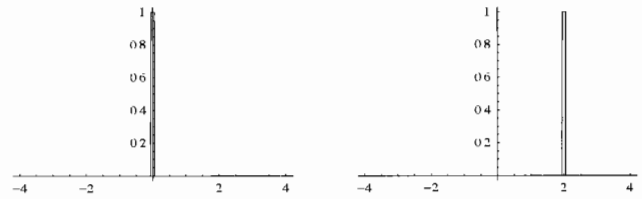


Figure B.2 The sampling function, at two arbitrary positions.

As an input signal we take an example function consisting of a sum of sine functions of different frequencies. We can approximate this function as a set of points very close to each other. The amplitude of each point is modulated with the input function, so we plot the product **f[α] fpnt[x-α]**.

```
f[x_] = Sin[x] + 0.5 Sin[5 x];
Block[{$DisplayFunction = Identity},
pl = Plot[f[x], {x, -4, 4}];
  p2 = Show[Table[Plot[f[α] fpnt[x - α],
      {x, -4, 4}, PlotPoints -> 160], {α, -4, 4, .1}]]];
Show[GraphicsArray[{pl, p2}], ImageSize -> 350];
```

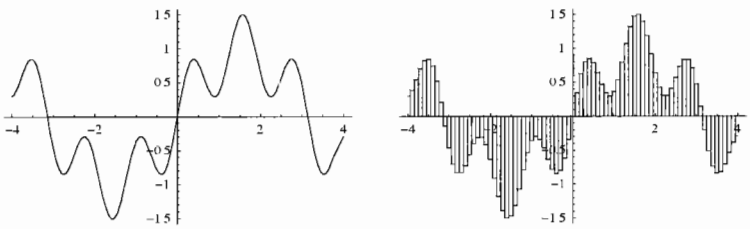


Figure B.3 Left: The input signal $f(x)$, right: the sampled representation.

Every point is blurred on its own, so we replace every pointfunction **fpnt[α]** by its blurred version **g[α]**. Every point is widened substantially, and the final step is adding all these responses together.

```
Block[{$DisplayFunction = Identity},
  pl = Show[Table[Plot[f[α] gauss[x - α, 1],
      {x, -4, 4}, PlotPoints -> 160], {α, -4, 4, .1}]];
  h[x_] = Sum[f[α] gauss[x - α, 1], {α, -6, 6, .1}];
p2 = Plot[h[x], {x, -4, 4}]];
```

```
Show[GraphicsArray[{p1, p2}], ImageSize -> 350];
```

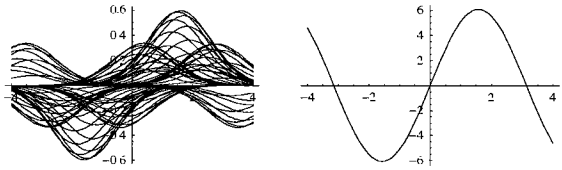


Figure B.4 Left: every sample gives rise to a kernel, at the respective position with the amplitude of the sample. Right: the summed response is the convolution.

In the limit of making the plot functions smaller and smaller we get a summation described by an integral: the convolution-integral. So we may write for the output **h[x]**:

$$\mathbf{h[x]} = \int_{-\infty}^{\infty} \mathbf{f[\alpha]\ g[x-\alpha,\ \sigma]}\ d\alpha$$

It describes exactly how the output **h[x]** looks like when an input signal **f[x]** is processed by a system **g[x]**. We say that the signal **f[x]** is filtered with a filter or kernel **g[x]**. We also call **g[x]** the pointspread-function of the system. Note that the convolution-integral integrates from $-\infty$ to ∞, indicating that we integrate over the whole length of the signal. The parameter α is the so-called shift variable. (sometimes the convolution integral is called the shift-integral).

Actually, the sum above is an approximation. The exact output is given by the convolution-integral. Let us plot the integral, and draw your conclusions yourself. We have approximated the solution quite well:

```
h[x_] := ∫_-∞^∞ f[α] g[x - α, 1] dα;
p5 = Plot[Evaluate[h[x]], {x, -4, 4}, ImageSize -> 160];
```

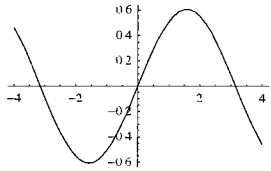


Figure B.5 The convolution as an exact integral of the analytical input function and kernel.

The output signal looks different from the input signal, and this is exactly what we wanted: we have filtered the high frequencies out of the signal. The frequency **Sin[5x]** is virtually eliminated, as the figure above is almost the low frequency **Sin[x]** function, as we put in the input. This low frequency filter is however 40% attenuated due to the filtering (check the amplitudes).

A short form for writing the convolution integral is the symbol $\otimes$, the convolution operator. The function **h** is the convolution of the function **f** with **g**

```
h[x] = f[x] ⊗ g[x]
```

The convolution operator is a *linear* operator:

```
(h1[x] + h2[x]) ⊗ g[x] = h1[x] ⊗ g[x] + h2[x] ⊗ g[x];
(a f[x]) ⊗ g[x] = a (f[x] ⊗ g[x])
```

In 2-D we get:

$$\mathtt{h[x\_, y\_] := \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f[\alpha, \beta]\, g[x-\alpha, y-\beta]\, d\alpha\, d\beta}$$

B.2 Convolution is a product in the Fourier domain

It is often not so easy to calculate the convolution integral. It is often much easier to calculate the integral of the convolution in the Fourier domain. This section explains how this works. We start from the convolution of the function $h(x)$ with the kernel $g(x)$:

$$\mathtt{f(x) = \int_{-\infty}^{\infty} h(y)\, g(x-y)\, dy}$$

The Fourier transform $F(\omega)$ of $f(x)$ is then by definition

$$\mathtt{F(\omega) = \int_{-\infty}^{\infty} f(x)\, e^{-i\omega x}\, dx = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(y)\, g(x-y)\, dy\, e^{-i\omega x}\, dx}$$

We reorder the terms in the integral and make the substitution $x - y = \tau$, so we have $x = \tau + y$ and $e^{-i\omega x} = e^{-i\omega\tau} . e^{-i\omega y}$:

$$\mathtt{\int_{-\infty}^{\infty} h(y) \int_{-\infty}^{\infty} e^{-i\omega x}\, g(x-y)\, dx\, dy = \int_{-\infty}^{\infty} h(y) \int_{-\infty}^{\infty} e^{-i\omega(\tau+y)}\, g(\tau)\, d(\tau+y)\, dy}$$

When we integrate to τ, we keep y constant, so $d(\tau+y)$ is equal to $d\tau$. So we get:

$$\mathtt{\int_{-\infty}^{\infty} h(y) \int_{-\infty}^{\infty} e^{-i\omega\tau}\, e^{-i\omega y}\, g(\tau)\, d\tau\, dy}$$

Because $e^{-i\omega y}$ is constant in the integration to τ, we may bring it as a constant outside the integral:

$$\mathtt{\int_{-\infty}^{\infty} h(y)\, e^{-i\omega y}\, dy \int_{-\infty}^{\infty} e^{-i\omega\tau}\, g(\tau)\, d\tau = H(\omega) . G(\omega)}$$

where $H(\omega)$ is the Fourier transform of the function $h(x)$ and $G(\omega)$ is the Fourier transform of the kernel $g(x)$. So the Fourier transform of a convolution is the product of the Fourier transform of the filter with the Fourier transform of the signal. To put it otherwise:

$$\begin{array}{ccccc} f(x) & = & h(x) & \otimes & g(x) \\ \updownarrow & & \updownarrow & & \updownarrow \\ F(\omega) & = & H(\omega) . G(\omega) & & \end{array}$$

We can calculate a convolution $h \otimes g$ by calculating the Fourier transforms $H(\omega)$ and $G(\omega)$, multiply them to get $F(\omega)$, and we get $f(x)$ by taking the inverse Fourier transform of $F(\omega)$. Often this is a much faster method then calculating the convolution integral, because the routines that calculate the Fourier transform are so fast. We look at an example where we filter noise from the data. We simulate a signal of a noisy ECG recording by some sine functions:

```
data = Table[
   N[Sin[2 π/256 2 t] + Sin[2 π/256 6 t] + 0.75 (Random[ ] - 1 / 2)], {t, 256} ] ;
ListPlot[data, AspectRatio -> .2, PlotJoined -> True,
  ImageSize -> 400, AxesLabel -> {"time", "Ampl"}];
```

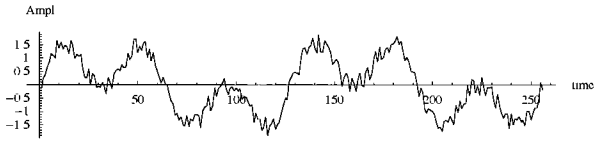


Figure B.6 A discrete input signal with additive uncorrelated uniform noise.

We make a filter with a Gaussian shape and shift it to the origin:

```
σ = 5.; kernel = Table[gauss[x, σ], {x, -128, 127}];
kernel = RotateLeft[kernel, 128];
ListPlot[kernel, PlotRange -> {{0, 256}, All}, PlotJoined -> True,
  AspectRatio -> .2, AxesLabel -> {"Freq", "Ampl"}, ImageSize -> 400];
```

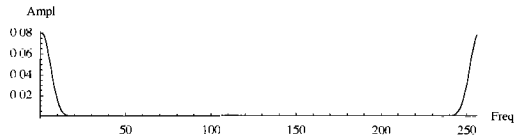


Figure B.7 The Gaussian kernel in the spatial domain, shifted to the origin. The right half of the frequency axis (i.e. 129-256) is often called the negative frequency axis.

This is a low-pass filter. We now filter (convolve) in the Fourier domain (note that a space denotes multiplication in *Mathematica*):

```
conv = Sqrt[256] InverseFourier[Fourier[data] Fourier[kernel]];
ListPlot[ conv, AspectRatio -> .2, PlotJoined -> True,
  AxesLabel -> {"time", "Ampl"}, ImageSize -> 400];
```

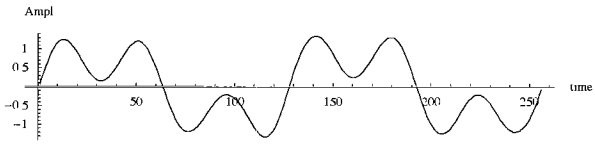


Figure B.8 The filtered result.

As expected, the noise did not pass the filter.

C. Installing the book and packages

C.1 Content

TheCD-ROM with the book "Front-End Vision and Multiscale Image Analysis" contains the following:

- the ZIP archive file **FrontEndVision.zip** with all the necessary files with pathnames. This file contains:
- the directory **FrontEndVision/Documentation/English** with the collection of *Mathematica* notebooks of the total book. All chapters, appendices, table of contents and the referencelist are separate notebooks. This is the sourcecode of the book.
- the notebook **FrontEndVision/FEV.nb** and package **FrontEndVision/FEV.m**. This notebook contains the initializations and the image analysis functions used throughout the book. The package is loaded as the first command in every notebook of this book that contains executable code. Do not edit this package, as it is automatically generated from **FEV.nb** upon saving.
- the directory **FrontEndVision/images** with the images, figures and other data used in the book.
- the directory **FrontEndVision/Documentation/English/pdf** with the PDF versions of all notebooks.
- the directory **FrontEndVision/LiveGraphics3D** with the source code and an example of the interactive 3D Java viewer LiveGraphics3D.
- the stylenotebook
SystemFiles/FrontEnd/StyleSheets/FrontEndVision.nb, which contains the style directives (layout and makeup) of the notebooks on your screen and printer.

- the stylenotebook
SystemFiles/FrontEnd/StyleSheets/PackageNotebook.nb, which contains directives for notebooks to generate a package automatically (as **FEV.nb**).
- the file **FrontEndVision/binary.exe** with is a compiled C++ executable, to read binary data fast. This is now integrated in *Mathematica*, it may be useful for older versions.

When the kernel of *Mathematica* is not available, the notebooks can be read with the free stand-alone notebook reader ***MathReader***. This is the front-end program, to read and view the *Mathematica* notebooks of the book. The latest version of ***MathReader*** can be downloaded for free from www.wolfram.com/products/mathreader. Versions are supplied for Windows, Macintosh, Linux and Unix.
The notebooks of this book can only be run on *Mathematica* 4.0 and later.

C.2 Installation for all systems

For: Windows 95/98/2000/NT/XP, Macintosh, Unix, Linux.
Run the command **$TopDirectory**. This returns the directory where the file **FrontEndVision.zip** file should be copied and executed to uncompress the files in that directory.

```
$TopDirectory
```

```
C:\Program Files\Wolfram Research\Mathematica\4.2
```

All files will be automatically installed in the right locations. E.g. for Windows users:
At the end you should have the following directory where you find your notebooks:
C:\Program Files\Wolfram Research\Mathematica\4.1\AddOns\Applications\
FrontEndVision\Documentation\English

C.3 Viewing the book in the Help Browser

The book can best be interactively viewed in the Help Browser of *Mathematica:*

1. Run the menu command "Rebuild Help Index" in the Help menu in the top menubar in *Mathematica.*

The commands in the file
$TopDirectory\AddOns\Applications\FrontEndVision\Documentation\English\BrowserCategories.m contains the instructions to generate the appropriate menus in the Help browser;

2. Open de Help Browser and click on the button "AddOns". In the left menu panel the book is now available.

The notebooks viewed with the Help Browser can all be executed as normal notebooks. However, the Help browser files are read-only. A particular useful feature is to copy and paste the commands in the book through the Help Browser as directly usable code into your own notebooks.

The book has an extensive reference list. The last chapter is the extensive alphabetical index. The Help Browser enables quick hypertext jumping to the appropriate chapter and cell where the selected keyword appears. The same feature is true for the TableOfContent notebook.

Some suggestions:

- Increase the magnification of the Help Browser for all notebooks to e.g. 150% by setting the magnification globally for this notebook. This is accomplished , after selection the Help Browser as current notebook, with the Option Inspector, available in the Format menu. 'Show option values for:' *global.* Search for the keyword 'magnification' and set the value to your liking.

C.4 Sources of additional applications

Get the last versions of
- The OpenGL 3D viewer MathGL3d:
phong.informatik.uni-leipzig.de/~kuska/mathgl3dv3/;
- The Java interactive 3D animation viewer LiveGraphics3D:
wwwvis.informatik.uni-stuttgart.de/~kraus/LiveGraphics3D/.

D. First Start with *Mathematica*: Tips & Tricks

This notebook is only available in the electronic version.

1. Evaluation

1.1 Startup

1.2 Cells

1.3 Prevent unwanted (large) output to the screen

1.4 Interrupt, stop and start over

1.5 Manual shortcuts

1.6 The help browser

1.7 Packages

1.8 Setting the Path to find files

1.9 Write in mathematical notation

1.10 The size of notebooks

1.11 Checking for proper print layout

1.12 Suggestions

2. Images

2.1 Read an image

2.2 Take a submatrix, a subimage

2.3 Sampling points from an image

2.4 Draw a contour on the image

2.5 Generate an animation

2.6 Automatic numbering objects

2.7 Check the internal structure of a cell

2.8 Resizing Graphics without regenerating it

2.9 Display a group of images

2.10 Text on graphics

3. Programming

3.1 Fast summation

3.2 Random numbers and noise images

3.3 Gaussian noise

3.4 Interpolation

3.5 Pure functions

3.6 Use internal functions for speed

3.7 Good practice

4. 3D

4.1 Changing the ViewPoint

4.2 Interactive 3D display

4.3 LiveGraphics3D functions

References

[1] P. Abbott (ed.), "Tricks of the trade", The *Mathematica* Journal, Wolfram Media Inc., vol. 7, no. 2, 105-127, 1998.

[2] M. Abramowitz, "Handbook of mathematical functions". Dover Publications, 1971.

[3] S. T. Acton, A. C. Bovik, and M. M. Crawford, "Anisotropic diffusion pyramids for image segmentation", in Proc. first Intern. Conf. on Image Processing, pp. 478-482, IEEE, 1994.

[4] S. T. Acton, "Diffusion-based edge detectors", in: Handbook of image and video processing, A. Bovik ed., Academic Press, San Diego, 2000.

[5] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion", Journal of the Optical Society of America-A, vol. 2, no. 2, pp. 284-299, 1985. Also appeared as MIT-MediaLab-TR148. September 1990.

[6] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion", Journal of the Optical Society of America-A, vol. 2, no. 2, pp. 284-299, 1985. Also appeared as MIT-MediaLab-TR148. September 1990.

[7] J. Aggarwall and N. Nandhakumar, "On the computation of motion from sequences of images", IEEE Tr. PAMI, vol. 76, no. 8, pp. 917-935, 1988. A review.

[8] A. Almansa and T. Lindeberg, "Enhancement of fingerprint images using shape-adaptated scale-space operators", IEEE Tr. on Image Processing. In: J. Sporring, M. Nielsen, L. Florack, and P. Johansen (eds.) Gaussian Scale-Space Theory: Proc. PhD School on Scale-Space Theory , Copenhagen, Denmark, May 1996, Kluwer Academic Publishers, 1997.

[9] J. M. Alonso and L. M. Martinez, "Functional connectivity between simple cells and complex cells in cat striate cortex", Nature Neuroscience, vol. 1, pp. 395-403, 1998.

[10] L. Alvarez, P. L. Lions, and J. M. Morel, "Image selective smoothing and edge detection by nonlinear diffusion. II", SIAM Journal on Numerical Analysis, vol. 29, pp. 845-866, June 1992.

[11] L. Alvarez, F. Guichard, P. L. Lions, and J. M. Morel, "Axioms and fundamental equations of image processing", Archives for Rational Mechanics, vol. 123, pp. 199-257, September 1993.

[12] L. Alvarez and J. M. Morel, "Formalization and computational aspect of image analysis", Acta Numerica, 1994.

[13] L. Alvarez and L. Mazorra, "Signal and image restoration using shock filters and anisotropic diffusion", SIAM Journal on Numerical Analysis, vol. 31, pp. 590-605, January 1994.

[14] L. Alvarez and J. M. Morel, "Morphological approach to multiscale analysis", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), Computational Imaging and Vision, pp. 229-254, Kluwer Academic Publishers B.V., 1994.

[15] L. Alvarez, "Images and PDE's", in Proc. of 12th Intern. Conf. on Analysis and Optimization of Systems (M.-O. Berger, R. Deriche, I. Herlin, J. Jaffré, and J.-M. Morel, eds.), vol. 219 of Lecture Notes in Control and Information Sciences, pp. 3-14, Springer, London, 1996.

[16] W. F. Ames, "Numerical methods for partial differential equations". New York, San Francisco: Academic Press, 1977.

[17] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 9, pp. 855-867, 1990.

[18] F. R. Amthor, E. S. Takahashi and C. W. Oyster, "Morphologies of rabbit ganglion cells with concentric receptive fields", Jorn. of Comparitive Neurology, vol. 280, pp. 72-96, 1989.

[19] S. Angenent, "On the formation of singularities in the curve shortening flow", Journal of Differential Geometry, vol. 33, pp. 601-633, 1991.

[20] V. I. Arnold, "Singularity theory". Cambridge: Cambridge University Press, 1981.

[21] H. Asada and M. Brady, "The curvature primal sketch", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, no. 1, pp. 2-14, 1986.

[22] J. J. Atick and A. N. Redlich, "Mathematical model of the simple cells in the visual cortex", Biological Cybernetics, vol. 63, pp. 99-109, 1990.

[23] N. Ayache, "Medical computer vision, virtual reality and robotics", Image and Vision Computing, vol. 13, pp. 295-313, May 1995.

[24] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda, "Uniqueness of the Gaussian kernel for scale-space filtering", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, no. 1, pp. 26-33, 1986.

[25] S. Back, H. Neumann, and H. S. Stiehl, "On segmenting computed tomograms", in Proceedings of the 3rd Intern. Symposium CAR '89 (H. U. Lemke, M. L. Rhodes, C. C. Jaffee, and R. Felix, eds.), Berlin, Springer-Verlag, 1989.

[26] S. Back, H. Neumann, and H. S. Stiehl, "On scale-space edge detection in computed tomograms", in Proceedings of the 11th DAGM-Symposium, Hamburg (H. Burkhardt, K.-H. Hoehne, and B. Neumann, eds.), Berlin, Springer-Verlag, 1989.

[27] R. Balart, "Matrix reformulation of the Gabor transform", Optical Engineering, vol. 31, pp. 1235-1242, June 1992.

[28] P. Baldi and W. Heiligenberg, "How sensory maps could enhance resolution through ordered arrangements of broadly tuned receivers", Biological Cybernetics, vol. 59, pp. 313-318. 1988.

[29] C. Ballester and M. Gonzalez, "Affine invariant multiscale segmentation by variational methods", in Eighth Workshop on Image and Multidimensional Image Processing, (Cannes), pp. 220-221, IEEE, September 8-10 1993.

[30] D. Bar-Natan, "Random dot stereograms", The Mathematica Journal, vol. 1, no. 3, pp. 69-75, 1991.

[31] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques", Intern. Journal of Computer Vision, vol. 12, no. 1, pp. 43-77, 1994.

[32] R. Bauer and B. M. Dow, "Local and global properties of striate cortical organization: an advanced model", Biological Cybernetics, vol. 64, pp. 477-483, 1991.

[33] S. Beauchemin and J. Barron, "The computation of optic flow", ACM Computing Surveys, vol. 27, no. 3, pp. 433-467, 1995.

[34] A. Bebernes and D. Eberly, Mathematical Problems from Combustion Theory. Springer-Verlag, 1989.

[35] J. V. Beck, K. D. Cole, A. Haji-Sheikh, and B. Litkouhi, Heat Conduction using Green's Functions. London: Hemispere Publishing Corporation, 1992.

[36] W. Beil, "Steerable filters and invariance theory", Pattern Recognition Letters, vol. 16, no. 11, pp. 453-460, 1994.

[37] B. Bell and L. F. Pau, "Contour tracking and corner detection in a logic programming environment", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 9, pp. 913-917, 1990.

[38] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 509-522, 2002.

[39] B. M. Bennett, D. D. Hoffman, and C. Prakash, Observer Mechanics. A Formal Theory of Perception. London: Academic Press, 1989.ISBN 0-12-0888635-9.

[40] F. Bergholm, "Edge focusing", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 9, pp. 726-741, November 1987.

[41] G. L. Bilbro, W. E. Snyder, S. J. Garnier, , and J. W. Gault, "Mean field annealing: A formalism for constructing GNC-like algorithms", IEEE Trans. Neural Networks, vol. 3, January 1992.

[42] T. O. Binford, "Inferring surfaces from images," Artificial Intelligence, vol. 17, pp. 205-244, 1981.

[43] M. Bister, J. Cornelis, and A. Rosenfeld, "A critical view of pyramid segmentation algorithms", Pattern Recognition Letters, vol. 11, pp. 605-617, 1990.

[44] E. Björkman, J. C. Zagal, T. Lindeberg, P. E. Roland, "Evaluation of design options for scale-space primal sketch analysis of brain activation images", HBM'2000, Intern. Conf. on Functional Mapping of the Human Brain, San Antonio, Texas, 2000.

[45] N. Blachman, "*Mathematica*: A Practical Approach". 2nd edition, *Mathematica* 3.0, Prentice Hall, 631 pp., ISBN 0132592010, 1999.

[46] A. Blake and A. Zisserman, "Visual Reconstruction". Cambridge, Mass.: MIT Press, 1987.

[47] H. Blakemore, C. Blakemore and H. Barlow, Images and Understanding: Thoughts about Images, Ideas about Understanding, Cambridge University Press, April 1989.

[48] C. Blakemore (Ed.), Vision: Coding and Efficiency, Cambridge University Press, January 1990.

[49] W. Blaschke and K. Reidemeister, Differential Geometry, vol. 1-2. Springer-Verlag, 1923.

[50] G.G. Blasdel and G. Salama. "Voltage-sensitive dyes reveal a modular organization in money striate cortex", Nature, vol. 321, pp. 579-585, 1986.

[51] J. Blom, "Modellen voor de funktionele ordening van (1-dim.) zintuig-systemen", Tech. Rep. V-mff-38-85, Department of Medical and Physiological Physics, University of Utrecht, Princetonplein 5, 3584 CC Utrecht, Netherlands, 1985.

[52] J. Blom, "Affine Invariant Corner Detection", in: PhD Thesis, Utrecht University, NL-Utrecht, 1991.

[53] J. Blom, "Topological and Geometrical Aspects of Image Structure". PhD thesis, Utrecht University, 1992.

[54] J. Blom, B. M. ter Haar Romeny, and J. J. Koenderink, "Affine invariant corner detection", tech. rep., 3D Computer Vison Research Group, Utrecht University NL, 1992.

[55] J. Blom, J. J. Koenderink, B. M. ter Haar Romeny, and A. M. L. Kappers, Topological image-structure for a discrete image on a hexagonal lattice with finite intensity sampling", J. of Vis. Comm. and Im. Repr., 1992.

[56] J. Blom, B. M. ter Haar Romeny, A. Bel, and J. J. Koenderink, "Spatial derivatives and the propagation of noise in Gaussian scale-space", J. of Vis. Comm. and Image Repr., vol. 4, pp. 1-13, March 1993.

[57] J. A. Bloom and T.R. Reed, "A Gaussian Derivative-Based Transform", IEEE Tr. on Image Processing, Vol. 5, No. 3, 1996.

[58] D. Blostein and N. Ahuja, "Representation and three-dimensional interpretation of image texture: An integrated approach", in Proc. 1st Int. Conf. on Computer Vision, (London), pp. 444-449, IEEE Computer Society Press, 1987.

[59] H. Blum, "Biological shape and visual science", J. Theor. Biology, vol. 38, pp. 205-287, 1973.

[60] G. Bluman and S. Kumei, "A remarkable nonlinear diffusion equation", Journal of Mathematical Physics, vol. 21, pp. 1019-1023, May 1980.

[61] 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". J. Neurosci. vol. 13, pp. 4157-4180, 1993.

[62] F. L. Bookstein, "Principal warps: thin-plate splines and the decomposition of deformations", IEEE trans. Pattern Analysis and Machine Intelligence, Vol. 11, No. 6, pp. 567-585, 1989.

[63] G. Borgefors, "Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm", IEEE Trans. Pattern Anal. Machine Intell., vol. 10, no. 6, 1988.

[64] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 1, pp. 55-73, 1990.

[65] A. Bovik, "Handbook of image and video processing", Academic Press, 2000.

[66] B. B. Boycott and H. Wässle. "The morphological types of ganglion cells of the domestic cat's retina". Journal of Physiology, 240:379-419, 1974.

[67] C. B. Boyer, A History of Mathematics. Brooklyn,New York: Princeton University Press, January 1968. First Princeton Paperback Printing 1985.

[68] K. A. Brakke, "The motion of a surface by its mean curvature", tech. rep., Princeton University Press, Princeton NY, 1978.

[69] A. Brauer, "Über die Nullstellen der Hermiteschen Polynome", Mathematische Annalen, vol. 107, pp. 87-89, 1933.

[70] L. Bretzner and T. Lindeberg, "Feature tracking with automatic selection of spatial scales", In Linde, Sparr (Eds.): Proc. Swedish Symposium on Image Analysis, SSAB'96 Lund, Sweden, pp. 24-28, March 1996. Extended version in: Computer Vision and Image Understanding. vol. 71, pp. 385--392, Sept. 1998.

[71] L. Bretzner and T. Lindeberg, "On the handling of spatial and temporal scales in feature tracking", Proc. First Intern. Conf. on Scale-Space Theory in Computer Vision, Utrecht, Netherlands, B.M. ter Haar Romeny ed., Springer-Verlag Lecture Notes in Computer Science, volume 1252. July 2-4, 1997.

[72] L. Bretzner and T. Lindeberg, "Qualitative multi-scale feature hierarchies for object tracking", in M. Nielsen, P. Johansen, O. F. Olsen and J. Weickert (Eds) Proc. 2nd Intern. Conf. on Scale-Space Theory in Computer Vision, Corfu, Greece, September 1999. Springer Lecture Notes in Computer Science, vol 1682, pp. 117--128. Extended version in J. of Visual Communication and Image Representation, 11, 115-129, 2000.

[73] R. W. Brockett and P. Maragos, "Evolution equations for continuous-scale morphology", in Proc. Intern. Conf. on Acoustics, Speech and Signal Processing, pp. 125-128, IEEE, 1992.

[74] J. W. Bruce and P. J. Giblin, Curves and Singularities.Cambridge: Cambridge University Press, 1984.

[75] V. Bruce, P. R. Green, and M. A. Georgeson, Visual Perception. Hove, East Sussex, UK: Psychology Press, 1996.

[76] A. M. Bruckstein and A. N. Netravali, "On differential invariants of planar curves and recognizing partially occluded planar shapes", in Proc. of Visual Form Workshop, (Capri), Plenum Press, May 1990.

[77] A. M. Bruckstein, R. J. Holt, A. N. Netravali, and T. J. Richardson, "Invariant signatures for planar shape recognition under partial occlusion", in Proceedings of the 11th IAPR international conference on pattern recognition, 1992. Long version in Computer Vision, Graphics and Image Processing: Image Understanding, vol. 58, nr. 1, pp. 49-65, 1993.

[78] K. Brunnstrom, J.-O. Eklundh, and T. Lindeberg, "On scale and resolution in active analysis of local image structure", Image and Vision Computing, vol. 8, no. 4, pp. 289-296, 1990.

[79] K. Brunnstrom, T. Lindeberg, and J.-O. Eklundh, "Active detection and classification of junctions by foveation with a head-eye system guided by the scale-space primal sketch", in Proc. second European Conf. on Computer Vision (G. Sandini, ed.), vol. 588 of Lecture Notes in Computer Science, (Santa Margherita Ligure, Italy), pp. 701-709, Springer-Verlag, May 1992.

[80] B. Buck, A. C. Merchant, and S. M. Perez, "An illustration of Benford's first digit law using alpha decay half lives". European Journal of Physics, vol. 14, pp. 59-63, 1993.

[81] C. A. Burbeck and S. M. Pizer, "Object representation by cores: indentifying and representing primitive spatial regions", Tech. Rep. TR94-048b, University of North Carolina at Chapel Hill, 1994.

[82] P. J. Burt, T. H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation", IEEE Tr. on Systems, Man, and Cybernetics, vol. 11, no. 12, pp. 802-825, 1981.

[83] P. J. Burt, "Fast filter transforms for image processing", Computer Vision, Graphics, and Image Processing, vol. 16, pp. 20-51, 1981.

[84] P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code", IEEE Trans. Communications, vol. 9, no. 4, pp. 532-540, 1983.

[85] P. J. Burt, "Multiresolution images precessing and analysis", chapter in: The pyramid as a structure for efficient computation, pp. 6-35. Berlin: Springer Verlag, 1984. A. Rosenfeld, Ed.

[86] P. Buser and M. Imbert, "Vision". London, England: The MIT Press, 1994.

[87] J. Canny, "A computational approach to edge detection", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679-698, 1986.

[88] V. Cantoni and S. Levialdi, eds., Pyramidal Systems for Computer Vision. Berlin: Springer-Verlag, 1986.

[89] E. Cartan, "La théorie des groupes finis et continus et la géometrie differentielle traitées par la méthode du répère mobile". Gauthiers-Villars, 1937.

[90] E. Cartan, "Les problèmes d'équivalence", in Oeuvres Complètes, vol. 2, pp. 1311-1334, Paris: Gauthiers-Villars, 1952.

[91] E. Cartan, "Leçons sur la geometrie des espaces de Riemann". Paris: Gauthier-Villars, 2 ed., 1963.

[92] D. Casasent and D. Psaltis, "Position, rotation, and scale invariant optical correlation", Applied Optics, vol. 15, no. 7, pp. 1795-1799, 1976.

[93] V. Caselles, F. Catté, T. Coll, and F. Dibos, "A geometric model for active contours in image processing", Numerische Mathematik, vol. 66, pp. 1-31, 1993.

[94] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic snakes", tech. rep., Department of Mathematics, University of Illes Balears, Palma de Mallorca, Spain, 1994.

[95] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours", in Proc. Fifth Intern. Conf. on Computer Vision (E. Grimson, S. Shafer, A. Blake, and K. Sugihara, eds.), pp. 694-699, 1995.

[96] S. Castan, J. Zhao and J. Shen, "Optimal filter for edge detection methods and results". In Proc. First Eur. Conf. on Computer Vision, pp. 13-17, 1990.

[97] F. Catté, P. L. Lions, J. M. Morel, and T. Coll, "Image selective smoothing and edge detection by nonlinear diffusion", SIAM Journal on Numerical Analysis, vol. 29, pp. 182-193, February 1992.

[98] F. Catté, F. Guichard, and G. Köpfler, "A morphological approach to mean curvature motion", Tech. Rep. 9310, CEREMADE, Université Paris Dauphine, 1993.

[99] F. Catté, "Convergence of iterated affine and morphological filters by nonlinear semigroup theory", in Proc. of 12th Intern. Conf. on Analysis and Optimization of Systems (M. O. Berger, R. Deriche, I. Herlin, J. Jaffré, and J. M. Morel, eds.), vol. 219 of Lecture Notes in Control and Information Sciences, pp. 125-133, Springer, London, 1996.

[100] A. Cayley, "On contour and slope lines", The London, Edinghburgh and Dublin Philosophical Magazine and J. of Science, vol. 18, no. 120, pp. 264-268, 1859.

[101] A. Chehikian and J. L. Crowley, "Fast computation of optimal semi-octave pyramids", in Proc. 7th Scand. Conf. on Image Analysis, (Aalborg, Denmark), pp. 18-27, August 1991.

[102] J. S. Chen and G. Medioni, "Detection, localization and estimation of edges", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 11, pp. 191-198, 1989.

[103] M. Chen and P. Yan, "A multiscale approach based on morphological filtering", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 694-700, 1989.

[104] W. Chen, T. Kato et al., "LGN activation during visual imagery tasks shown by fMRI", Proc. 2nd Intern. Conf. on Functional Mapping of the Human Brain 1996.

[105] F. H. Cheng and W. H. Hsu, "Parallel algorithm for corner finding on digital curves", Pattern Recognition Letters, vol. 8, pp. 47-53, 1988.

[106] P. S. Churchland and T. J. Sejnowski, "The Computational Brain". MIT Press, 1992.

[107] J. Clark, "Singularity theory and phantom edges in scale-space", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 10, no. 5, 1988.

[108] J. J. Clark, "Authenticating edges produced by zero-crossing algorithms", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 11, pp. 43-57, 1989.

[109] A. Clebsch, "Theorie der Binären Algebraischen Formen". Leipzig: Verlag von Teubner, 1872.

[110] J. Coggins and A. Jain, "A spatial filtering approach to texture analysis", Pattern Recognition Letters, vol. 3, pp. 195-203, 1985.

[111] M. A. Cohen and S. Grossberg, "Neural dynamics of brightness perception: Features, boundaries, diffusion, and resonance", Perception and Psychophysics, vol. 36, no. 5, pp. 428-456, 1984.

[112] I. Cohen, L. D. Cohen, and N. Ayache, "Using deformable surfaces to segment 3D images and infer differential structures", Computer Vision, Graphics, and Image Processing, vol. 56, pp. 242-263, 1992.

[113] T. Cohignac, F. Eve, F. Guichard, and J. M. Morel, "Numerical analysis of the fundamental equation of image processing", Tech. Rep. 9254, CEREMADE, Universite Paris Dauphine, 1992.

[114] T. Cohignac, F. Guichard, and J. M. Morel, "Multiscale analysis of shapes, images and textures", in Eighth Workshop on Image and Multidimensional Image Processing, (Cannes), pp. 142-143, IEEE, September 8-10 1993.

[115] T. Cohignac, F. Eve, F. Guichard, and C. Lopez, "Affine morphological scale-space: Numerical analysis of its fundamental equation", tech. rep., Ceremade, Universite Paris Dauphine, 1993.

[116] R. Cormack and R. Fox, "The computation of retinal disparity", Perception and Psychophysics, vol. 37, no. 2, pp. 176-178, 1985.

[117] T. N. Cornsweet, "Visual perception", Academic Press, New York, 1970.

[118]G. H. Cottet, "Diffusion approximation on neural networks and applications for image processing", in Proc. Sixth European Conf. on Mathematics in Industry (F. Hodnett, ed.), (Stuttgart), pp. 3-9, Teubner, 1992.

[119] J. Crank, "Mathematics of diffusion". London: Oxford University Press, 1956.

[120] J. R. Crowley, "A representation for shape based on peaks and ridges in the Difference of Low-Pass Transform", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 6, no. 2, pp. 156-170, 1984.

[121] J. L. Crowley and R. M. Stern, "Fast computation of the difference of low pass Transform", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 6, pp. 212-222, 1984.

[122] J. L. Crowley and A. C. Sanderson, "Multiple resolution representation and probabilistic matching of 2-D gray-scale shape", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 9, no. 1, pp. 113-121, 1987.

[123] C. A. Curcio, K. R. Sloan, R. E. Kalina, A. E. Hendrickson, "Human photoreceptor topography". Journal of Comparative Neurology, vol. 292, pp. 497-523, 1990.

[124] D. M. Dacey and S. Brace, "A coupled network for parasol but not for midget ganglion cells in the primate retina", Visual Neuroscience, vol. 9, pp. 279-290, 1992.

[125] G. Dalmaso, J. M. Morel, and S. Solimini, "A variational method in image segmentation: Existence and approximation results", Acta Math., vol. 168, pp. 89-151, 1992.

[126] E. Dam & M. Lillholm, "Generic Events for the Isophote Curvature", Graduate project, University of Copenhagen, http://www.it-c.dk/people/erikdam, March 1999.

[127] E. B. Dam and Mads Nielsen, "Non-linear diffusion for interactive multi-scale watershed segmentation. Proceedings for MICCAI 2000, Lecture Notes in Computer Science, volume 1935, October 2000.

[128] E. Dam & M. Nielsen, "Non-Linear Diffusion for Interactive Multi-scale Watershed Segmentation", in Proceedings for MICCAI 2000, Pittsburg, Lecture Notes in Computer Science, vol. 1935, pp. 216-225, 2000.

[129] E. Dam, P. Johansen, O.F. Olsen, A. Thomsen, T. Darvann, A.B. Dobrzeniecki, N.V. Hermann, N. Kitai, S. Kreiborg, P. Larsen and M. Nielsen: "Interactive Multi-Scale Segmentation in Clinical Use", CompuRAD, ECR 2000, Vienna, 2000.

[130] J. Damon, "Local Morse theory for solutions to the heat equation and Gaussian blurring", J. of Differential Equations, vol. 115, no. 2, pp. 368-401, January 1995.

[131] J. Damon, Chapter "Local Morse Theory for Solutions to the Heat Equation and Gaussian Blurring" in "Gaussian Scale-Space Theory" (Edited by J. Sporring, M. Nielsen, L. Florack, & P. Johansen), Kluwer Academic Publishers, 1997.

[132] J. Damon, "Ridges and cores for two-dimensional images", Jour. Math. Imag. and Vision, vol. 10, pp. 163-174, 1999.

[133] P.-E. Danielsson and O. Seger, "Rotation invariance in gradient and higher order derivative detectors", Computer Vision, Graphics, and Image Processing, vol. 49, pp. 198-221, 1990.

[134] J. G. Daugman, "Uncertainty Relation for Resolution in Space, Spatial Frequency, and Orientation Optimized by Two-Dimensional Visual Cortical Filters", J. Opt. Soc. Am., Vol. 2, 1985.

[135] J. G. Daugman, "Two-dimensional spectral analysis of cortical receptive fields profile", Vision Research, vol. 20, pp. 847-856, 1980.

[136] J. G. Daugman, "Six formal properties of anisotropic visual filters: structural principles and frequency/orientation selectivity", IEEE Trans. Systems, Man, and Cybernetics, vol. 13, pp. 882-887, 1983.

[137] J. G. Daugman, "Uncertainty relation for resolution in space spatial frequency, and orientation optimized by two-dimensional visual cortical filters", Journal of the Optical Society of America-A, vol. 2, pp. 1160-1169, 1985.

[138] J. G. Daugman, "Pattern and motion vision without Laplacian zero crossings", Journal of the Optical Society of America-A, vol. 5, no. 7, pp. 1142-1148, 1988.

[139] J. G. Daugman, "Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression", IEEE Tr. Acoust., Speech, Signa Processing, vol. 36, no. 4, pp. 1169-1179, 1988.

[140] L. S. Davis, "A survey on edge detection techniques", Comp. Graph. and Image Proc., vol. 4, no. 3, pp. 248-270, 1975.

[141] G. C. DeAngelis, I. Ohzawa, and R. D. Freeman, "Depth is encoded in the visual cortex by a specialized receptive field structure.", Nature, vol. 352, pp. 156-159, 1991.

[142] G. C. DeAngelis, I. Ohzawa, and R. D. Freeman, "Spatiotemporal organization of simple-cell receptive fields in the cat's striate cortex", Journal of Neurophysiology, vol. 69, no. 4, pp. 1091-1135, 1993.

[143] G. C. DeAngelis, I. Ohzawa, and R. D. Freeman, "Receptive field dynamics in the central visual pathways", Trends Neurosci., vol. 18, pp. 451-458, 1995.

[144] G. C. DeAngelis, Geoffrey M. Ghose, Izumi Ohzawa, and Ralph D. Freeman, "Functional Micro-Organization of Primary Visual Cortex: Receptive Field Analysis of Nearby Neurons", The Journal of Neuroscience, vol. 19, no. 10, pp. 4046-4064, 1999. URL

[145] E. De Boer and P. Kuyper, "Triggered correlation", IEEE Trans. Biomedical Engineering, vol. 15, pp. 169-179, 1968.

[146] C. de Boor, "A practical guide to splines". Springer Verlag, New York, 1978.

[147] E. De Micheli. B. Caprile, P. Ottonello, and V. Torre, "Localization and noise in edge detection", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 10, no. 11, pp. 1106-1117, 1989.

[148] R. Deriche, "Using Canny's criteria to derive an optimal edge detector recursively implemented". Intern. J. of Computer Vision, Vol. 1, pp. 167-187, 1987.

[149] R. Deriche, "Fast algorithms for low-level vision", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 1, pp. 78-87, 1990.

[150] R. Deriche, "Recursively implementing the Gaussian and its derivatives", Proc. Second Intern. Conf. on Image Processing, pp. 263-267, Singapore, 1992.

[151] R. L. De Valois, N. P. Cottaris, L. E. Mahon, S. D. Elfar. J. A. Wilson, "Spatial and temporal receptive fields of geniculate and cortical cells and directional selectivity", Vision Research, vol. 40, pp. 3685-3702, 2000.

[152] F. Devernay and O. Faugeras, "Computing differential properties of 3-D shapes from stereoscopic images without 3-D models", in Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, (Seattle, Washington), pp. 208-213, June 1994.

[153] S. Dickinson, "Object representation and recognition", in: E. Lepore and Z. Pylyshyn (eds.), "What is Cognitive Science?", Basil Blackwell publishers, pp. 172--207, 1999.

[154] R. W. Ditchburn and B. L. Ginsborg, "Vision with a stabilized retinal image", Nature, vol. 170, pp. 36-37, 1952.

[155] B. Doolittle and E. Maclay, "The forest has eyes". The Greenwich Workshop Press, 1998.

[156] A. J. van Doorn, J. J. Koenderink, and M. A. Bouman, "The influence of the retinal inhomogeneity on the perception of spatial patterns", Kybernetik, vol. 10, pp. 223-230, 1972.

[157] L. Dorst and R. van den Boomgaard, "Morphological signal processing and the slope transform", Signal Processing, vol. 38, pp. 79-98, 1994.

[158] L. Dorst and R. van den Boomgaard, "Orientation-based representations for mathematical morphology". in Shape, Structure and Pattern Recognition (D. Dori and A. Bruckstein, eds.). (Nahariya,Israel), pp. 13-22, October 1993.

[159] B. Dubuc and S. W. Zucker, "Complexity, confusion, and perceptual grouping. Part I and II, Int. J. of Computer Vision, 42(1/2): 55-115, 2001; reprinted in J. Math. Imaging and Vision, 15(1/2): 55-115, 2001.

[160] R. Duits, L. M. J. Florack, B. M. ter Haar Romeny, and J. de Graaf, "On the axioms of scale-space theory." Proc. Fourth IASTED International Conference on Signal and Image Processing (SIP 2002), August 12-14, 2002. Kauai. Hawaii, USA.

[161] D. Eberly, D. Fritsch, and C. Kurak, "Filtering with a normalized Laplacian of a Gaussian filter", in Proceedings of the SPIE Intern. Symposium, Mathematical Methods in Medical Imaging, (San Diego, CA), 1992.

[162] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, "Ridges for image analysis." Journal of Mathematical Imaging and Vision, July 1993.

[163] D. Eberly, Geometric Analysis of Ridges in N-Dimensional Images. PhD thesis, University of North Carolina at Chapel Hill, Computer Science Department, 1994.

[164] D. Eberly, "A differential geometric approach to anisotropic diffusion", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), pp. 371-392, Dordrecht: Kluwer Academic Publishers, 1994.

[165] D. H. Eberly, Geometric Methods for Analysis of Ridges in N-Dimenional Images. PhD thesis, The University of North Carolina, Chapel Hill. North Carolina, January 1994. Department of Computer Vision.

[166] J. Elder and S.W. Zucker. Local scale control for edge detection and blur estimation. In Lecture Notes in Computer Science, pages 57-69, New York, 1996. Proc. 4th European Conf. on Computer Vision, Springer Verlag.

[167] J. Elder, J. and S. W. Zucker, "Evidence for boundary-specific grouping in human vision", Vision Research, 38(1): 143-152, 1998.

[168] H. Farid and E. P. Simoncelli, "Optimally rotation-equivariant directional derivative kernels", 7th Int'l Conf. on Computer Analysis of Images and Patterns, Kiel, Germany. September 10-12, 1997.

[169] O. Faugeras, "On the motion of 3D curves and its relationship to optical flow", in Proc. ECCV'90 (O. Faugeras. ed.), (Antibes, France), pp. 107-117, Springer-Verlag, April 1990.

[170] O. Faugeras, "Cartan's moving frame method and its applications to the geometry and evolution of curves in the Euclidean, affine and projective planes", Tech. Rep. TR-2053, INRIA, 1993.

[171] O. Faugeras, "Computer vision research at INRIA", Intern. Journal of Computer Vision, vol. 10, no. 2, pp. 91-99, 1993.

[172] O. Faugeras, Three-Dimensional Computer Vision.MIT Press, 1994.

[173] O. Faugeras and R. Keriven, "Affine curvature from affine scale-space", in Proc. Fifth Intern. Conf. on Computer Vision, 1995.

[174] D. Ferster, K. D. Miller, "Neural Mechanisms of Orientation Selectivity in the Visual Cortex", Annual Reviews of Neuroscience, Vol. 23, pp. 441-471, 2000.

[175] M. Fidrich and J.-P. Thirion, "Multiscale representation and analysis of features from medical images", in Intern. Conf. on Computer Vision, Virtual Reality and Robotics in Medicine (N. Ayache, ed.), vol. 905 of LNCS, Nice, pp. 358-364, April 1995.

[176] M. Fidrich and J.-P. Thirion, "Multiscale extraction of features from medical images", in Intern. Conf. on Computer Analysis of Images and Patterns (V. Hlavac and R. S'ara, eds.), vol. 970 of LNCS, (Prague), pp. 637-642, September 1995.

[177] B. Fischer, "Overlap of receptive field centers and representation of the visual field in the cat's optic tract", Vision Research, vol. 13, pp. 2113-2120, 1973.

[178] D. J. Fleet and A. D. Jepson, "Hierarchical construction of orientation and velocity selective filters", PAMI, vol. 11, pp. 315-325, March 1989.

[179] D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information", Intern. Journal of Computer Vision, vol. 5, no. 1, pp. 77-104, 1990.

[180] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Scale and the differential structure of images", Image and Vision Computing, vol. 10, pp. 376-388, 1992.

[181] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "General intensity transformations and second order invariants", in Theory and Applications of Image Analysis (P. Johansen and S. Olsen, eds.), vol. 2 of Series in Machine Perception and Artificial Intelligence, pp. 22-29, Singapore: World Scientific, 1992.

[182] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Cartesian differential invariants in scale-space", Journal of Mathematical Imaging and Vision, vol. 3, pp. 327-348, November 1993.

[183] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Images: Regular tempered distributions", in Proceedings of the NATO Advanced Research Workshop Shape in Picture - Mathematical description of shape in greylevel images (Y.-L. O, A. Toet, H. J. A. M. Heijmans, D. H. Foster, and P. Meer, eds.), vol. 126 of NATO ASI Series F, pp. 651-660, Springer Verlag, Berlin, 1994.

[184] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Linear scale-space", Journal of Mathematical Imaging and Vision, vol. 4, no. 4, pp. 325-351, 1994.

[185] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "General intensity transformations and differential invariants", Journal of Mathematical Imaging and Vision, vol. 4, pp. 171-187, May 1994.

[186] L. M. J. Florack and M. Nielsen, "The intrinsic structure of the optic flow field", ERCIM Technical Report 07/94-R033, 1994.

[187] L. M. J. Florack, A. H. Salden, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Nonlinear scale-space", Image and Vision Computing, vol. 13, pp. 279-294, May 1995.

[188] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "The Gaussian scale-space paradigm and the multiscale local jet", Intern. Journal of Computer Vision, vol. 18, pp. 61-75, April 1996.

[189] L. M. J. Florack, "Data, models and images", in IEEE Intern. Conf. on Image Processing ICIP'96, P. Delogne, ed., Lausanne, CH, pp. 469-472, September 16-19 1996.

[190] L. M. J. Florack, "The concept of a functional integral - a potentially interesting method for image processing", Tech. Rep. 96/7, Institute of Datalogy, University of Copenhagen, 1996.

[191] L. M. J. Florack, "Image structure", Kluwer Academic Publishers, Dordrecht, the Netherlands, 1997.

[192] L. M. J. Florack, "The intrinsic structure of optic flow incorporating measurement duality", Intern. Journal of Computer Vision, vol. 27, no. 3, pp. 263-286, 1998.

[193] L. M. J. Florack, R. Maas, and W. J. Niessen, "Pseudo-linear scale space theory", International Journal of Computer Vision, vol. 31, no. 2/3, pp. 247-259, 1999.

[194] L. M. J. Florack, "A spatio-frequency trade-off scale for scale-space filtering", IEEE Tr. on Pattern Anal. and Mach. Intell. PAMI, vol. 22, no. 9, pp. 1050-1055, 2000.

[195] L. M. J. Florack and A. Kuijper, "The topological structure of scale-space images", Journal of Mathematical Imaging and Vision, Vol. 12, No. 1, pp. 65-79, 2000.

[196] L. M. J. Florack, "Motion extraction - an approach based on duality and gauge theory," in R. Klette, H. S. Stiehl, M. A. Viergever, and K. L. Vincken, eds., Performance Characterization in Computer Vision, vol. 17 of Computational Imaging and Vision Series, Kluwer Academic Publishers, pp. 69-80, 2000.

[197] L. M. J. Florack, "A geometric model for cortical magnification," S.-W. Lee, H. H. Bülthoff, and T. Poggio, eds., Biologically Motivated Computer Vision, vol. 1811 of Lecture Notes in Computer Science, Berlin: Springer-Verlag, pp. 574-583, May 2000.

[198] L. M. J. Florack, "Scale-space theories for scalar and vector images," in M. Kerckhove, ed., Scale-Space and Morphology in Computer Vision: Proceedings of the Third International Conference, Scale-Space 2001, Vancouver, Canada, vol. 2106 of Lecture Notes in Computer Science, Berlin: Springer-Verlag, pp. 193-204, July 2001.

[199] L. M. J. Florack, "Non-linear scale-spaces isomorphic to the linear case with applications to scalar, vector and multispectral images," Journal of Mathematical Imaging and Vision, vol. 15, pp. 39-53, July/October 2001.

[200] M. A. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centers of circular features", in Proc. Intercommission Workshop of the Int. Soc. for Photogrammetry and Remote Sensing, (Interlaken, Switzerland), 1987.

[201] J. B. Fourier, "The Analytical Theory of Heat". New York: Dover Publications, Inc., 1955.Replication of the English translation that first appeared in 1878 with previous corrigenda incorporated into the text, by Alexander Freeman, M. A. Original work: "Theorie Analytique de la Chaleur", Paris, 1822.

[202] R. D. Freeman and I. Ohzawa, "On the neurophysiological organization of binocular vision", Vision Research, vol. 30, no. 11, pp. 1661-1676, 1990.

[203] W. T. Freeman and E. H. Adelson, "Steerable filters for early vision, image analysis and wavelet decomposition", in Proc. 3rd Int. Conf. on Computer Vision, (Osaka, Japan), IEEE Computer Society Press, December 1990.

[204] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 13, pp. 891-906, September 1991.

[205] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition", in Proc. of the IEEE Int. Workshop on Automatic Face and Gesture Recognition, 1995.

[206] D. Fritsch, Registration of Radiotherapy Images Using Multiscale Medial Descriptions of Image Structure. PhD thesis, The University of North Carolina at Chapel Hill, Department of Biomedical Engineering, 1993.

[207] J. Frohlich and J. Weickert, "Image processing using a wavelet algorithm for nonlinear diffusion", Tech. Rep. 104, Laboratory of Technomathematics, Univ. of Kaiserslautern, Germany, March 1994.

[208] R. E. Frye and R. S. Ledley, "Derivative of Gaussian functions as receptive field models for disparity sensitive neurons of the visual cortex", Proceedings of the 1996 Fifteenth Southern Biomedical Engineering Conf., pp. 270-273, 1996.

[209] D. Gabor, "Theory of communications", Journal IEEE London, vol. 93, pp. 429-457, 1946.

[210] M. Gage, "An isoperimetric inequality with applications to curve shortening", Duke Mathematical Journal, vol. 50, pp. 1225-1229, 1983.

[211] M. Gage, "Curve shortening makes convex curves circular", Invent. Math., vol. 76, pp. 357-364, 1984.

[212] M. Gage and R. S. Hamilton, "The heat equation shrinking convex plane curves", Journal of Differential Geometry, vol. 23, pp. 69-96, 1986.

[213] M. Gage, "On an area-preserving evolution equation for plane curves", Contemporary Mathematics, vol. 51, pp. 51-62, 1986.

[214] E. B. Gamble and T. Poggio, "Visual integration and detection of discontinuities: The key role of intensity edges", tech. rep., MIT A.I. Lab, 1987. A.I. Memo No. 970.

[215] J. Gårding, "Shape from texture for smooth curved surfaces in perspective projection", Journal of Mathematical Imaging and Vision, vol. 2, pp. 329-352, 1992.

[216] J. Gårding and T. Lindeberg, "Direct computation of shape cues by multi-scale retinotopic processing." submitted, 1993.

[217] J. Gårding and T. Lindeberg, "Direct estimation of local surface shape in a fixating binocular vision system", in Proc. 3rd European Conf. on Computer Vision (J.-O. Eklundh, ed.), vol. 800 of Lecture Notes in Computer Science, (Stockholm, Sweden), pp. 365-376, Springer-Verlag, May 1994.

[218] J. Gårding and T. Lindeberg, "Direct computation of shape cues based on scale-adapted spatial derivative operators", Intern. Journal of Computer Vision, vol. 17, no. 2, pp. 163-192, 1996.

[219] J. L. Gardner, A. Anzai, I. Ohzawa, and R. D. Freeman, "Linear and nonlinear contributions to orientation tuning of simple cells in the cat's striate cortex". Visual Neuroscience 16: 1115-1121, 1999.

[220] L. J. Garey (Ed.): "Brodmann's localisation in the cerebral cortex", Smith Gordo & Co, 1992.

[221] J.M. Gauch, "Image Segmentation and Analysis via Multiscale Gradient Watershed Hierachies", IEEE Transactions on Image Processing, 8(1):69-79, January 1999.

[222] D. Geiger and A. Yuille, "A common framework for image segmentation", Intern. Journal of Computer Vision, vol. 6, no. 3, pp. 227-243, 1991.

[223] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 6, pp. 721-741, 1984.

[224] R. Geraets, A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Affine scale-space for discrete pointsets", in Proc. Soc. for Neural Networks (C. Gielen, ed.), Nijmegen, the Netherlands, SNN, 1995.

[225] R. Geraets, A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Object recognition by affine evolution of measured interest points", in Proc. Computing Science in the Netherlands, Utrecht, the Netherlands, pp. 86-97, SION, 1995.

[226] G. Gerig, O. K"ubler, R. Kikinis, and F. A. Jolesz, "Nonlinear anisotropic filtering of MRI data", Journal of Mathematical Imaging and Vision, vol. 11, pp. 221-232, June 1992.

[227] G. Gerig, G. Szekely, and T. Koller, "Line-finding in 2-D and 3-D by multi-valued non-linear diffusion of feature maps", in DAGM Symposium, Informatik aktuell (S. J. Poeppl and H. Handels, eds.), pp. 289-296, Springer-Verlag, 1993.Mustererkennung 1993, 15.

[228] G. Gerig, G. Sz'ekely, G. Israel, and M. Berger. "Detection and characterization of unsharp blobs by curve evolution", in Proc. Information Processing in Medical Imaging (IPMI'95) (Y. B. et al., ed.), Series on Computational Imaging and Vision, pp. 165-176, Kluwer Academic Publishers, June 1995.

[229] H. J. M. Gerrits, B. de Haan, and A. J. H. Vendrik, "Experiments with retinal stabilized images. relations between the observations and neural data", Vision Research, vol. 6, pp. 427-440, 1966.

[230] J. M. Geusebroek, A. Dev, R. van den Boomgaard, A. W. M. Smeulders. F. Cornelissen and H. Geerts, "Color Invariant edge detection". In: Scale-Space theories in Computer Vision, Lecture Notes in Computer Science, vol. 1252, pp. 459-464, Springer-Verlag, 1999.

[231] J. M. Geusebroek, R. van den Boomgaard, A. W. M. Smeulders and A. Dev. "Color and scale: the spatial structure of color images". In: Eur. Conf. on Computer Vision 2000. Lecture Notes in Computer Science, Vol. 1842, Springer, pp. 331-341, June 26 - July 1, 2000.

[232] J. M. Geusebroek, A. W. M. Smeulders and R. van den Boomgaard, "Measurement of Color Invariants". Proc. CVPR, vol. 1, pp. 50-57, June 13-15, 2000.

[233] J. M. Geusebroek, D. Koelma, A. W. M. Smeulders and Th. Gevers. "Image Retrieval and Segmentation based on Color Invariants". Proc. CVPR, June 13-15, 2000.

[234] T. Gevers and A. W. Smeulders. "Color based object recognition". Patt. Recogn. 32, 453-464, 1999.

[235] G. M. Ghose, I. Ohzawa, and R. D. Freeman, "Receptive field maps of correlated discharge between pairs of neurons in the cat's visual cortex", Journal of Neurophysiology, vol. 71, no. 1, pp. 330-346, 1994.

[236] J. Gibbon and R. M. Church, "Time-left: linear versus logarithmic subjective time". Journal of Experimental Psychology: Animal Behavior Processes, vol. 7, pp. 87-108, 1981.

[237] J. J. Gibson, The Perception of the Visual World. Boston: Houghton-Mifflin. 1950.

[238] B. Gillam and B. Rogers, "Orientation disparity, deformation, and stereoscopic slant perception", Perception, vol. 20, pp. 441-448, 1991.

[239] R. Gilmore, Catastrophe theory for scientists and engineers. New York: Wiley-Interscience, 1981.

[240] G. H. Golub and C. F. Van Loan, Matrix Computations. Baltimore: The Johns Hopkins University Press, 1989. Second Edition.

[241] Goodchild, A.K., K.K. Ghosh, and P.R. Martin. "Comparison of photoreceptor spatial density and ganglion cell morphology in the retina of human, macaque monkey, cat, and the marmoset Callithrix jacchus". Journal of Comparative. Neurology. 366 : 55-75, 1996.

[242] C. d. Graaf, S. M. Pizer, A. Toet, J. J. Koenderink, and P. P. van Rijk. "Pyramid segmentation of medical 3D images", in Proc. of the 1984 Int. Joint Alpine Symposium, pp. 71-77, IEEE, 1984.

[243] N. Graham, "The visual system does a crude Fourier analysis of patterns", in SIAM-AMS Proceedings (S. Grossberg, ed.), vol. 13, (Hillsdale, New Jersey), pp. 1-16, American Mathematical Society, Lawrence Erlbaum Associates, 1981.

[244] A. Gray, "Modern differential geometry of curves and surfaces". CRC Press Inc., Boca Raton, 1993, second edition 1997.

[245] M. Grayson, "The heat equation shrinks embedded plane curves to round points", Journal of Differential Geometry, vol. 26, pp. 285-314, 1987.

[246] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, and C. H. Anderson, "Overcomplete steerable pyramid filters and rotation invariance", in Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition. CVPR'94, pp. 222-228, IEEE, 1994.

[247] W. E. L. Grimson, "From images to surfaces". Cambridge MA: MIT Press, 1981.

[248] S. Grossberg and D. Todorovic, "Neural dynamics of 1-D and 2-D brightness perception: A unified model of classical and recent phenomena", Perception and Psychophysics, vol. 43, pp. 241-277, 1988.

[249] F. Guichard, "Multiscale analysis of movies", in Eighth Workshop on Image and Multidimensional Image Processing, (Cannes), pp. 236-237, IEEE, September 8-10 1993.

[250] A. Guiducci, "Corner characterization by differential geometry techniques", Pattern Recognition Letters, vol. 8, pp. 311-318, 1988.

[251] R. W. Guillery, "A quantitative study of synaptic interconnections in the dorsal lateral geniculate nucleaus in the cat", Zeitschrift für Zellforschung, vol. 96, pp. 39-48, 1969.

[252] R. W. Guillery, "The organization of synaptic interconnections in the laminae of the dorsal lateral geniculate nucleaus in the cat", Zeitschrift für Zellforschung, vol. 96, pp. 1-38, 1969.

[253] R. W. Guillery, "Patterns of synaptic interconnections in the dorsal lateral geniculate nucleaus of cat and monkey: A brief review", Vision Research (Suppl.), vol. 3, pp. 211-227, 1971.

[254] M. M. Gupta and G. K. Knopf, eds., "Neuro-vision systems, principles and applications". A selected reprint volume, IEEE Press, New York, 1993.

[255] B. Gurevich, "Foundations of the theory of algebraic invariants". Groningen: P. Noordhoff, 1979.

[256] W. Hackbush, "Multi-grid methods and applications". New York: Springer-Verlag, 1985.

[257] J. S. Hadamard, "Sur les problèmes aux derivées partielles el leur signification physique." Bull. Univ. Princeton, vol. 13, pp. 49-62, 1902.

[258] T. Hall, "Carl Friedrich Gauss, a biography". M.I.T. Press, Cambridge, 1970.

[259] R. M. Haralick, "Zero-crossing of second directional derivative edge operator", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 6, pp. 58-68, 1984.

[260] H. K. Hartline, "The receptive fields of optic nerve fibers", American Joernal of Physiology, vol. 130, pp. 690-699, 1940.

[261] M. Hashimoto and J. Sklansky, "Multiple order derivatives for detecting local image characteristics", Computer Vision, Graphics, and Image Processing, vol. 39, pp. 28-55, 1987.

[262] D. J. Heeger, "Optical flow using spatiotemporal filters", Intern. Journal of Computer Vision, vol. 1, pp. 279-302, 1988.

[263] H. J. A. M. Heijmans, "Mathematical morphology: a geometrical approach in image processing", Nieuw Archief voor Wiskunde, vol. 10, pp. 237-276, November 1992.

[264] F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, and O. Kübler, "Simulation of neural contour mechanisms: from simple to end-stopped cells", Vision Research, vol. 32, no. 5, pp. 963-981, 1992.

[265] E. Hering. "Outlines of a theory of the light sense". Harvard University Press, Cambridge, 1964.

[266] D. Hilbert, "Über die Theorie der algebraischen Formen", Math. Annalen, vol. 36, pp. 473-534, 1890.

[267] D. Hilbert, "Über die vollen Invariantensystemen", Math. Annalen, vol. 42, pp. 313-373, 1893.

[268] H. Hildreth, "The detection of intensity changes by computer and biological visual systems", Computer Vision, Graphics, and Image Processing, vol. 22, pp. 1-27, 1983.

[269] E. Hildreth, "The measurement of visual motion".Cambridge, Mass.: M. I. T. Press, 1983.

[270] E. C. Hildreth, "Computations underlying the measurement of visual motion", Artificial Intelligence, vol. 23, pp. 309-354, 1984.

[271] B. K. P. Horn and B. Schunck, "Determining optic flow". Artificial Intelligence, vol. 23, pp. 185-203, 1981.

[272] B. K. P. Horn, "Robot Vision". Cambridge MA: MIT Press, 1986.

[273] B. Horn and M. Brooks, eds., "Shape from shading".Cambridge. Mass.: M. I. T. Press, 1989.

[274] D. H. Hubel and T. N. Wiesel, "Receptive fields. binocular interaction, and functional architecture in the cat's visual cortex", Journal of Physiology, vol. 160, pp. 106-154, 1962.

[275] D. H. Hubel and T. N. Wiesel, "Brain mechanisms of vision", Scientific American, vol. 241, pp. 45-53, 1979.

[276] D. H. Hubel, "Eye, brain and vision", vol. 22 of Scientific American Library. New York: Scientific American Press, 1988.

[277] R. A. Hummel, B. B. Kimia, and S. W. Zucker, "Gaussian blur and the heat equation: Forward and inverse solutions", in Proc. CVPR, pp. 668-671, 1985.

[278] R. A. Hummel and D. Lowe, "Computing Gaussian blur", in Proc. ICPR 1986, pp. 910-912, 1986.

[279] R. A. Hummel, B. B. Kimia, and S. W. Zucker, "Deblurring Gaussian blur", Computer Vision, Graphics, and Image Processing, vol. 38, pp. 66-80, 1987.

[280] R. A. Hummel, "The scale-space formulation of pyramid data structures", in Parallel Computer Vision (L. Uhr, ed.), pp. 187-123, Academic Press, New York, 1987.

[281] R. A. Hummel, "Representations based on zero crossings in scale-space", in Proc. IEEE Computer Vision and Pattern Recognition Conf., pp. 204-209, June 1986. Also in: "Readings in Computer Vision: Issues, Problems, Principles and Paradigms", M. Fischler and O. Firschein (eds.), Morgan Kaufmann, 1987.

[282] R. A. Hummel and R. Moniot, "Reconstructions from zero-crossings in scale-space", IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 37, no. 12, pp. 2111-2130, 1989.

[283] T. Iijima, "Basic theory on the normalization of a pattern", Bulletin of Electrical Laboratory, vol. 26, pp. 368-388, 1962.In Japanese.

[284] T. Iijima, "Basic equation of figure and observational transformation", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-C, no. 9, pp. 37-38, 1971. English Abstracts.

[285] T. Iijima, "Basic theory on the construction of figure space", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-C, no. 8, pp. 35-36, 1971. English Abstracts.

[286] T. Iijima, "A suppression kernel of a figure and its mathematical characteristics", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-C, no. 9, pp. 30-31, 1971. English Abstracts.

[287] T. Iijima, "Basic theory on normalization of figure", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-C, no. 11, pp. 24-25. 1971. English Abstracts.

[288] T. Iijima, "A system of fundamental functions in an abstract figure space", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-C, no. 11, pp. 26-27, 1971. English Abstracts.

[289] T. Iijima, "Basic theory on feature extraction of figures", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-C, no. 12, pp. 35-36, 1971. English Abstracts.

[290] T. Iijima, "Basic theory on the structural recognition of a figure", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 55-D, no. 8, pp. 28-29, 1972. English Abstracts.

[291] T. Iijima, "Theoretical studies on the figure identification by pattern matching", Tr. of the Institute of Electronics and Communication Engineers of Japan, vol. 54-D, no. 8, pp. 29-30, 1972. English Abstracts.

[292] T. Jackway, "Morphological scale-space", in Proc. 11th Int. Conf. Pattern Recognition (ICPR 11), vol. C, (The Hague), pp. 252-255, Aug. 30 - Sept. 3 1992.

[293] T. Jackway and M. Deriche, "Scale-space properties of multiscale morphological dilation-erosion", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 18, no. 1, pp. 38-51, 1996.

[294] P. T. Jackway and M. Deriche, "Scale-space properties of the multiscale morphological dilation-erosion", IEEE PAMI, vol. 18, no. 1, pp. 38-51, 1996.

[295] M. Jägersand, "Saliency maps and attention selection in scale and spatial coordinates: An information theoretic approach", in Proc. Fifth Intern. Conf. on Computer Vision (E. Grimson, S. Shafer, A. Blake, and K. Sugihara, eds.), (MIT Cambridge, MA), pp. 195-202, IEEE, June 20-23 1995.

[296] E. T. Jaynes, "Prior probabilities", Proc. IEEE SSC-4, pp. 227-241, 1968.

[297] A. D. Jepson and D. J. Fleet, "Scale-space singularities", in Proc. first European Conf. on Computer Vision (O. Faugeras, ed.), (Berlin), pp. 50-56, Springer-Verlag, 1990. Lecture Notes in Computer Science.

[298] P. Johansen, S. Skelboe, K. Grue, and J. D. Andersen, "Representing signals by their top points in scale-space", in Proceedings of the 8-th Intern. Conf. on Pattern Recognition, (Paris), pp. 215-217, October 27-31 1986.

[299] Johansen, "On the classification of toppoints in scale-space", Journal of Mathematical Imaging and Vision, vol. 4, no. 1, pp. 57-68, 1994.

[300] J.-J. Jolion and A. Rozenfeld, "A pyramid framework for early vision". Dordrecht, Netherlands: Kluwer Academic Publishers, 1994.

[301] J. P. Jones and L. A. Palmer, "The two-dimensional spatial structure of simple receptive fields in cat striate cortex", Journal of Neurophysiology, vol. 58, pp. 1187-1211, 1987.

[302] G. J. Jones and J. Malik, "A computational framework for determining stereo correspondence from a set of linear spatial filters", Image and Vision Computing, vol. 10, no. 10, pp. 699-708, 1992.

[303] D. B. Judd and G. Wyszecki. "Color in business, science and industry". Wiley, New York, NY, 1975.

[304] J. Kacur, K. Mikula, "Slow and fast diffusion effects in image processing", Computing and Visualization in Science, Springer Berlin, vol. 3, nr. 4, 2001.

[305] S. N. Kalitzin, B. M. ter Haar Romeny, A. H. Salden, P. F. M. Nacken, and M. A. Viergever, "Topological numbers and singularities in scalar images. scale-space evolution properties". Journal of Mathematical Imaging and Vision, vol. 7, 1996.

[306] S. N. Kalitzin, B. M. ter Haar Romeny, and M. A. Viergever, "Invertible orientation bundles on 2D scalar images", in Proc. First Intern. Conf. on Scale-Space Theory in Computer Vision, Lecture Notes in Computer Science, (Utrecht, the Netherlands), pp. 77-88, Springer Verlag, July 1997.

[307] S. N. Kalitzin, B. M. ter Haar Romeny, and M. A. Viergever, "On topological deep-structure segmentation", in Proc. Intern. Conf. on Image Processing, (Santa Barbara, California), pp. 863-866, October 26-29 1997.

[308] S. N. Kalitzin, "Topological numbers and singularities in scalar images. scale-space evolution properties", in Gaussian Scale-Space Theory (Jon Sporring, Mads Nielsen, Luck Florack and Peter Johansen, Eds.), pp. 181-189, Kluwer Academic Publishers, 1997.

[309] S. N. Kalitzin, B. M. ter Haar Romeny, and M. A. Viergever, "Invertible apertured orientation filters in image analysis". Intern. J. of Computer Vision, vol. 31, no. 2/3, pp. 145-158, 1998.

[310] K. Kanatani, "Group-theoretical methods in image understanding", vol. 20 of Series in Information Sciences. Springer-Verlag, 1990.

[311] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, "Principles of Neural Science". McGraw-Hill Companies, New York, fourth edition, 2000.

[312] D. Kapur and J. L. Mundy, "Geometric Reasoning". MIT Press, 1995.

[313] N. Karssemeijer, "Detection of stellate distortions in mammograms using scale-space operators", in Proc. Information Processing in Medical Imaging, pp. 335-346, 1995.

[314] N. Karssemeijer and G. te Brake, "Detection of stellate distortions in mammograms", IEEE Tr. Medical Imaging, vol. 15, no. 5, pp. 611-619, 1996.

[315] M. Kass and A. Witkin, "Analyzing oriented patterns", Computer Vision, Graphics, And Image Processing, vol. 37, pp. 362-385, 1985.

[316] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models", Intern. Journal of Computer Vision, vol. 1, no. 4, pp. 321-331, 1988.

[317] T. P. Kaushal, "Towards visually convincing image segmentation", Image and Vision Computing, vol. 10, pp. 617-624, November 1992.

[318] R. G. Kessel and R. H. Kardon, "Tissues and organs, a text atlas of scanning electron microscopy", W. H. Freeman and Company, San Francisco and Oxford, 1979.

[319] S. Kichenassamy, "Nonlinear diffusions and hyperbolic smoothing for edge enhancement", in Proc. of 12th Intern. Conf. on Analysis and Optimization of Systems (M. O. Berger, R. Deriche, I. Herlin, J. Jaffré, and J. M.

Morel, eds.), vol. 219 of Lecture Notes in Control and Information Sciences, pp. 119-124, Springer, London, 1996.

[320] B. B. Kimia, "Deblurring Gaussian blur, continuous and discrete approaches", Master's thesis, McGill University, Electrical Eng. Dept., Montreal, Canada, 1986.

[321] B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "Towards a computational theory of shape, an overview", in Proc. first European Conf. on Computer Vision, vol. 427 of Lecture Notes in Computer Science, (New York), pp. 402-407, Springer-Verlag, 1990.

[322] B. B. Kimia, "Entropy scale-space", in Proc. of Visual Form Workshop, (Capri, Italy), Plenum Press, May 1991.

[323] B. B. Kimia, A. Tannenbaum, and S. W. Zucker, "On the evolution of curves via a function of curvature I: the classical case", Journal of Mathematical Analysis and Applications, vol. 163, pp. 438-458, 1992.

[324] B. B. Kimia and S. W. Zucker, "Analytic inverse of discrete Gaussian blur", Optical Engineering, vol. 32, no. 1, pp. 166-176, 1986.

[325] B. B. Kimia and K. Siddiqi, "Geometric heat equation and nonlinear diffusion of shapes and images", Computer Vision and Image Understanding, vol. 64, pp. 305-322, 1996.

[326] L. Kitchen and A. Rosenfeld, "Gray-level corner detection", Pattern Recognition Letters, vol. 1, pp. 95-102, 1982.

[327] F. Klein, "Erlanger Programm", Math. Annalen, vol. 43, pp. 63-100. 1893.

[328] C. B. Knudsen and H. I. Christensen, "On methods for efficient pyramid generation", in Proc. 7th Scand. Conf. on Image Analysis, (Aalborg, Denmark), pp. 28-39, August 1991.

[329] H. Kobayashi, L. White, Joseph, and A. Abidi, Asad, "An active resistor network for Gaussian filtering of images", IEEE Journal of Solid-State Circuits, vol. 26, pp. 738-748, May 1991.

[330] J. J. Koenderink and A. J. van Doorn, "Geometry of binocular vision and a model for stereopsis", Biological Cybernetics, vol. 21, pp. 29-35, 1976.

[331] J. J. Koenderink and A. J. van Doorn, "Visual detection of spatial contrast; influence of location in the visual field, target extent and illuminance level", Biological Cybernetics, vol. 30, pp. 157-167, 1978.

[332] J. J. Koenderink and A. J. van Doorn, "The structure of two-dimensional scalar fields with applications to vision", Biological Cybernetics, vol. 33, pp. 151-158, 1979.

[333] J. J. Koenderink and A. J. van Doorn, "The internal representation of solid shape with respect to vision", Biological Cybernetics, no. 32, pp. 211-216, 1979.

[334] J. J. Koenderink and A. J. van Doorn, "Photometric invariants related to solid shape", Optica Acta, vol. 27, pp. 981-996, 1980.

[335] J. J. Koenderink and A. J. van Doorn, "A description of the structure of visual images in terms of an ordered hierarchy of light and dark blobs", in Second Int. Visual Psychophysics and Medical Imaging Conf., 1981.IEEE Cat. No. 81 CH 1676-6.

[336] J. J. Koenderink, "The structure of images", Biological Cybernetics, vol. 50, pp. 363-370, 1984.

[337] J. J. Koenderink, "Simultaneous order in nervous nets from a functional standpoint", Biological Cybernetics, vol. 50, pp. 35-41, 1984.

[338] J. J. Koenderink, "Geometrical structures determined by the functional order in nervous nets", Biological Cybernetics, vol. 50, pp. 43-50, 1984.

[339] J. J. Koenderink, "The concept of local sign", in Limits in Perception (A. J. van Doorn, W. A. van de Grind, and J. J. Koenderink, eds.), pp. 495-547, Utrecht: VNU Science Press. 1984.

[340] J. J. Koenderink, A. J. van Doorn, and W. A. van de Grind, "Spatial and temporal parameters of motion detection in the peripheral visual field", Journal of the Optical Society of America-A, vol. 2, pp. 252-259, February 1985.

[341] J. J. Koenderink, "The structure of the visual field", in The Physics of Structure Formation. Theory and Simulation (W. Guettinger and G. Dangelmayr, eds.), Springer-Verlag, 1986. Proceedings of an Intern. Symposium, Tuebingen, Fed. Rep. of Germany, October 27-November 2.

[342] J. J. Koenderink and A. J. van Doorn, "Dynamic shape", Biological Cybernetics, vol. 53, pp. 383-396, 1986.

[343] J. J. Koenderink, "Optic flow", Vision Research, vol. 1, pp. 161-180, 1986.

[344] J. J. Koenderink, "Image structure", in Mathematics and Computer Science in Medical Imaging (M. A. Viergever and A. Todd-Pokropek, eds.), (Berlin), Springer-Verlag, 1986. Proceedings of the NATO Advanced Study Institute of Mathematics and Computer Science in Medical Imaging, held in Il Ciocco, Italy, September 21 - October 4.

[345] J. J. Koenderink and A. J. van Doorn, "Representation of local geometry in the visual system", Biological Cybernetics, vol. 55, pp. 367-375, 1987.

[346] J. J. Koenderink, "Design principles for a front-end visual system", in Neural Computers (R. Eckmiller and C. v. d. Malsburg, eds.), Springer-Verlag, 1987. Proceedings of the NATO Advanced Research Workshop on Neural Computers, held in Neuss, Fed. Rep. of Germany, September 28-October 2.

[347] J. J. Koenderink, "An internal representation for solid shape based on the topological properties of the apparent contour", in Image Understanding (J. Richards and S. Ullman, eds.), pp. 257-285, Norwood, New Jersey: Alex Publishing Corporation, 1987.

[348] J. J. Koenderink and A. J. van Doorn, "Facts on optic flow", Biological Cybernetics, vol. 56, pp. 247-254, 1987.

[349] J. J. Koenderink, "Image structure", in Mathematics and Computer Science in Medical Imaging (Viergever/Todd-Pokropek, ed.), pp. 67-104, Springer-Verlag, 1988. NATO ASI F39.

[350] J. J. Koenderink: "Scale-Time", Biological Cybernetics, vol. 58, pp. 159-162, 1988.

[351] J. J. Koenderink and A. J. Van Doorn, "Operational significance of receptive field assemblies", Biological Cybernetics, vol. 58, pp. 163-171, 1988.

[352] J. J. Koenderink and W. Richards, "Two-dimensional curvature operators", Journal of the Optical Society of America-A, vol. 5, no. 7, pp. 1136-1141, 1988.

[353] J. J. Koenderink and A. J. van Doorn, "The basic geometry of a vision system", pp. 481-485. Kluwer Academic Publishers, 1988. Trappl, R. (Ed.).

[354] J. J. Koenderink, "Design for a sensorium", pp. 185-207. D-6940 Weinheim, Federal Republic of Germany: VCH Verlagsgesellschaft mbH, 1988. Editors: von Seelen, Werner and Shaw, Gordon and Leinhos, Ulrich M.

[355] J. J. Koenderink, "A hitherto unnoticed singularity of scale-space", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 11, no. 11, pp. 1222-1224, 1989.

[356] J. J. Koenderink, "Solid Shape". Cambridge, Mass.: MIT Press, 1990.

[357] J. J. Koenderink and A. J. van Doorn, "Receptive field families", Biological Cybernetics, vol. 63, pp. 291-298, 1990.

[358] J. J. Koenderink, "The brain a geometry engine", Psychological Research, vol. 52, pp. 122-127, 1990.

[359] J. J. Koenderink, "Perception and control of self-motion". In: Some theoretical aspects of optic flow, pp. 53-68. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc., 1990.R. Warren, Ed.

[360] J. J. Koenderink and A. J. van Doorn, "Advances in neural computers", chapter in: Receptive field taxonomy. Elsevier, 1990.R. Eckmuller, Ed.

[361] J. J. Koenderink, "Mapping formal structures on networks", pp. 93-98. North-Holland: Elsevier Science Publishers B. V., 1991. Eds.: Kohonen, T. and Mäkisara, K. and Simula, O. and Kangas, J.

[362] J. J. Koenderink and A. J. van Doorn, "Affine structure from motion", Journal of the Optical Society of America-A, vol. 8, no. 2, pp. 377-385, 1991.

[363] J. J. Koenderink, "Local image structure", in Proc. Scand. Conf. on Image Analysis, (Aalborg, DK), pp. 1-7, August 1991.

[364] J. J. Koenderink and A. J. van Doorn, "Receptive field assembly pattern specifity", J. of Vis. Comm. and Im. Repr., vol. 3, no. 1, pp. 1-12, 1991.

[365] J. J. Koenderink and A. J. van Doorn, "Generic neighborhood operators", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 14, pp. 597-605, June 1992.

[366] J. J. Koenderink, "Local image structure", in Theory and Applications of Image Analysis (P. Johansen and S. Olsen, eds.), vol. 2 of Series in Machine Perception and Artificial Intelligence, pp. 15-21, Singapore: World Scientific, 1992.

[367] J. J. Koenderink, Fundamentals of Bicentric Perspective, vol. 653 of Lecture Notes in Computer Science, pp. 233-251. Heidelberg Berlin: Springer Verlag, 1992. Bensoussan A. and Verjus J. P. (Eds.).

[368] J. J. Koenderink, A. Kappers, and A. van Doorn, "Local operations: The embodiment of geometry", in Artificial and Biological Vision Systems (G. A. Orban and H. H. Nagel, eds.), ESPRIT: Basic Research Series, pp. 1-23, DG XIII Commision of the European Communities, 1992.

[369] J. J. Koenderink, "Iseikonic invariants in bicentric perspective", Tech. Rep. UBI-T-92.MF-058, Utrecht Biophysics Research Institute, Division: Medical and Physiological Physics, Buys Ballot Laboratory, Utrecht University, 1992.

[370] J. J. Koenderink and A. J. van Doorn, "Second order optic flow", Journal of the Optical Society of America-A, vol. 8, no. 2, pp. 530-538, 1992.

[371] J. J. Koenderink and A. J. van Doorn, "Surface shape and curvature scales", Image & Vision Computing, vol. 10, pp. 557-565, 1992.

[372] J. J. Koenderink and A. J. van Doorn, "Local features of smooth shapes: Ridges and courses", in Proc. SPIE Geometric Methods in Computer Vision II, vol. 2031, (San Diego, CA), pp. 2-13, Proceedings SPIE, July, 12-13 1993.

[373] J. J. Koenderink and A. J. van Doorn, "Two-plus-one dimensional differential geometry", Pattern Recognition Letters, vol. 21, pp. 439-443, May 1994.

[374] J. J. Koenderink and A. J. van Doorn. "Illuminance texture due to surface mesostructure". J. Opt. Soc. Am. A, vol. 13, no. 3, pp. 452-463, 1996.

[375] J. J. Koenderink and A. J. van Doorn. "Metamerism in complete sets of image operators". In: Advances in Image Understanding, Bowyer K., Ahuja N. (eds.), IEEE Computer Society Press, Los Alamitos, California, 113-129, 1996.

[376] J. J. Koenderink, A. J. van Doorn, C. Christou and J. S. Lappin, "Shape constancy in pictorial relief". Perception, vol. 25, pp. 155-164, 1996.

[377] J. J. Koenderink, A. J. van Doorn, C. Christou and J. S. Lappin, "Perturbation study of shading in pictures". Perception, vol. 25, pp. 1009-1026, 1996.

[378] J. J. Koenderink, A. J. van Doorn and A. M. L. Kappers, "Pictorial surface attitude and local depth comparisons". Perception & Psychophysics, vol. 58, no. 2, pp. 163-173, 1996.

[379] J. J. Koenderink, A. J. van Doorn and M/ Stavridi, "Bidirectional reflection distribution function expressed in terms of surface scattering modes". In: Proc. Europ. Conf. on Computer Vision - ECCV '96, B. Buxton, R. Cipolla (eds.), Springer Verlag, Berlin, pp. 28-39, 1996.

[380] J. J. Koenderink, A. M. L. Kappers, J. T. Todd, J. F. Norman and F. Phillips, "Surface range and attitude probing in stereoscopically presented dynamic scenes". Journal of Experimental Psychology: Human Perception and Performance, vol. 22, pp. 869-878, 1996.

[381] J. J. Koenderink, "Scale in perspective", in Gaussian Scale-Space, Kluwer Academic Press, 1997. Sporring, J. et al. (eds.).

[382] J. J. Koenderink and A. J. van Doorn, "The generic bilinear calibration-estimation problem". Intern. Journal of Computer Vision, vol. 23, no. 3, pp. 217-234, 1997.

[383] J. J. Koenderink, A. J. van Doorn, A. M. L. Kappers and J. Todd, "The visual contour in depth". Perception & Psychophysics, vol. 59, no. 6, pp. 828-838, 1997.

[384] J. J. Koenderink, A. M. L. Kappers, F. E. Pollick and M. Kawato, "Correspondence in pictorial space". Perception & Psychophysics, vol. 59, no. 6, pp. 813-827, 1997.

[385] J. J. Koenderink, "Receptive field calculus". In: Progress in Neural Networks, Vol. 4: Machine Vision, Omidvar O.M., Mohan R. (eds.), Ablex Publishing Corporation, Greenwich, Connecticut, pp. 1-28, 1997.

[386] J. J. Koenderink, "Pictorial relief". In: Advances in visual form analysis, Arcelli C., Cordella L.P., Sanniti di Baja G. (eds.), World Scientific, Singapore, pp. 308-317, 1997.

[387] J. J. Koenderink and A. J. van Doorn, "Image structure". In: Mustererkennung 1997, Paulus E., Wahl F.M. (eds.), Springer-Verlag, Berlin, pp. 3-35, 1997.

[388] J. J. Koenderink and A. J. van Doorn, "Local image operators and iconic structure". In: Algebraic frames for the perception-action cycle, Sommer G., Koenderink J.J. (eds.), Springer-Verlag, Berlin, 66-93, 1997.

[389] J. J. Koenderink, "Color Space". Utrecht University, the Netherlands, 1998.

[390] Koenderink J.J., Doorn A.J. van, "The structure of locally orderless images". Intern. Journal of Computer Vision, vol. 31, no. 2/3, pp. 159-168, 1999.

[391] J. J. Koenderink and A. J. van Doorn, "Blur and disorder". In: Scale-space theories in computer vision, M. Nielsen, P. Johansen, O. F. Olsen and J. Weickert (eds.), Lecture Notes in Computer Science, Vol. 1682, Springer, Berlin, pp. 1-9, 1999.

[392] J. J. Koenderink and A. J. van Doorn, "The structure of colorimetry". In: Algabraic frames for the perception-action cycle, Sommer G., Zeevi Y.Y. (eds.), Springer, Berlin, pp. 69-77, 2000.

[393] G. Koepfler, C. Lopez, and J. M. Morel, "A multiscale algorithm for image segmentation by variational method", SIAM Journal on Numerical Analysis, 1994.

[394] A. S. E. Koster, K. L. Vincken, C. N. De Graaf, O. C. Zander, and M. A. Viergever, "Heuristic linking models in multiscale image segmentation", Computer Vision and Image Understanding, vol. 65, no. 3, pp. 382-402, 1997.

[395] A. Kuijper, L. M. J. Florack, "Calculations on critical points under Gaussian blurring", in Proc. 2nd Intern. Conf. on Scale-Space Theory in Computer Vision (Corfu, Greece), Lecture Notes in Computer Vision, vol. 1682, pp. 318-329, 1999.

[396] A. Kuijper, L. M. J. Florack, "Hierarchical pre-segmentation without prior knowledge", in Proc. 8th IEEE Intern. Conf. on Computer Vision (Vancouver CA), pp. 487-493, 2001.

[397] A. Kuijper, L. M. J. Florack, "Understanding and modeling the evolution of critical points under Gaussian blurring", in Proc. 7th Eur. Conf. on Computer Vision (Copenhagen DK), Lecture Notes in Computer Vision, vol. 2350, pp. 143-157, 2001.

[398] A. Kuijper, L. M. J. Florack, "The relevance of non-generic events in scale-space models", in Proc. 7th Eur. Conf. on Computer Vision (Copenhagen DK), Lecture Notes in Computer Vision, vol. 2350, pp. 190-204, 2001.

[399] A. Kuijper, L. M. J. Florack, "Scale-space hierarchy", J. of Math. Imaging and Vision, 2002.

[400] A. Kuijper, "The deep structure of Gaussian scale-space images", PhD thesis, Utrecht University, 2002.

[401] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, A. Baumgartner, "Automatic extraction of roads from aerial images based on scale-space and snakes", Machine Vision and Applications, 2000.

[402] T. S. Lee, "Image representation using 2D Gabor wavelets", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, pp. 959-971, 1996.

[403] R. Lenz, "Group theoretical methods in image processing", vol. 413 of Lecture Notes in Computer Science, Goos, G. and Hartmanis, J. (Eds.), Berlin: Springer Verlag, 1990.

[404] W. R. Levick, "Sampling of information space by retinal ganglion cells", in Visual Neuroscience (J. D. Pettigrew, K. J. Sanderson, and W. R. Levick, eds.), ch. 3, pp. 33-43, Cambridge University Press, 1986.

[405] Z. Li and J. J. Atick, "Towards a theory of striate cortex", Neural Computation, vol. 6, pp. 127-146, 1994.

[406] X. Li and T. Chen, "Optimal L_1 approximation of the Gaussian kernel with application to scale-space construction", IEEE Tr. Patter Anal. and Machine Intell., vol. 17, no. 10, pp. 1015-1019, 1995.

[407] L. M. Lifshitz and S. M. Pizer, "A multiresolution hierarchical approach to image segmentation based on intensity extrema", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 6, pp. 529-541, 1990.

[408] T. Lindeberg, "Scale-space for discrete signals", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 3, pp. 234-245, 1990.

[409] T. Lindeberg and J. O. Eklundh, "Scale detection and region extraction from a scale-space primal sketch", in Proc. 3rd Int. Conf. on Computer Vision, (Osaka, Japan), pp. 416-426, December 1990.

[410] T. Lindeberg, "Discrete scale-space theory and the scale-space primal sketch". PhD thesis, Royal Institute of Technology, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden, May 1991.

[411] T. Lindeberg and J. O. Eklundh, "On the computation of a scale-space primal sketch", Journal of Visual Comm. and Image Rep., vol. 2, pp. 55-78, 1991.

[412] T. Lindeberg, "Scale-space behaviour of local extrema and blobs", Journal of Mathematical Imaging and Vision, vol. 1, pp. 65-99, March 1992.

[413] T. Lindeberg, "On the behaviour in scale-space of local extrema and blobs", in Theory and Applications of Image Analysis (P. Johansen and S. Olsen, eds.), vol. 2 of Series in Machine Perception and Artificial Intelligence, pp. 38-47, Singapore: World Scientific, 1992.

[414] T. Lindeberg, "Effective scale: A natural unit for measuring scale-space lifetime", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 15, October 1993.

[415] T. Lindeberg and L. M. J. Florack, "On the decrease of resolution as a function of eccentricity for a foveal vision system", Tech. Rep. TRITA-NA-P9229, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, November 1992.

[416] T. Lindeberg and J. O. Eklundh, "The scale-space primal sketch: Construction and experiments", Image and Vision Computing, vol. 10, pp. 3-18, January 1992.

[417] T. Lindeberg and J. Gårding, "Shape from texture from a multi-scale perspective", in Proceedings of the fourth ICCV, H. H. Nagel et al., eds., Berlin, Germany, pp. 683-691, IEEE Computer Society Press, 1993.

[418] T. Lindeberg, "Detecting salient blob-like image structures and their scales with a scale-space primal sketch - a method for focus-of-attention", Intern. Journal of Computer Vision, vol. 11, no. 3, pp. 283-318, 1993.

[419] T. Lindeberg, "Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction", Journal of Mathematical Imaging and Vision, vol. 3, no. 4, pp. 349-376, 1993.

[420] T. Lindeberg, "Feature detection with automatic scale selection", Tech. Rep. ISRN KTH/NA/P-96/18-SE, KTH - NADA, May 1996. Earlier version presented in Proc. 8th Scandinavian Conf. on Image Analysis, Tromso, Norway, pp 857-866, 1993.

[421] T. Lindeberg, "Scale-Space Theory in Computer Vision". The Kluwer Intern. Series in Engineering and Computer Science, Dordrecht, the Netherlands: Kluwer Academic Publishers, 1994.

[422] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales", J. of Applied Statistics, 21(2), Supplement on Advances in Applied Statistics: Statistics and Images: 2, pp. 224--270, 1994.

[423] T. Lindeberg and J. Gårding, "Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D structure", in Proc. 3rd European Conf. on Computer Vision (J.-O. Eklundh, ed.), vol. 800 of Lecture Notes in Computer Science, (Stockholm, Sweden), pp. 389-400, Springer-Verlag, May 1994.

[424] T. Lindeberg, "On scale selection for differential operators", Proc. 8th Scandinavian Conf. Image Analysis (K. H. K. A. Hogdra, B. Braathen, ed.), (Tromso, Norway), pp. 857-866, Norwegian Society for Image Processing and Pattern Recognition, May 1994.

[425] T. Lindeberg, "Scale-space behaviour and invariance properties of differential singularities", in Proc. of the NATO Advanced Research Workshop Shape in Picture -- Mathematical Description of Shape in Greylevel Images (Y.-L. O, A. Toet, H. J. A. M. Heijmans, D. H. Foster, and P. Meer, eds.), vol. 126 of NATO ASI Series F, pp. 591-600, Springer Verlag, Berlin, 1994.

[426] T. Lindeberg and B. M. ter Haar Romeny, "Linear scale-space: I. Basic theory. II. Early visual operations", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), Computational Imaging and Vision, pp. 1-38,39-72, Dordrecht, the Netherlands: Kluwer Academic Publishers, 1994.

[427] T. Lindeberg, "Junction detection with automatic selection of detection scales and localization scales", in Proc. First Intern. Conf. on Image Processing, vol. I, Austin, TX, pp. 924-928, IEEE CS, November 1994.

[428] T. Lindeberg and L. M. J. Florack, "Foveal scale-space and linear increase of receptive field size as a function of eccentricity", Tech. Rep. ISRN KTH/NA/P-94/27-SE, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, August 1994.

[429] T. Lindeberg, "Scale-space for N-dimensional discrete signals", in Proc. of the NATO Advanced Research Workshop Shape in Picture - Mathematical Description of Shape in Greylevel Images (Y.-L. O, A. Toet, H. J. A. M. Heijmans, D. H. Foster, and P. Meer, eds.), vol. 126 of NATO ASI Series F, pp. 571-590, Springer Verlag, Berlin, 1994.

[430] T. Lindeberg, "Scale-space theory: a basic tool for analyzing structures at different scales", Journal of Applied Statistics, vol. 21, no. 2, pp. 223-261, 1994. Special issue on Statistics and Images.

[431] T. Lindeberg, "Direct estimation of affine deformations of brightness patterns using visual front-end operators with automatic scale selection", in Proc. 5th Intern. Conf. on Computer Vision (E. Grimson, ed.), (Cambridge, MA), pp. 134-141, IEEE Computer Society Press, June 1995.

[432] T. Lindeberg, "A scale selection principle for estimating image deformations", Tech. Rep. ISRN KTH/NA/P-96/16-SE, KTH - NADA, May 1996. Shortened version in Proc. 5th Int. Conf. on Computer Vision, Boston, Massachussetts, pp 134-141, 1995.

[433] T. Lindeberg and D. Fagerström, "Scale-space with causal time direction", Proc. 4th Europ. Conf. on Computer Vision, Cambridge, UK, (B. Buxton and R. Cipolla, Eds). April 14-18, vol. 1064 of Lecture Notes in Computer Science, pp. 229-240, Springer-Verlag, Berlin, 1996.

[434] T. Lindeberg, "Linear spatio-temporal scale-space", Proc. First Intern. Conf. on Scale-Space Theory in Computer Vision, B.M. ter Haar Romeny ed., Utrecht, Netherlands, Springer-Verlag Lecture Notes in Computer Science, volume 1252, July 2-4, 1997.

[435] T. Lindeberg, "On automatic selection of temporal scales in time-casual scale-space", in Proc. AFPAC'97: Algebraic Frames for the Perception-Action Cycle", G. Sommer and J. J. Koenderink, eds., vol. 1315 of Lecture Notes in Computer Science, Kiel, Germany, pp. 94--113, Springer Verlag, Berlin, Sept. 1997.

[436] T. Lindeberg and J. Gårding, "Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure", Image and Vision Computing, vol. 15, pp. 415-434, 1997.

[437] T. Lindeberg and Li, "Segmentation and classification of edges using minimum description length approximation and complementary junction cues", Computer Vision and Image Understanding, vol. 67, no. 1, pp. 88-98, 1997.

[438] T. Lindeberg, "Feature detection with automatic scale selection", Intern. Journal of Computer Vision, vol. 30, no. 2, pp. 77-116, 1998.

[439] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection", Intern. Journal of Computer Vision, vol. 30, no. 2, pp. 117-154, 1998.

[440] T. Lindeberg, "A scale selection principle for estimating image deformations", Image and Vision Computing, vol. 16, no. 14, pp. 961-977, 1998.

[441] T. Lindeberg, Lidberg and Roland, "Analysis of brain activation patterns using a 3-D scale-space primal sketch", Human Brain Mapping, vol 7, no 3, pp 166--194, 1999.

[442] M. Lindenbaum, M. Fischer, and A. Bruckstein. "On Gabor's contribution to image enhancement", Pattern Recognition Society, vol. 27, no. 1, pp. 1-8, 1994.

[443] A. Liu, S. M. Pizer, D. Eberly, B. Morse, J. Rosenman, and V. Carrasco, "Volume registration using the 3D core", in Proc. SPIE Medical Imaging VIII, (Newport Beach, CA), February 1994.

[444] J. Llacer, B. M. ter Haar Romeny, L. M. J. Florack, and M. A. Viergever, "The use of geometric prior information in bayesian tomographic image reconstruction: a preliminary report", in Proc. SPIE Conf. on Mathematical Methods in Medical Imaging, vol. 1768, San Diego, pp. 82-96, 23-24 July 1992.

[445] J. Llacer, B. M. ter Haar Romeny, L. M. J. Florack, and M. A. Viergever, "The representation of medical images by visual response functions", IEEE Engineering in Medicine and Biology, vol. 3, no. 93, pp. 40-47, 1993.

[446] N. K. Logothetis, H. Guggenberger, S. Peled, and J. Pauls. "Functional imaging of the monkey brain". Nature Neuroscience, volume 2, no 6, pp. 555-562, 1999.

[447] M. Loog, J. J. Duistermaat, and L. M. J. Florack, "On the behavior of spatial critical points under Gaussian blurring. A folklore theorem and scale-space constraints," in M. Kerckhove, ed., Scale-Space and Morphology in Computer Vision: Proceedings of the Third International Conference, Scale-Space 2001, Vancouver, Canada, vol. 2106 of Lecture Notes in Computer Science. Berlin: Springer-Verlag, pp. 183-192, 2001.

[448] H. Lotze, "Mikrokosmos". Leipzig: Hirzel, 1884.

[449] A. M. López, F. Lumbreras, J. Serrat and J. J. Villanueva, "Evaluation of methods for ridge and valley detection", IEEE Tr. on Pattern Analysis and Machine Intelligence (PAMI), vol. 21, pp. 327-335, 1999.

[450] A. M. López, F. Lumbreras, J. Serrat and J. J. Villanueva, "New improvements in the multiscale analysis of trabecular bone patterns", Frontiers in Artificial Intelligence and Applications, Volumen: Pattern Recognition and Applications, IOS Press - Ohmsa, pags: 251-260, 2000.

[451] A. M. López, F. Lumbreras, J. Serrat and J. J. Villanueva, "Multilocal creaseness based on the level set extrinsic curvature", Computer Vision and Image Understanding (CVIU), vol. 77, pp. 111-144, 2000.

[452] K.-C. Low and J. M. Coggins, "Multiscale vector fields for image pattern recognition", tech. rep., Univ. North Carolina, Dept. of Comp. Science, 1989.

[453] D. G. Lowe, "Organization of smooth image curves at multiple scales", Intern. Journal of Computer Vision, vol. 3, pp. 119-130, 1989.

[454] Y. Lu and R. C. Jain, "Behaviour of edges in scale-space", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 11, no. 4, pp. 337-357, 1989.

[455] B. Lucas and T.Kanade, "An iterative image-registration technique with an application to stereo vision", Proc. IJCAI, pp. 674-679, Vancouver, Ca, 1981.

[456] R. Maas, M. Nielsen, W. J. Niessen, B. M. ter Haar Romeny, and M. A. Viergever, "A scale-space approach to binocular stereo". In: Abstracts of the ASCI Imaging Workshop 1995 (L. J. van Vliet and I. T. Young, eds.), (Venray, The Netherlands), p. 34, ASCI, 25-27 October 1995.

[457] R. Maas, M. Nielsen, W. J. Niessen, B. M. ter Haar Romeny, L. M. J. Florack, and M. A. Viergever, "Local disparity measurements using scalable operators", in Proc. DIKU PhD Summerschool on Gaussian Scale-Space Theory (P. Johansen, ed.), no. 96/19 in Tech. Rep., (Copenhagen, Denmark), pp. 80-87, DIKU, 10-13 May 1996. Also in Proc. 5th Danish Conf. on Pattern Recognition and Image Analysis (P. Johansen, ed.), no. 96/22 in Tech. Rep., (Copenhagen, Denmark), pp. 99-106, 26-27 August 1996.

[458] J. MacLean, S. Raab, L. A. Palmer, "Contribution of linear mechanisms to the specifications of local motion by simple cells in areas 17 and 18 of the cat", Visual Neuroscience, vol. 1, 271-294, 1994.

[459] R. Mäder. Programming in *Mathematica.* Addison-Wesley Pub. Co. 3rd edition, 1996.

[460] R. Mäder. The *Mathematica* programmer II. Academic Press, 1996.

[461] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever, "Extraction of invariant ridgelike features for CT and MR brain image matching", in Proc. Int. Conf. on Volume Image Processing (M. A. Viergever, ed.), (Utrecht), pp. 129-132, SCVR, 1993.

[462] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever, "Evaluation of ridge seeking operators for multimodality medical image matching", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 18, no. 4, pp. 353-365, 1996.

[463] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever, "Comparison of feature-based matching of CT and MR brain images", in CVRMed (N. Ayache, ed.), vol. 905 of Lecture Notes in Computer Science, (Berlin), pp. 219-228, Springer-verlag, 1995.

[464] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever, "Comparison of edge-based and ridge-based registration of CT and MR brain images", Medical Image Analysis, vol. 1, no. 2, pp. 151-161, 1996.

[465] J. B. A. Maintz, F. J. Beekman, W. de Bruin, P. A. van den Elsen, P. P. van Rijk, and M. A. Viergever, "Automatic registration and intensity scaling of SPECT brain images", Journal of nuclear medicine, vol. 37, no. 5, supplement, p. 213P, 1996. abstract.

[466] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever, "Registration of SPECT and MR brain images using a fuzzy surface", in Medical Imaging '96 - Image processing (M. H. Loew and K. M. Hanson, eds.), vol. 2710, (Bellingham, WA, USA), pp. 821-829, SPIE, 1996.

[467] J. B. A. Maintz, P. A. van den Elsen, and M. A. Viergever, "Registration of 3D medical images using simple morphological tools", in IPMI '97 (J. Duncan and G. Gindi, eds.), vol. 1230 of Lecture Notes in Computer Science, pp. 204-217, 1997.

[468] J. Malik and P. Perona, "A computational model of texture segmentation", in Proc. CVPR 1989, pp. 326-332, 1989.

[469] J. Malik and P. Perona, "Preattentive texture discrimination with early vision mechanisms", Journal of the Optical Society of America, vol. 7, pp. 923-932, May 1990.

[470] J. Malik and R. Rosenholtz, "A differential method for computing local shape-from-texture for planar and curved surfaces", in Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition, pp. 267-273, 1993.

[471] R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: a level set approach", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 17, no. 2, pp. 158-174, 1995.

[472] R. Malladi and J. A. Sethian, "Level sets and fast marching methods in image processing and computer vision", in Proc. third Intern. Conf. on Image Processing (P. Delogne, ed.), pp. 489-492, IEEE, 1996.

[473] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 674-694, 1989.

[474] S. G. Mallat and S. Zhong, "Characterization of signals from multi-scale edges", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 14, pp. 710-723, 1992.

[475] Z. Z. Manseur and D. C. Wilson, "Decomposition methods for convolution operators", Computer Vision, Graphics, and Image Processing, vol. 53, pp. 428-434, September 1991.

[476] S. Marcelja, "Mathematical Description of the Responses of Simple Cortical Cells", J. Opt. Soc. Am., Vol. 70, 1980.

[477] D. Marr, "Vision".W. H. Freeman and Co., 1882.

[478] D. C. Marr and E. C. Hildreth, "Theory of edge detection", Proc. Roy. Soc. London rm B, vol. 207, pp. 187-217, 1980.

[479] J. B. Martens, "Deblurring digital images by means of polynomial transforms", Computer Vision, Graphics, and Image and Stochastic Processing, vol. 50, pp. 157-176, 1990.

[480] C. Mason and E. R. Kandel, "Central visual pathways", in Principles of Neural Science, pp. 420-434, Prentice-Hall Intern. Inc., 1991. Kandel, E. R. and Schwartz, J. H. and Jessell, T. M. (Eds.).

[481] S. Massey and G.A. Jones, "Decomposition and Hierarchy: Efficient Structural Matching of Large Multi-scale Representations", Proceedings 2. International Conference on Scale-Space Theories in Computer Vision, Lecture Notes in Computer Vision, volume 1682, 1999.

[482] J. C. Maxwell, "On hills and dales", The London, Edinghburgh and Dublin Philosophical Magazine and J. of Science, vol. 40, no. 269, pp. 421-425, 1870.Reprinted in Niven, W. D The Scientific Papers of James Clark Maxwell, Vol II 1956, Dover Publications New York.

[483] B. A. McGuire, C. D. Gilbert, P. K. Rivlin, T. N. Wiesel, "Targets of horizontal connections in macaque primary visual cortex. J. Comp. Neurol. vol. 305, pp. 370-392, 1991.

[484] T. Mclnerney and D. Terzopoulos, "Deformable models in medical images analysis: a survey", Medical Image Analysis, vol. 2, pp. 91-108, 1996.

[485] J. McLean and L. A. Palmer, "Contribution of linear spatiotemporal receptive field structure to velocity selectivity of simple cells in area 17 of cat.", Vision Research, vol. 29, pp. 675-679, 1989.

[486] T. A. McMahon and J. T. Bonner, "On size and life", Scientific American Books, Inc., W. H. Freeman and Company, New York, 1983.

[487] M. Michaelis, Low level image processing using steerable filters. PhD thesis, Technische Fakultät der Christian-Albrechts-Universität Kiel, Germany, Dec. 1995.

[488] M. Michaelis, G. Sommer, "A Lie group approach to steerable filters". Pattern Recognition Letters, vol. 16, no.11, pp. 1165-1174, 1995.

[489] J. Milnor, "Morse theory", vol. 51 of Annals of Mathematics Studies. Princeton University Press, 1963.

[490] A. M. Misha and M. Carver, "The silicon retina", Scientific American, pp. 40-45, May 1991.

[491] C. W. Misner, K. S. Thorne, and J. A. Wheeler, Gravitation. San Francisco: Freeman, 1973.

[492] F. Mokhtarian and A. Mackworth, "Scale-based description of planar curves and two-dimensional shapes", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, pp. 34-43, 1986.

[493] F. Mokhtarian, "Multi-scale description of space curves and three-dimensional objects", in Proc. IEEE CVPR, (Ann Arbor, Michigan), 1988.

[494] F. Mokhtarian, "The renormalized curvature scale-space and the evolutions properties of planar curves", in Proc. IEEE CVPR, (Ann Arbor, Michigan), pp. 318 - 326, 1988.

[495] F. Mokhtarian, "Evolution properties of space curves", in Proc. IEEE CVPR, (Tarpon Springs, Florida), pp. 100-105, 1988.

[496] F. Mokhtarian, "Fingerprint theorems for curvature and torsion zero-crossing", in Proc. IEEE CVPR, (San Diego, California), pp. 269-275, 1989.

[497] F. Mokhatarian and A. Mackworth, "A theory of multi-scale, curvature-based shape representation for planar curves", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 14, pp. 789-805, 1992.

[498] F. Mokhtarian, "Multi-scale torsion-based shape representations for space curves", in Proc. IEEE CVPR, (New York City, NY), 1993.

[499] O. Monga, N. Ayache, and P. T. Sander, "From voxel to intrinsic surface features", Image and Vision Computing, vol. 10, pp. 403-417, July/August 1992.

[500] O. Monga and S. Benayoun, "Using partial derivatives of 3D images to extract typical surface features", Computer Vision and Image Understanding, vol. 61, no. 2, pp. 171-189, 1995.

[501] J. M. Morel and S. Solimini, "Segmentation of images by variational methods: A constructive approach", Rev. Matematica de la Universidad Complutense, vol. 1, no. 3, pp. 169-182, 1988.

[502] J. M. Morel and S. Solimini, Variational Methods in Image Segmentation. No. 14 in Progress in Non-linear Differential Equations and their Applications, Basel, Switzerland: Birkhäuser Verlag, 1995.ISBN 3-7643-3720-6.

[503] Morrison, "Powers of ten: about the relative size of things in the universe", W. H. Freeman and Company, 1985. See also

[504] B. S. Morse, S. M. Pizer, and A. Liu, "Multiscale medial analysis of medical images", in Information Processing in Medical Imaging (IPMI 14) (H. Barrett and A. Gmitro, eds.), (Berlin), Springer-Verlag, 1993.

[505] B. S. Morse, S. M. Pizer, D. T. Puff, and C. Gu, "Zoom-invariant vision of figural shape: Effects on cores of image disturbances", Tech. Rep. TR96-005, University of North Carolina, Dept. of Computer Science, 1996.

[506] D. Mumford and J. Shah, "Boundary detection by minimizing functionals", in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, (San Francisco), 1985.

[507] D. Mumford and J. Shah, "Optimal approximations by piecewise smooth functions and associated variational problems", Communications on Pure and Applied Mathematics, vol. XLII, pp. 577-685, July 1989.

[508] D. Mumford, "On the computational architecture of the neocortex. I. the role of the thalamo-cortical loop", Biological Cybernetics, vol. 65, pp. 135-145, 1991.

[509] D. Mumford, "On the computational architecture of the neocortex: II. The role of cortico-cortical loops:. Biological Cybernetics, vol. 66, pp. 241-251, 1992.

[510] D. Mumford, "Bayesian rationale for the variational formulation", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), Computational Imaging and Vision, pp. 135-146, Kluwer Academic Publishers B.V., 1994.

[511] J. L. Mundy and A. Zisserman, eds., Geometric Invariance in Computer Vision.Cambridge, Massachusetts: MIT Press, 1992.

[512] H. Neumann, H. Ottenberg, and H. S. Stiehl, "Accuracy of regularized differential operators for discontinuity localization in 1-D and 2-D intensity functions", Tech. Rep. FBI-HH-M-186/90, Universit"at Hamburg. Fachbereich Informatik, 1990.

[513] H. Neumann and H. S. Stiehl, "A competitive/cooperative (artificial neural) network approach to the extraction of n-th order junctions", in Proceedings of the 11th DAGM-Symposium. Hamburg (H. Burkhardt, K.-H. Hoehne, and B. Neumann, eds.), (Berlin), Springer-Verlag, 1989.

[514] M. Nielsen, "Isotropic regularization", in Proc. British Machine Vision Conf., pp. 135-144, 1993.

[515] M. Nielsen. "From paradigm to algorithms in computer vision". PhD thesis, Datalogisk Institut Kopenhagen University, Denmark, Dept. of Computer Science, Universitetsparken 1, DK-2100 Kopenhagen 0, Denmark, 1995. ISSN 0107-8283.

[516] M. Nielsen and R. Deriche, "Binocular dense depth reconstruction using isotropy constraint", in Proc. 9th Scand. Conf. on Image Analysis, (Uppsala, Sweden), pp. 49-56, 1995.

[517] M. Nielsen, L. M. J. Florack, and R. Deriche, "Regularization, scale-space, and edge detection filters", in Proc. Fourth European Conf. on Computer Vision, (Cambridge, UK). April 14-18 1996.

[518] M. Nielsen, R. Maas, W. Niessen, L. Florack, and B. M. ter Haar Romeny, "Local disparity structure by scale-space operators", Tech. Rep. 96-17. DIKU Computer Science Department. Copenhagen University, 1996.

[519] M. Nielsen, "Scale-Space Generators and Functionals". In J. Sporring, M. Nielsen, L. Florack, and P. Johansen (eds.) Gaussian Scale-Space Theory, pp. 99-114, Kluwer Academic Publishers. 1997.

[520] M. Nielsen, L. M. J. Florack, and R. Deriche, "Regularization, scale space, and edge detection filters", J. Mathematical Imaging and Vision, vol. 7, pp. 291-307, October 1997.

[521] M. Nielsen and O. F. Olsen, "The structure of the optic flow field", Proc. ECCV, Lecture Notes in Computer Science, vol. 1407, pp. 271-287, 1994.

[522] M. Nielsen, P. Johansen. O. F. Olsen, J. Weickert (Eds.), "Scale-space theories in computer vision", Lecture Notes in Computer Science, Vol. 1682, Springer, Berlin, 1999. ISBN 3-540-66498-X.

[523] M. Nielsen, M. Lillholm, "What do features tell about images?", In "Scale-space theories in computer vision", Lecture Notes in Computer Science, Vol. 2106, pp. 39-50, Springer, Berlin, 2001.

[524] W. J. Niessen, B. M. ter Haar Romeny, and M. A. Viergever, "Numerical analysis of geometry-driven diffusion equations", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), vol. 1 of Computational Imaging and Vision, pp. 393-410, Dordrecht: Kluwer Academic Publishers, 1994.

[525] W. J. Niessen, B. M. ter Haar Romeny, L. M. J. Florack, A. H. Salden, and M. A. Viergever, "Nonlinear diffusion of scalar images using well-posed differential operators", in Proc.of Computer Vision and Pattern Recognition, (Seattle, WA), pp. 92-97, IEEE Computer Society Press, 1994.

[526] W. J. Niessen, J. S. Duncan, L. M. J. Florack, B. M. ter Haar Romeny, and M. A. Viergever, "Spatiotemporal operators and optic flow", in Physics-Based Modeling in Computer Vision (S. T. Huang and D. N. Metaxas, eds.), pp. 78-84, IEEE Computer Society Press. 1995.

[527] W. J. Niessen, J. S. Duncan, B. M. ter Haar Romeny, and M. A. Viergever, "Spatiotemporal analysis of left ventricular motion", in Medical Imaging 95: Image Processing (M. H. Loew, ed.), pp. 250-261, SPIE Press, Bellingham, 1995.

[528] W. J. Niessen, J. S. Duncan, and M. A. Viergever, "A scale-space approach to motion analysis", in Computing Science in the Netherlands 95 (J. C. Van Vliet, ed.), pp. 170-181, Stichting Mathematisch Centrum, Amsterdam, 1995.

[529] W. J. Niessen, B. M. ter Haar Romeny, L. M. J. Florack, and M. A. Viergever, "A general framework for geometry-driven evolution equations", Intern. Journal of Computer Vision, vol. 21, no. 3, pp. 187-205, 1997.

[530] W. J. Niessen, K. L. Vincken, and M. A. Viergever, "Comparison of multiscale representations for a linking-based image segmentation model", in Proc. IEEE Workshop on Mathematical Methods in Biomedical Image Analysis, (San Francisco), pp. 263-272, 1996.

[531] W. J. Niessen, M. Nielsen, L. M. J. Florack, R. Maas, B. M. ter Haar Romeny, and M. A. Viergever, "Multiscale optic flow using physical constraints", in Proc. DIKU PhD Summerschool on Gaussian Scale-Space Theory, no. 96/19 in DIKU Tech. Rep., (Copenhagen, Denmark), 1996.

[532] W. J. Niessen, J. S. Duncan, M. Nielsen, L. M. J. Florack, B. M. ter Haar Romeny, and M. A. Viergever, "A multi-scale approach to image sequence analysis", Computer Vision and Image Understanding, vol. 65, no. 2, pp. 259-268, 1997.

[533] W. J. Niessen and R. Maas, "Multiscale optic flow and stereo", in Gaussian Scale-Space Theory (J. Sporring, M. Nielsen, L. Florack, and P. Johansen, eds.), Computational Imaging and Vision, pp. 31-42, Dordrecht: Kluwer Academic Publishers, 1997.

[534] W. J. Niessen, B. M. ter Haar Romeny, L. M. J. Florack, and M. A. Viergever, "A general framework for geometry-driven evolution equations", Intern. Journal of Computer Vision, vol. 21, no. 3, pp. 187-205, 1997.

[535] W. J. Niessen, A. M. Lopez, W. J. Van Enk, P. M. Van Roermund, B. M. ter Haar Romeny, and M. A. Viergever, "In vivo analysis of trabecular bone architecture", in Proc. Information Processing in Medical Imaging 1997 (J. S. Duncan and G. Gindi, eds.), vol. 1230 of Lecture Notes in Computer Science, pp. 435-440, 1997.

[536] W.J. Niessen, K.L. Vincken, J. Weickert, M.A. Viergever, "Nonlinear multiscale representations for image segmentation", Computer Vision and Image Understanding, Vol. 66. 233-245, 1997.

[537] W. J. Niessen, K. L. Vincken, J. Weickert, and M. A. Viergever, "Three-dimensional MR brain segmentation", in Proc. Sixth Int. Conf. on Computer Vision (ICCV '98, Bombay, Jan. 4-7, 1998), 53-58, 1998.

[538] W.J. Niessen, K.L. Vincken, J. Weickert, B.M. ter Haar Romeny, M.A. Viergever, "Multiscale segmentation of three-dimensional MR brain images", Intern. Journal of Computer Vision, Vol. 31, 185-202, 1999.

[539] M. Nitzberg and T. Shiota, "Nonlinear image filtering with edge and corner enhancement", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 14, no. 8, pp. 826-833, 1992.

[540] N. Nordström, "Biased anisotropic diffusion -- a unified regularization and diffusion approach to edge detection", Image and Vision Computing, vol. 8, no. 11, pp. 318-327, 1990. Also in: Proc. 1st European Conf. on Computer Vision, LNCS-Series Vol. 427, Springer-Verlag, pages 18-27.

[541] I. Ohzawa, G. C. DeAngelis, and R. D. Freeman, "Stereoscopic depth discrimination in the visual cortex: Neurons ideally suited as disparity detectors", Science, vol. 249, pp. 1037-1041, August 1990.

[542] I. Ohzawa, G. C. DeAngelis, and R. D. Freeman, "Encoding of binocular disparity by simple cells in the cat's visual cortex", Journal of Neurophysiology, vol. 75, no. 5, pp. 1779-1805, 1996.

[543] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching", Intern. Journal of Computer Vision, vol. 7, no. 2, pp. 143-162, 1992.

[544] O. F. Olsen and M. Nielsen, "Multi-scale gradient magnitude watershed segmentation". Lecture Notes in Computer Science, vol. 1310, pp 6-13, 1997.

[545] O. F. Olsen, "Generic image structure", PhD Thesis, University of Copenhagen, Dept. of Computer Science, Techn. Rep. DIKU-00-04, June 2000.

[546] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". Nature, 381: 607-609, 1996.

[547] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?", Vision Research, 37: 3311-3325, 1997.

[548] J. Olver, "Applications of Lie Groups to Differential Equations", vol. 107 of Graduate Texts in Mathematics. Springer-Verlag, 1993.

[549] J. Olver, G. Sapiro, and A. Tannenbaum, "Differential invariant signatures and flows in computer vision: A symmetry group approach", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), Computational Imaging and Vision, pp. 255-306, Dordrecht: Kluwer Academic Publishers, 1994.

[550] J. Olver. Equivalence, Invariants, and Symmetry.Cambridge University Press, 1995.

[551] A. H. J. Oomes and P. R. Snoeren, "Structural information in scale space", in Proceedings DIKU PhD Summerschool on Classical Scale-Space Theory, (Copenhagen, Denmark), 1996.

[552] S. Osher and J. Sethian, "Fronts propagating with curvature dependent speed: algoritms based on the Hamilton-Jacobi formalism", Journal of Computational Physics, vol. 79, pp. 12-49, 1988.

[553] G. Østerberg, Topography of the layer of rods and cones in the human retina", Acta Ophthalmologica, vol. 6, pp. 1-103, 1935.

[554] N. Otsu, Mathematical Studies on Feature Extraction in Pattern Recognition. PhD thesis, Researches of the Electrotechnical Laboratory, Ibaraki, Japan, 1981.

[555] N. Otte and H.-H. Nagel, "Optical flow estimation: Advances and comparisons", in: Proc. European Conf. on Computer Vision (Stockholm), Lecture Notes in Computer Science, vol. 800, pp. 51-60, Springer Berlin, 1994.

[556] J. Pacacios, Dimensional Analysis. London: MacMillan and Co. Ltd. 1964.

[557] R. Pankhurst, "Dimensional Analysis and Scale Factors". Chapman and Hall Ltd, London, 1964.

[558] A. C. Papanicolaou, "Fundamentals of Functional Brain Imaging", Swets & Zeitlinger, 2000.

[559] E. J. Pauwels, M. Proesmans, L. J. Van Gool, T. Moons, and A. Oosterlinck, "Image enhancement using coupled anisotropic diffusion equations", in Proc. on the 11th European Conf. on Circuit Theory and Design, vol. 2, pp. 1459-1464, 1993.

[560] E. J. Pauwels, P. Fiddelaers, T. Moons, and L. J. van Gool, "An extended class of scale-invariant and recursive scale-space filters", IEEE Tr. on Pattern Anal. and Machine Perception, vol. 17, no. 1, pp. 691-701, 1995.

[561] S. Pedersen and M. Nielsen, "The Hausdorff dimension and scale-space normalisation of natural images, J. of Visual Communication and Image Representation, vol. 11, no. 2, pp. 266-277, 2000.

[562] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion", in IEEE Computer Society Workshop on Computer Vision, (Miami, FL), pp. 16-22, 1987.

[563] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, pp. 629-639, July 1990.

[564] P. Perona, "Deformable kernels for early vision", in IEEE CVPR, pp. 222-227, June 1994.

[565] P. Perona, "Steerable-scalable kernels for edge detection and junction analysis", in Proc. 2nd European Conf. on Computer Vision, (Santa Margherita Ligure, Italy), pp. 3-18, May 1992.

[566] Perona, T. Shiota, and J. Malik, "Anisotropic diffusion", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), Computational Imaging and Vision, pp. 73-92, Kluwer Academic Publishers B.V., 1994.

[567] P. Perona, "Deformable kernels for early vision." IEEE Pattern Anal. and Machine Perc., vol. 17, no. 5, pp. 488-499, 1995.

[568] E. Peterhans and R. von der Heydt, "Subjective contours - bridging the gap between psychophysics and physiology", Trends in Neurosciences, vol. 14, no. 3, pp. 112-119, 1991.

[569] E. Peterhans and R. von der Heydt, Representation of vision. Trends and tacit assumptions, ch. Elements of form perception in monkey prestriate cortex, pp. 111-124.Cambridge: Cambridge University Press, 1991. A. Gorea, Y. Fregnac, Z. Kapoula and J. Findlay, Eds.

[570] M. Petrou and P. Bosdogianni, "Image processing; the fundamentals", John Wiley & Sons, Chichester, 1999.

[571] C. A. Pickover, "Strange Brains and Genius: The Secret Lives of Eccentric Scientists and Madmen", Plenum Publishing, 1998. ISBN 0-306-45784-9.

[572] M. A. Piech, "Decomposing the Laplacian", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 12, no. 8, pp. 830-831, 1990.

[573] S. M. Pizer, J. J. Koenderink, L. M. Lifshitz, L. Helmink, and A. D. J. Kaasjager, "An image description for object definition, based on extremal regions in the stack", Information Processing in Medical Imaging, Proceedings of the 8th conference, pp. 24-37, 1985.

[574] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. M. ter Haar Romeny, J. B. Zimmerman and K. J. Zuiderveld: "Adaptive Histogram Equalization and its variations", Computer Vision, Graphics and Image Processing, vol. 39, pp. 355-368, 1987.

[575] S. M. Pizer, J. M. Gauch, L. M. Lifshitz, and W. R. Oliver, "Image description via annihilation of essential structures", Tech. Rep. TR88-001, University of North Carolina at Chapel Hill, 1988.

[576] S. M. Pizer, J. M. Gauch, J. M. Coggins, R. E. Fredericksen, T. J. Cullip, and V. L. Interrante, "Multiscale, geometric image descriptions for interactive object definition", in Mustererkennung 1989 (Proc. 11th Symposium of DAGM), Informatik-Fachberichte 219, pp. 229-239, DAGM [The German Association for Pattern Recognition], Springer-Verlag, 1989.

[577] S. M. Pizer, J. M. Gauch, T. J. Cullip, and R. E. Fredericksen, "Descriptions of intensity structure via scale and symmetry", in Proceedings First Conf. on Visualization in Biomedical Computing, pp. 94-101, 1990.

[578] S. M. Pizer and B. M. ter Haar Romeny, "Fundamental properties of medical image perception", Journal of Digital Imaging, vol. 4, pp. 1-20, Febr. 1990.

[579] S. M. Pizer, C. A. Burbeck, J. M. Coggins, D. S. Fritsch, and B. S. Morse, "Object shape before boundary shape: Scale-space medial axes", Journal of Mathematical Imaging and Vision, vol. 4, no. 3, pp. 303-313, 1994.

[580] T. Poggio, H. Voorhees, and A. Yuille, "A regularized solution to edge detection", AI memo 833, MIT, May 1985.

[581] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory", Nature, vol. 317, pp. 314-319, 1985.

[582] G. F. Poggio, F. Gonzalez, and F. Krause, "Stereoscopic mechanisms in monkey visual cortex: Binocular correlation and disparity sensitivity", J. Neurosc, vol. 8, no. 12, pp. 4531-4550, 1988.

[583] D. A. Pollen and S. F. Ronner, "Phase Relationships Between Adjacent Simple Cells in the Visual Cortex, Science Vol. 212, 1981.

[584] D. A. Pollen and S. F. Ronner, "Spatial computation performed by simple and complex cells in the visual cortex of the cat", Vision Research, vol. 22, pp. 101-118, 1982.

[585] E. Pöppel, "Time Perception", In: Handbook of Sensory Physiology, R. Held, H.W. Leibowitz, H.-L. Teuber, eds., pp. 713-729, Springer, Heidelberg (1978).

[586] T. Poston and I. Steward, "Catastrophe theory and its applications". London: Pitman, 1978.

[587] F. Preteux, "Watershed and skeleton by influence zones: A distance-based approach", Journal of Mathematical Imaging and Vision, vol. 1, pp. 239-256, September 1992.

[588] K. H. Pribram, "Brain and perception: Holonomy and structure in figural processing". Lawrence Erlbaum Associates, 1991.

[589] M. Proesmans, E. Pauwels, and L. Van Gool, "Coupled geometry-driven diffusion equations for low level vision", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), pp. 191-228, Kluwer Academic Publishers B.V., 1994.

[590] M. H. Protter and H. F. Weinberger, "Maximum principles in differential equations". New York: Prentice-Hall, 1984.

[591] D. Puff, D. Eberly, and S. Pizer, "Object-based interpolation via the multiscale medial axis", in Proc. SPIE Medical Imaging VIII, February 1994.

[592] E. Radmoser, O. Scherzer, J. Weickert, "Scale-space properties of regularization methods",
M. Nielsen, P. Johansen, O.F. Olsen, J. Weickert (Eds.), Scale-space theories in computer vision, Lecture Notes in Computer Science, Vol. 1682, Springer, Berlin, 211-222, 1999.

[593] E. Radmoser, O. Scherzer, J. Weickert, "Scale-space properties of nonstationary iterative regularization methods", Journal of Visual Communication and Image Representation (Special Issue on Scale-Space Theories in Computer Vision, invited paper), 2000.

[594] S. V. Raman, S. Sarkar, and K. L. Boyer, "Tissue boundary refinement in magnetic resonance images using contour-based scale-space matching", IEEE Tr. on Medical Imaging, vol. 10, pp. 109-121, June 1991.

[595] K. Rangarajan, M. Shah, and D. Van Brackle, "Optimal corner detector", in Proc. IEEE ICCV, (Tampa, FL), pp. 90-94, 1988.

[596] A. R. Rao and B. G. Schunk, "Computing oriented texture fields", Computer Vision, Graphical Models and Image Processing, vol. 53, pp. 157-185, 1991.

[597] R. P. N. Rao, B. A. Olshausen, and Michael S. Lewicki (Eds.), "Probabilistic models of the brain: perception and neural function", MIT Press, 2001.

[598] Lord Rayleigh, "The principle of similitude", Nature, vol. XCV, pp. 66-68, 644, March 1915.

[599] W. E. Reichardt, "Autocorrelation, a principle for the evaluation of sensory information by the central nervous system". W. A. Rosenblith (ed.). MIT Press, Cambridge Mass., 1961.

[600] W. E. Reichardt, "Movement perception in insects", In W. E. Reichardt (ed.), Processing of optical data by organisms and by machines. New York, Academic Press, 1961.

[601] W. E. Reichardt and M. Egelhaaf, "Properties of individual movement detectors as derived from behavioural experiments on the visual system of the fly", Biological Cybernetics, vol. 58, pp. 287-294, 1988.

[602] Z. Réti, "Deblurring images blurred by the discrete Gaussian", AML, vol. 8, no. 4, pp. 29-35, 1995.

[603] W. Richards, ed., Natural computation.Cambridge, Ma.: MIT Press, 1988.

[604] J. H. Rieger, "Generic evolutions of edges on families of diffused greyvalue surfaces", Journal of Mathematical Imaging and Vision, vol. 5, pp. 207-217, September 1995.

[605] D. L. Ringach, G. Sapiro and R. Shapley, "A subspace reverse correlation technique for the study of visual neurons", Vision Research, Vol 37, No 17, pp. 2455-2464, 1997.

[606] J. Rissanen, "Modeling by the shortest data description," Automatica, vol. 14, pp. 465-471, 1978.

[607] G. X. Ritter and J. N. Wilson, "Handbook of computer vision algorithms in image algebra", CRC Press, Boca Raton, 2001.

[608] R. W. Rodieck, "The first steps in seeing". Sinauer Associates, Inc., Sunderland MA, 1998.

[609] A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis", IEEE Trans. on Computers, vol. C-20, pp. 562-569, May 1971.

[610] A. Rosenfeld, "Multiresolution Image Processing and Analysis", vol. 12 of Springer Series in Information Sciences. Springer-Verlag, 1984.

[611] P. L. Rosin, A. C. F. Colchester, and D. J. Hawkes, "Early image representation using regions defined by maximum gradient profiles between singular points", Pattern Recognition, vol. 25, no. 7, pp. 695-711, 1992.

[612] L. Rosin, "Representing curves at their natural scales", Pattern Recognition, vol. 25, no. 11, pp. 1315-1325, 1992.

[613] J. Rubner and K. Schulten, "Development of feature detectors by self-organization", Biological Cybernetics, vol. 62, pp. 193-199, 1990.

[614] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms", Physica D, vol. 60, pp. 259-268, 1992.

[615] H. Ruskeepää, "*Mathematica* Navigator: Graphics and methods of applied Mathematics". Academic Press, London, 1999. ISBN 0126036403.

[616] J. C. Russ, The Image Processing Handbook. Boca Raton: CRC Press, 1994. Second Edition.

[617] P. Saint-Marc, J. S. Chen, and G. Medioni, "Adaptive smoothing: A general tool for early vision", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 13, no. 6, pp. 514-529, 1991.

[618] A. H. Salden, B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, "A complete and irreducible set of local orthogonally invariant features of 2-dimensional images", in Proceedings 11th IAPR Internat. Conf. on Pattern Recognition (I. T. Young, ed.), (The Hague, the Netherlands), pp. 180-184, IEEE Computer Society Press, Los Alamitos, August 30-September 3 1992.

[619] A. H. Salden, L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever, "Multi-scale analysis and description of image structure", Nieuw Archief voor Wiskunde, vol. 10, no. 3, pp. 309-326, 1992.

[620] A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Image structure generating normalised geometric scale spaces", in Volume Image Processing '93 (M. A. Viergever, ed.), (Utrecht, the Netherlands), pp. 141-143, 1993.

[621] A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Dynamic scale-space theories", in Proc. Conf. on Differential Geometry and Computer Vision: From Pure over Applicable to Applied Differential Geometry, (Nordfjordeid, Norway), August 1-7 1995.

[622] A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Classical scale space theory from physical principles", Journal of Mathematical Imaging and Vision, 1998.

[623] A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Linear scale space theory from physical properties", J. of Mathematical Imaging and Vision, vol. 9. no.2, pp. 103-140, 1998.

[624] A. H. Salden, B. M. ter Haar Romeny, and M. A. Viergever, "Algebraic invariants of linear scale spaces." Journal of Mathematical Imaging and Vision, March 1999.

[625] P. Sander and S. W. Zucker: "Singularities of principal direction fields from 3D images", IEEE tr. on Pattern Analysis and Machine Intelligence, vol. 14, no. 3., pp. 309-317, 1992.

[626] W. Sanns, "Catastrophe theory with *Mathematica*, a geometric approach", Der Andere Verlag, Osnabrück, Germany. ISBN 3-934366-76-7.

[627] G. Sapiro and A. Tannenbaum, "Affine invariant scale-space", Intern. Journal of Computer Vision, vol. 11, pp. 25-44, 1993.

[628] G. Sapiro and A. Tannenbaum. "On invariant curve evolution and image analysis", Indiana Journal of Mathematics, vol. 42. no. 3, pp. 985-1009, 1993.

[629] G. Sapiro and A. Tannenbaum. "Area and length preserving geometric invariant scale-spaces", Tech. Rep. LIDS-2200, MIT, 1993. Accepted for publication in IEEE-PAMI. Also in Proc. ECCV '94, Stockholm, May 1994.

[630] G. Sapiro, "From active contours to anisotropic diffusion: connections between basic pde's in image processing", in Proc. third Intern. Conf. on Image Processing (P. Delogne. ed.), pp. 477-480, IEEE, 1996.

[631] W. Scanns, "Catastrophe theory with *Mathematica*, a geometric approach". Der Andere Verlag, ISBN 3934366767, 2000.

[632] H. Scharr, J. Weickert, "An anisotropic diffusion algorithm with optimized rotation invariance", G. Sommer, N Krüger, C. Perwass (Eds.), Mustererkennung 2000, Springer, Berlin, 460-467, 2000.

[633] O. Scherzer and J. Weickert, "Relations between regularization and diffusion filtering", J. of Math. Imaging and Vision, vol. 12. no. 1, pp. 43-63. 2000. Revised version of Technical Report DIKU-98/23, Dept. of Computer Science, University of Copenhagen, Denmark, 1998.

[634] C. Schmidt and R. Mohr, "Combining greyvalue invariants with local constraints for object recognition", in Proc. Intern. Conf. on Computer Vision and Pattern Recognition CVPR, (San Francisco), IAPR, June 16-20 1996.

[635] C. Schmidt and R. Mohr, "Object recognition using local characterization and semi-local constraints", tech. rep., INRIA, 1996.

[636] C. Schnörr, J. Weickert, "Variational image motion computation: theoretical framework, problems and perspectives", G. Sommer, N Krüger, C. Perwass (Eds.), Mustererkennung 2000, Springer, Berlin, 476-487, 2000. Invited paper.

[637] I. J. Schönberg, "On smoothing operations and their generating functions". Bull. Amer. Math. Soc., vol. 59, pp. 199-230, 1953.

[638] B. G. Schunck, "The motion constraint equation for optical flow", in Proceedings of the 7th Intern. Conf. on Pattern Recognition, pp. 22-24, 1984.

[639] L. Schwartz, "Theorie des distributions", vol. I, II of Actualites scientifiques et industrielles; Vol. 1091 and 1122. Paris: Publications de l'Institut de Mathématique de l'Université de Strasbourg, 1950-1951. See also: Hermann, Paris, 1951, 2nd edition 1966.

[640] E. L. Schwartz, "Topographical mapping in primate visual cortex: history, anatomy, and computation". In D.H. Kelly, editor, Visual Science and Engineering: models and applications, chapter 8, pages 293-360. Marcell Dekker, Inc, New York, 1994.

[641] A. Seckel, "The art of optical illusions", Carlton Books Ltd., 2000.

[642] J. Serra, Image Analysis and Mathematical Morphology. London, New York, Paris, San Diego, San Francisco, S ao Paulo, Sydney, Tokyo and Toronto: Academic Press. 1982.

[643] J. A. Sethian, An Analysis of Flame Propagation. Ph.D. thesis, Dept. of Mathematics, University of California, Berkeley, CA, 1982.

[644] J. A. Sethian. "Curvature and the evolution of fronts", Communications Mathematical Physics, vol. 101, pp. 487-499, 1985.

[645] J. A. Sethian, "A review of recent numerical algorithms for hypersurfaces moving with curvature dependent speed", J. Differential Geometry, vol. 31, pp. 131-161, 1989.

[646] Sethian, J.A.. Fast Marching Methods and Level Set Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Sciences, Cambridge University Press. 1999.

[647] J. Shah. "Segmentation by nonlinear diffusion", Proc. Conf. on Computer Vision and Pattern Recognition, pp. 202-207, June 1991.

[648] J. Shah, "Segmentation by minimizing functionals: Smoothing properties", SIAM J. Control and Optimization, vol. 30, pp. 99-111, January 1992.

[649] S. M. Sherman and C. Kock, "The control or retinogeniculate transmission in the mammalian lateral geniculate nucleus", Experimental Brain Research, vol. 63, pp. 1-20, 1986.

[650] S. M. Sherman and C. Kock, "Thalamus". in The Synaptic Organization of the Brain (G. M. Shepherd, ed.), pp. 246-278. New York: Oxford University Press, 1990.Third Edition.

[651] S. M. Sherman, "Dynamic gating of retinal transmission to the visual cortex by the lateral geniculate nucleus", in Thalamic Networks for Relay and Modulation (D. Minciacchi, M. Molinari, G. Macchi, and E. G. Jones, eds.), pp. 61-79, Oxford: Pergamon Press, 1993.

[652] S. M. Sherman, "Dual response modes in lateral geniculate neurons: Mechanisms and functions", Visual Neuroscience, vol. 13, no. 2, pp. 205-213, 1990.

[653] D. A. Sholl, "Dendritic organization in the neurons of the visual cortices of the cat", Journal of Anatomy, 87: 387-406, 1953.

[654] A. Shokoufandeh, S. Dickinson, C. Jonsson, L. Bretzner, and T. Lindeberg, "On the representation and matching of qualitative shape at multiple scales", Proceedings European Conference on Computer Vision, Copenhagen, May, pp. 759-775, 2002.

[655] D. Shy and P. Perona, "X-y separable pyramid steerable scalable kernels", in Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition, CVPR'94, pp. 237-244, IEEE, 1994.

[656] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. W. Zucker, "Shock graphs and shape matching", Int. J. of Computer Vision, 35(1): 13-32, 1999.

[657] J. G. Simmonds, A Brief on Tensor Analysis.Undergraduate Texts in Mathematics, Springer-Verlag, 1995. Second Edition.

[658] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. Heeger, "Shiftable multi-scale transforms", IEEEIT, vol. 38, pp. 587-607, March 1992.

[659] E. P. Simoncelli and W. Freeman, "The steerable pyramid: a flexible architecture for multi-scale derivative computation", in Proc. of the 2nd Annual IEEE Int. Conf. on Image Processing, IEEE, oct 1995.

[660] E. P. Simoncelli and H. Farid, "Steerable wedge filters for local orientation analysis", IEEE Tr. on Image Processing, vol. 5, no. 9, pp. 1377, 1996.

[661] E. P. Simoncelli, "A rotation invariant pattern signature", in Proc. of the 3rd IEEE Int. Conf. on Image Processing, pp. 185-188, 1996.

[662] E. P. Simoncelli, "A rotation invariant pattern signature", in Proc. of the 3rd IEEE Int. Conf. on Image Processing, pp. 185-188, 1996.

[663] W. Snyder, Y.-S. Han, G. Bilbro, R. Whitaker, and S. Pizer, "Image relaxation: Restoration and feature extraction", IEEE Tr. PAMI, vol. 17, no. 6, pp. 620-624, 1995.

[664] G. Sperling and Z. L. Lu, "A systems analysis of visual motion perception". In: High-level motion processing. Takeo Watanabe (Ed). Cambridge MA: MIT Press, pp.153-183, 1998.

[665] L. Spillmann and J. S. Werner, Visual Perception: the Neurophysiological Foundations. Academic Press Inc., 1989.

[666] M. Spivak, Calculus on Manifolds. New York, New York, USA: W. A. Benjamin, Inc., 1965.

[667] M. Spivak, A Comprehensive Introduction to Differential Geometry, vol. I-V. Houston, Texas: Publish or Perish, Inc., second edition ed., 1979.

[668] J. Sporring, "The entropy of scale-space", in Proceedings 13th ICPR, Austria, 1996.

[669] J. Sporring, M. Nielsen, L. Florack, and P. J. (Eds.), "Gaussian Scale-Space". Dordrecht: Kluwer Academic Publishers, 1996.

[670] J. Sporring and J. Weickert, "On generalized entropies and scale-space", in Scale-Space Theory in Computer Vision (B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, eds.), vol. 1252 of Lecture Notes in Computer Science, pp. 53-64, Springer, Berlin, 1997.

[671] J. Sporring, M. Nielsen, J. Weickert, O.F. Olsen, "A note on differential corner measures", Proc. 14th Int. Conf. Pattern Recognition (ICPR 14, Brisbane, Aug. 17-20, 1998), IEEE Computer Society Press, Los Alamitos, Vol. 1, 652-654, 1998.

[672] J. Sporring, M. Nielsen, O.F. Olsen, J. Weickert, "Smoothing images creates corners", Image and Vision Computing, Vol. 18, 261-266, 2000. Revised version of Technical Report DIKU-98/1, Dept. of Computer Science, University of Copenhagen, Denmark, 1998.

[673] J. Staal, S. Kalitzin, B. M. ter Haar Romeny and M. A. Viergever, "Detection of critical structures in scale-space". Lecture Notes of Computer Science, vol. 1682, pp. 105-116, 1999.

[674] P. Tavan, H. Grubmuller, and Kuhnel, "Self-organization of associative memory and pattern classification: Recurrent signal processing on topological feature maps", Biological Cybernetics, vol. 64, pp. 95-105, 1990.

[675] P. C. Teo, Y. Hel-Or, Lie generators for computing steerable functions. Pattern Recognition Letters, vol. 19, pp. 7-17, 1998.

[676] B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, "Invariant third order properties of isophotes: T-junction detection", in Proc.7th Scand. Conf. on Image Analysis (P. Johansen and S. Olsen, eds.), Aalborg DK, pp. 346-353, August 1991. Also in: Theory & Applications of Image Analysis (P. Johansen and S. Olsen, eds.), vol. 2 of Series in Machine Perception and Artificial Intelligence, pp. 30-37, Singapore: World Scientific, 1992.

[677] B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, "Scale-space: Its natural operators and differential invariants", in Information Processing in Medical Imaging (A. C. F. Colchester and D. J. Hawkes, eds.), vol. 511 of Lecture Notes in Computer Science, pp. 239-255, Springer-Verlag, Berlin, July 1991.

[678] B. M. ter Haar Romeny and L. M. J. Florack, "A multiscale geometric model of human vision", in Perception of Visual Information (W. R. Hendee and P. N. T. Wells, eds.), ch. 4, pp. 73-114, Berlin: Springer-Verlag, 1993. Second edition 1996.

[679] B. M. ter Haar Romeny, L. M. J. Florack, A. H. Salden, and M. A. Viergever, "Higher order geometrical image structure", in Proc. Information Processing in Medical Imaging '93, Flagstaff AZ (H. Barrett, ed.), (Berlin), pp. 77-93, Springer-Verlag, 1993.

[680] B. M. ter Haar Romeny, L. M. J. Florack, M. de Swart, J. Wilting, and M. A. Viergever, "Deblurring Gaussian blur", in Proceedings Mathematical Methods in Medical Imaging II, vol.~2299, (San Diego, CA), pp. 139--148, SPIE, July 25-26, 1994.

[681] B. M. ter Haar Romeny, W. J. Niessen, J. Wilting, and L. M. J. Florack, "Differential structure of images: Accuracy of representation", in Proc. First IEEE Internat. Conf. on Image Processing, (Austin, TX), pp. 21-25, IEEE, November, 13-16 1994.

[682] B. M. ter Haar Romeny (ed.), "Geometry-driven diffusion in computer vision". Dordrecht: Kluwer Academic Publishers, 1994.

[683] B. M. ter Haar Romeny, "Scale-space research at Utrecht University", in Proc. 12th Intern. Conf. on Analysis and Optimization of Systems: Images, Wavelets and PDE's (M.-O. Berger, R. Deriche, I. Herlin, J. Jaffré, and J.-M. Morel, eds.), Lecture Notes in Control and Information Sciences, vol. 219, pp. 15-30, Springer, London, June 26-28 1996.

[684] B. M. ter Haar Romeny, W. J. Niessen, J. Weickert, P. van Roermund, W. van Enk, A. Lopez, and R. Maas, "Orientation detection of trabecular bone", in Progress in Biophysics and Molecular Biology, vol. 65, pp. P-H5-43, August 11-16 1996. Proc. 12th Intern. Biophysics Congress.

[685] B. M. ter Haar Romeny, "Applications of scale-space theory", in Gaussian Scale-Space Theory (J. Sporring, M. Nielsen, L. Florack, and P. Johansen, eds.), Computational Imaging and Vision, pp. 3-19, Dordrecht: Kluwer Academic Publishers, 1997.

[686] B. M. ter Haar Romeny, L. M. J. Florack, J. J. Koenderink, and M. A. Viergever, eds., "Scale-Space '97: Proc. First Internat. Conf. on Scale-Space Theory in Computer Vision", vol. 1252 of Lecture Notes in Computer Science. Berlin: Springer Verlag, 1997.

[687] B. M. ter Haar Romeny, B. Titulaer, S. Kalitzin, G. Scheffer, F. Broekmans and E. te Velde, "Computer assisted human follicle analysis for fertility prospects with 3D ultrasound", Proceedings Intern. Conf. on Information processing in Medical Imaging (IPMI99), vol. 1613, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 1999.

[688] B. M. ter Haar Romeny, L.M. J. Florack, "Front-End Vision, a Multiscale Geometry Engine". Proc. First IEEE Intern. Workshop on Biologically Motivated Computer Vision (BMCV2000), May 15-17, 2000, Seoul, Korea. Lecture Notes in Computer Science, 2000.

[689] B. M. ter Haar Romeny, "Computer Vision and Mathematica", J. of Computing and Visualization in Science, 2002.

[690] E. ter Haar Romeny, "Edlef ter Haar Romeny, painter", Van Gruting Publishers, Westervoort, the Netherlands, 2002.

[691] D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, pp. 413-424, 1981.

[692] J. Thirio, S. Benayoun: "Image surface extremal points, new feature points for image registration". INRIA Technical Report RR-2003, 1993

[693] J.-P. Thirion and A. Gourdon, "Computing the differential characteristics of isodensity surfaces", Computer Vision, Graphics, and Image Processing: Image Understanding, vol. 61, pp. 190-202, March 1995.

[694] J.-P. Thirion: "The extremal mesh and the understanding of 3D surfaces". Intern. J. of Computer Vision, vol. 19, no. 2, pp. 115-128, 1996.

[695] R. Thom, "Structural stability and Morphogenesis" (transl. D. H. Fowler). New York: Benjamin-Addison Wesley, 1975.

[696] D. W. Thompson, "On Growth and Form". Cambridge University Press, 1942.

[697] P. Thompson, "Margaret Thatcher: a new illusion". Perception, vol. 9, pp. 483-484, 1980.

[698] A. N. Tikhonov and V. Y. Arsenin, "Solution of Ill-Posed Problems". Washington DC: Winston and Wiley, 1977.

[699] M. Tistarelli and G. Sandini, "On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 15, pp. 401-416, April 1993.

[700] A. Toet, J. Blom and J. J. Koenderink, "The construction of a simultaneous functional order in nervous systems", Biol. Cybern., vol. 57, pp. 115-125, 127-136, 331-340, 1987.

[701] V. Torre and T. A. Poggio, "On edge detection", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, no. 2, pp. 147-163, 1986.

[702] D. Y. Ts'o, R. D. Frostig, E. E. Lieke, and A. Grinvald, "Functional organization of primate visual cortex revealed by high resolution optical imaging". Science 249: 417-20, 1990.

[703] W.A. van de Grind, J. J. Koenderink and A. J. van Doorn, "Motion detection from photopic to low scotopic luminance levels". Vision Research, vol. 40, no. 2, pp. 187-199, 1999.

[704] R. van den Boomgaard. "The morphological equivalent of the Gauss convolution", Nieuw Archief voor Wiskunde (in English), vol. 10, pp. 219-236, November 1992.

[705] R. van den Boomgaard and A. W. M. Smeulders, "Morphological multi-scale image analysis", in Mathematical Morphology and its Applications to Signal Processing (J. Serra and P. Salembier, eds.), (Barcelona, Spain), pp. 180-185, Universitat Politechnica de Catalunya,, May 1993.

[706] R. van den Boomgaard and A. W. M. Smeulders, "The morphological structure of images, the differential equations of morphological scale-space", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 16, pp. 1101-1113, November 1994.

[707] R. van den Boomgaard and L. Dorst, "The morphological equivalent of Gaussian scale-space", Gaussian Scale-Space Theory, J. Sporring and M. Nielsen and L. Florack and P. Johansen (eds), Series: Computational Imaging and Vision, Kluwer Academic Publishers, pp 203-220, 1997.

[708] P. van den Elsen, E. J. D. Pol, J. B. A. Maintz, and M. A. Viergever, "Image fusion using geometrical features", in SPIE Vol. 1808 Visualization in Biomedical Computing (R. A. Robb, ed.), (Bellingham, WA), SPIE Press, 1992.

[709] P. van den Elsen and M. A. Viergever, "Fully automated CT and MR brain image registration by correlation of geometrical features", in Proc. Information Processing in Medical Imaging '93, Flagstaff AZ (H. Barrett, ed.), Berlin, Springer Verlag, 1993.

[710] A. van den Elsen, J. B. A. Maintz, E. J. D. Pol, and M. A. Viergever, "Automatic registration of CT and MR brain images using correlation of geometrical features", IEEE Tr. on Medical Images, vol. 14, no. 2, pp. 384-398, 1995.

[711] B. van Ginneken and B. M. ter Haar Romeny, "Applications of locally orderless images", In: Scale-space theories in computer vision, Lecture Notes in Computer Science, vol. 1682, pp. 10-21, Springer, Berlin, 1999.

[712] B. van Ginneken and B. M. ter Haar Romeny, "Applications of locally orderless images", Journal of Visual Communication and Image Representation, vol. 11, no. 2, pp. 196-208, June 2000.

[713] K. L. Vincken, C. N. de Graaf, A. S. E. Koster, M. A. Viergever, F. J. R. Appelman, and G. R. Timmens, "Multiresolution segmentation of 3D images by the hyperstack", in Proc. First Conf. on Visualization in Biomedical Computing, pp. 115-122, Los Alamitos, CA: IEEE Computer Society Press, 1990.

[714] K. L. Vincken, W. J. Niessen, and M. A. Viergever, "Blurring strategies for image segmentation using a multiscale linking model", in Proc. Computer Vision and Pattern Recognition, (San Francisco, CA), pp. 21-26, IEEE Computer Society Press, 1996.

[715] K. L. Vincken, A. S. E. Koster, and M. A. Viergever, "Probabilistic multiscale image segmentation", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 2, no. 19, pp. 109-120, 1997.

[716] R. von der Heydt and E. Peterhans, "Mechanisms of contour perception in monkey visual cortex I. Lines of pattern discontinuity", Journal of Neuroscience, vol. 9, no. 5, pp. 1731-1748, 1989.

[717] G. Wallis, "Neural Mechanisms Underlying Processing in the Visual Areas of the Occipital and Temporal Lobes", PhD thesis Max Planck Institute for Biological Cybernetics, Tubingen, Germany, 1994. URL: www.kyb.tuebingen.mpg.de/bu/people/guy/alln/alln.html.

[718] B. A. Wandell, Foundations of Vision. Sunderland MA: Sinauer Associates, Inc., 1995.

[719] H. Wässle, Visual neuroscience, ch. Sampling of visual space by retinal ganglion cells, pp. 19-32.Cambridge University Press, 1986. J. D. Pettigrew, K. J. Sanderson and W. R. Levick, Eds.

[720] H. Wässle, U. Gruenert, J. Roehrenbeck, and B. Boycott, "Retinal ganglion cell density and cortical magnification factor in the primate", Vision Research, vol. 30, pp. 1897-1911, 1990.

[721] H. Wässle and B. B. Boycott. "Functional architecture of the mammalian retina". Physiological Review, 71:447-480, 1991.

[722] A. B. Watson, "Summation of grating patches indicate many types of detector at one retinal location", Vision Research, vol. 22, 17-25, 1982.

[723] A. B. Watson, "The cortex transform: Rapid computation of simulated neural images", Computer Vision, Graphics, and Image Processing, 39 (3), 311-327, 1987.

[724] D. J. Watts, P. Sheridan Dodds, M. E. J. Newman: "Identity and search in social networks", Science vol. 296, no. 5571, pp. 1302-1305, 17 May 2002.

[725] J. Weber and J. Malik, "Robust computation of optical-flow in a multiscale differential framework", Intern. Journal of Computer Vision, vol. 14, no. 1, pp. 67-81, 1995.

[726] J. Weickert, "Anisotropic diffusion filters for image processing based quality control", in Proc. Seventh European Conf. on Mathematics in Industry (A. Fasano and M. Primicerio, eds.), pp. 355-362, Teubner, Stuttgart, 1994.

[727] J. Weickert, "Scale-space properties of nonlinear diffusion filtering with a diffusion tensor", Tech. Rep. 110, Laboratory of Technomathematics, Univ. of Kaiserslautern, Germany, October 1994.

[728] J. Weickert, "Multiscale texture enhancement", in Computer Analysis of Images and Patterns (V. Hlavac and R. Sara, eds.), vol. 970 of Lecture Notes in Computer Science, pp. 230-237, Springer, Berlin, 1995.

[729] J. Weickert, "Foundations and applications of nonlinear anisotropic diffusion filtering", Z. Angew. Math. Mech., Suppl. 1, vol. 76, pp. 283-286, 1996.

[730] J. Weickert, "Theoretical foundations of anisotropic diffusion in image processing", Computing Suppl., vol. 11, pp. 221-236, 1996.

[731] J. Weickert, "Nonlinear diffusion scale-spaces: From the continuous to the discrete setting", in ICAOS '96: Images, Wavelets and PDEs (M.-O. Berger, R. Deriche, I. Herlin, J. Jaffré, and J.-M. Morel, eds.), vol. 219 of Lecture Notes in Control and Information Sciences, pp. 111-118, Springer, London, 1996.

[732] J. Weickert, "A model for the cloudiness of fabrics", in Progress in Industrial Mathematics at ECMI 94 (H. Neunzert, ed.), pp. 258-265, Wiley-Teubner, Chichester, 1996.

[733] J. A. Weickert, B. M. ter Haar Romeny, and M. A. Viergever, "Conservative image transformations with restoration and scale-space properties", in Proc. 1996 IEEE Int. Conf. Image Processing, vol. I, (ICIP-96, Lausanne, Sept. 16-19, 1996), pp. 465-468, 1996.

[734] J. Weickert, S. Ishikawa, A. Imiya, "On the history of Gaussian scale-space axiomatics", in J. Sporring, M. Nielsen, L. Florack, P. Johansen (Eds.), Gaussian scale-space theory, Kluwer, Dordrecht, 45-59, 1997.

[735] J. Weickert, "Recursive separable schemes for nonlinear diffusion filters", in Scale-Space Theory in Computer Vision (B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, eds.), vol. 1252 of Lecture Notes in Computer Science, pp. 260-271, Springer, Berlin, 1997.

[736] J. Weickert and B. Benhamouda, "A semidiscrete nonlinear scale-space theory and its relation to the Perona-Malik paradox", in Advances in Computer Vision (F. Solina, W. G. Kropatsch, R. Klette, and R. Bajcsy, eds.), pp. 1-10, Springer, Wien, 1997.

[737] J. Weickert, "Nonlinear diffusion scale-spaces", in Gaussian Scale-Space Theory (J. Sporring, M. Nielsen, L. Florack, and P. Johansen, eds.), pp. 221-234, Dordrecht: Kluwer, 1997.

[738] J. Weickert, K. J. Zuiderveld, B. M. ter Haar Romeny, and W. J. Niessen, Parallel implementations of AOS schemes: A fast way of nonlinear diffusion filtering", in Proc. 1997 IEEE Int. Conf. Image Processing, vol. 3, (ICIP-97, Santa Barbara, Oct. 26-29, 1997), pp. 396-399, 1997.

[739] J. Weickert, "A review of nonlinear diffusion filtering", in Scale-Space Theory in Computer Vision (B. ter Haar Romeny, L. Florack, J. Koenderink, and M. Viergever, eds.), vol. 1252 of Lecture Notes in Computer Science, pp. 3-28, Springer, Berlin, 1997.

[740] J. Weickert, "Coherence-enhancing diffusion of colour images", A. Sanfeliu, J.J. Villanueva, J. Vitrià (Eds.), Proc. VII National Symposium on Pattern Recognition and Image Analysis (VII NSPRIA, Barcelona, April 21-25, 1997), Vol. 1, 239-244, 1997.

[741] J. Weickert, B. M. ter Haar Romeny, A. Lopez, and W. J. van Enk, "Orientation analysis by coherence-enhancing diffusion", in Proc. Symposium on Real World Computing, (RWC '97, Tokyo, Jan. 29-31, 1997), pp. 96-103, 1997.

[742] J. Weickert, "Anisotropic diffusion in image processing", ECMI Series, Teubner Verlag, Stuttgart, 1998. ISBN 3-519-02606-6.

[743] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering", IEEE Tr. on Image Processing, Vol. 7, 398-410, 1998.

[744] J. Weickert, "Fast segmentation methods based on partial differential equations and the watershed transformation", P. Levi, R.-J. Ahlers, F. May, M. Schanz (Eds.), Mustererkennung 1998, Springer, Berlin, 93-100, 1998.

[745] J. Weickert, "On discontinuity-preserving optic flow", S. Orphanoudakis, P. Trahanias, J. Crowley, N. Katevas (Eds.), Proc. Computer Vision and Mobile Robotics Workshop (CVMR '98, Santorini, Sept. 17-18, 1998), 115-122, 1998.

[746] J. Weickert, S. Ishikawa, A. Imiya, "Linear scale-space has first been proposed in Japan", J. Mathematical Imaging and Vision, Vol. 10, 237-252, 1999.

[747] J. Weickert, "Nonlinear diffusion filtering", B. Jähne, H. Haußecker, P. Geißler (Eds.), Handbook on Computer Vision and Applications, Vol. 2: Signal Processing and Pattern Recognition, Academic Press, San Diego, 423-450, 1999.

[748] J. Weickert, J. Heers, C. Schnörr, K.J. Zuiderveld, O. Scherzer, H.S. Stiehl, "Fast parallel algorithms for a broad class of nonlinear variational diffusion approaches", Real-Time Imaging, 2001. Revised version of Technical Report 5/1999, Computer Science Series, University of Mannheim, 68131 Mannheim, Germany, 1999.

[749] J. Weickert, "Coherence-enhancing diffusion filtering", Intern. Journal of Computer Vision, Vol. 31, 111-127, 1999.

[750] J. Weickert, "Coherence-enhancing diffusion of colour images", Image and Vision Computing, Vol. 17, 201-212, 1999.

[751] J. Weickert, "Design of nonlinear diffusion filters", B. Jähne, H. Haußecker (Eds.), Computer Vision and Applications, Academic Press, San Diego, 439-458, 2000.

[752] J. Weickert, C. Schnörr, "PDE-based preprocessing of medical images", Künstliche Intelligenz, No. 3, 5-10, 2000. Revised version of Technical Report 8/2000, Computer Science Series, University of Mannheim, 68131 Mannheim, Germany, February 2000.

[753] J. Weickert, C. Schnörr, "Variational optic flow computation with a spatio-temporal smoothness constraint", Journal of Mathematical Imaging and Vision, 2001. Revised version of Technical Report 15/2000, Computer Science Series, University of Mannheim, 68131 Mannheim, Germany, July 2000.

[754] J. Weickert, "Efficient image segmentation using partial differential equations and morphology", Pattern Recognition, 2001. Also available as Technical Report 3/2000, Computer Science Series, University of Mannheim, 68131 Mannheim, Germany, February 2000.

[755] J. Weickert, H. Scharr, "A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance", Journal of Visual Communication and Image Representation, 2001. Revised and shortened version of Technical Report 4/2000, Computer Science Series, University of Mannheim, 68131 Mannheim, Germany, February 2000.

[756] R. Weitzenböck, "Invariantentheorie". Groningen: P. Noordhoff, 1923.

[757] G. B. West. "Scale and Dimension - From animals to quarks". In: Particle Physics, a Los Alamos Primer. N.C. Cooper and G. B. West, Eds. Cambridge University Press, Cambridge 1988.

[758] H. Weyl, "The Classical Groups, their Invariants and Representations". Princeton, NJ: Princeton University Press, 1946.

[759] H. Weyl, "Symmetry". Princeton, NJ: Princeton University Press, 1983 (reprint of 1952).

[760] R. T. Whitaker and S. M. Pizer, "A multi-scale approach to nonuniform diffusion", Tech. Rep. TR91-040, Medical Image Display Group, Department of Radiation Oncology, The University of North Carolina, Chapel Hill, NC 27599-3175, September 1991.

[761] R. T. Whitaker and S. M. Pizer, "A multi-scale approach to nonuniform diffusion", Computer Vision, Graphics, and Image Processing: Image Understanding, vol. 57, pp. 99-110, January 1993.

[762] R. T. Whitaker, "Geometry-limited diffusion in the characterization of geometric patches in images", Computer Vision, Graphics, and Image Processing: Image Understanding, vol. 57, pp. 111-120, January 1993.

[763] R. T. Whitaker and S. M. Pizer, "Geometry-based image segmentation using anisotropic diffusion", in Proc. of the NATO Advanced Research Workshop Shape in Picture -- Mathematical Description of Shape in Greylevel Images (Y.-L. O, A. Toet, H. J. A. M. Heijmans, D. H. Foster, and P. Meer, eds.), vol. 126 of NATO ASI Series F, pp. 641-650, Springer Verlag, Berlin, 1994.

[764] R. Whitaker and G. Gerig, "Vector-valued diffusion", in Geometry-Driven Diffusion in Computer Vision (B. M. ter Haar Romeny, ed.), Computational Imaging and Vision, pp. 93-134, Kluwer Academic Publishers, 1994.

[765] D. J. Williams and M. Shah, "Edge contours using multiple scales", Computer Vision, Graphics, and Image Processing, vol. 51, pp. 256-274, 1990.

[766] R. W. Williams, "The human retina has a cone-enriched rim", Visual Neuroscience, vol. 6, pp. 403-406, 1991.

[767] R. Wilson and A. H. Bhalerao, "Kernel design for efficient multiresolution edge detection and orientation estimation", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 14, no. 3, pp. 384-390, 1992.

[768] A. G. Wilson and V. E. Johnson, "Priors on scale-space templates", in Proc. Mathematical Methods in Medical Imaging II, vol. 2299, (San Diego, CA), pp. 161-168, SPIE, July, 25-26 1994.

[769] R. A. Wilson and F. Keil, "The MIT Encyclopedia of the Cognitive Sciences", the MIT Press, 1999.

[770] A. Witkin, "Scale-space filtering", in Proc. Intern. Joint Conf. on Artificial Intelligence, (Karlsruhe, Germany), pp. 1019-1023, 1983.

[771] A. Witkin, "Scale-space filtering: A new approach to multi-scale description", in Image Understanding 1984 (S. Ullman and W. Richards, eds.), NJ: Norwood Ablex, 1984.

[772] A. Witkin, D. Terzopoulos, and M. Kass, "Signal matching through scale-space", Intern. Journal of Computer Vision, vol. 1, no. 2, pp. 134-144, 1988.

[773] S. Wolfram, "*Mathematica*: A System for doing Mathematics by Computer", Addison-Wesley, 1999. Version 4.

[774] S. Wolfram, "A new kind of science", Addison-Wesley, 2002.

[775] G. Wyszecki and W. S. Stiles, "Color science: concepts and methods, quantitative data and formulae", Wiley Series in Pure and Applied Optics, 2000.

[776] R. A. Young, "Oh say can you see? the physiology of vision", publication GMR-7364, General Motors Research Labs, Computer Science Dept., 30500 Mound Road, Box 9055, Warren, Michigan 48090-9055, May 31 1991.

[777] R. A. Young, "The Gaussian derivative theory of spatial vision: Analysis of cortical cell receptive field line-weighting profiles", publication GMR-4920, General Motors Research Labs, Computer Science Dept., 30500 Mound Road, Box 9055, Warren, Michigan 48090-9055, May 28 1985.

[778] R. A. Young, "The Gaussian derivative model for machine vision: Visual cortex simulation", publication GMR-5323, General Motors Research Labs, Computer Science Dept., 30500 Mound Road, Box 9055, Warren, Michigan 48090-9055, July 7 1986.

[779] R. A. Young, "Simulation of human retinal function with the Gaussian derivative model", in Proc. IEEE CVPR CH2290-5, (Miami, Fla.), pp. 564-569, 1986.

[780] R. A. Young, "The Gaussian derivative model for machine vision: I. retinal mechanisms", Spatial Vision, vol. 2, no. 4, pp. 273-293, 1987.

[781] A. L. Yuille and T. Poggio, "Fingerprint theorems for zero crossings", JOSA, "A", vol. 2, pp. 683-692, May 1985.

[[782] A. L. Yuille and T. A. Poggio, "Scaling theorems for zero-crossings", IEEE Tr. on Pattern Analysis and Machine Intelligence, vol. 8, pp. 15-25, January 1986.

[783] A. L. Yuille and T. A. Poggio, "Scaling and fingerprint theorems for zero-crossings", in Advances in Computer Vision (C. Brown, ed.), pp. 47-78, Lawrence Erlbaum, 1988.

[784] J. C. Zagal, E. Björkman, T. Lindeberg, P. E. Roland, "Significance determination for the scale-space primal sketch by comparison of statistics of scale-space blob volumes computed from PET signals versus residual noise", HBM'2000, Intern. Conf. on Functional Mapping of the Human Brain, San Antonio, Texas, in press, 2000.

[785] S. Zeki, "A vision of the brain". Oxford: Blackwell Scientific Publications. 1993.

[786] J. Zhang and J. P. Miller, "A mathematical model for resolution enhancement in layered sensory systems", Biological Cybernetics, vol. 64, pp. 357-364, 1991.

[787] S. W. Zucker and R. A. Hummel, "Receptive fields and the representation of visual information", in Seventh Intern. Conf. on Pattern Recognition (Montreal, Canada, July 30-August 2, 1984), IEEE Publ. 84CH2046-1, pp. 515-517, IEEE, IEEE, 1984.

[788] S. W. Zucker, "Early orientation selection: Tangent fields and the dimensionality of their support", Computer Vision, Graphics, and Image Processing, vol. 32, pp. 74-103, 1985.

[789] S. W. Zucker and R. A. Hummel, "Receptive fields and the representation of visual information", Human Neurobiology, vol. 5, pp. 121-128, 1986.

[790] S. W. Zucker, "Which computation runs in visual cortical columns?", in J. Leo van Hemmen and T. J. Sejnowski (eds.), Problems in Systems Neuroscience, Oxford University Press, 2001.

Index

3D, 424
3D rendering, 91
3D ultrasound, follicle detection, 225
  probe, 225

abort evaluation in Mathematica, 423
accuracy, of data representation, 281
accuracy of differentiation, 137
acuity, 171
adaptive filtering, 184
affine invariant, corner detection, 113
affine transformation, 113
algebraic polynomial invariants, 92
aliasing, 11
Alvarez, Luis, 378
Alvarez' equation, 109
amacrine cell, 157, 288
angular weight, 334
animation, 75, 423
anisotropy, 48
anisotropy ratio, 68
annihilation, 255
anti-aliasing, 11
anti-symmetric tensor, 99
aperture, measurement, 2
  temporal, 1, 345
aperture problem, 309
aperture property, 309
apertures, 1
arbitrary precision mode, 87
area 17, 185
astigmatic eye, 48
automatic numbering objects, 424
automatic scale selection, 216, 219
autoradiogram, 190
average grey level invariance, 38
axioms, 15

basilar membrane, 161
basis, 92
Benford's Law, 352
bifurcation, 249
binomial coefficients, 43
biomimicking, 200
biphasic cell, 356
biphasic index, 358
bipolar cells, 157
black body light source, 312
blob detector, 216
blobness, 218
blobs, 190
block function, 47
blood oxygenation level, 193
blurring, 33
BOLD fMRI, 193
boundaries, 83
brackets in Mathematica, 396
Brodmann's area, 185

calcarine sulcus, 185
calculus of variations, 26, 148
Cartesian tensor, 92
cascade property, 39, 151
Cassini function, 94
Castan filter, 149
catastrophe, 241
cusp, 249
  detection, 268
  fold, 248, 264
  swallowtail, 250
catastrophe germ, 245
catastrophy theory, 243
catchment basin, 232, 234
causality, 23
  temporal, 350
CCD element, 32
CD-ROM content, 419
cell internal structure, 424
central limit theorem, 46, 277
central nervous system, 161
cerebellum, 180
changing the 3D viewpoint, 424
characteristic equation, 119
Chebyshev polynomials, 57
checkerboard junction, 131
cylindrical shapes, 117
coincidence detector, 287
coincidenceness,
  of roots of a polynomial, 131
color, image formation, 311
color constancy problem, 313
color model, by Koenderink, 314
columnar structure, 185
commutation, 28
Compile, 86, 88
complex conjugate, 148
complex cells, 187–188
computational models, for vision, 155
computer aided analysis, 3D ultrasound,
  225
computer-aided diagnosis (CAD), 338
concave, 117
condition number, 307
conductivity coefficient, 364
cones, 157
  wavelength sensitivity, 158

constant motion flow, 390
constraint, 148
constraint equations, 26
context of voxels, 34
contextual structure, 154
contour linking, 268
contour on the image, 423
ContourPlot, 93
contraction, of tensor indices, 135
control parameter, 243
convex, 117
convolution, 25, 413
cyclic, 72
  in Fourier domain, 141
  in the Fourier domain, 416
convolution integral, 46
convolution integral, 29, 415
convolve, 39
cooking of a turkey, 16
coordinate systems, 96
cornea, 48
corner detection, 114
  affine invariant, 113
cornerness, 114
coronal plane, 406
correlation, of Gaussian derivatives, 60
correlation coefficient, 61
cortical activity map, 194
cortical columns, 187
cortical hypercolumn, 190
cortical receptive fields,
  Gabor function model, 65
cortical simple cell, 354
corticofugal, 184
corticofugal projections, to the LGN, 183
Craik-O'Brien-Cornsweet illusion, 175
creation, 255
crest line, 125
critical isophote, 93
  self-intersection, 262
CT scanner, 2
cubic splines, 144
cumulative Gaussian function, 42
curvature, 23
curve evolution, 274, 382
curvedness, 120
cusp catastrophe, evolution, 249
cyclotron, 194

Damon, James, 250
data smoothing, 144
deblurring, 371
  analytical methods, 277
  Gaussian blur, 277
  scale-space approach, 277
deep structure, 154
  toolbox, 265
deep structure, 35, 167, 215, 223
Default3D, 424
deformable templates, 144
deformation, of images, 289
degenerate Hessian, 244
Delta-Dirac function, 42
dendrites, 169
dendritic tree size, 170
dendritic tree area, as function of
  eccentricity, 170
density image, optic flow, 292
Deriche filter, 150
derivative, directional, 102
  Fourier transform, 138
derivatives, in the visual system, 156
  of sampled data, 27
Descartes, 97
detection of maxima, 216
difference of Gaussians, 175
differential operators, generic events in
  scale-space, 250
differential features, 29
differential geometry, 31, 91
differential operator, temporal, 346
differential structure, 91
differentiation, by integration, 28
diffuse boundaries, 78
diffusion, anisotropic, 363
  coherence enhancing, 363
  geometry-driven, 361, 363
  inhomogeneous, 363
  isotropic, 363
  linear, 363
  tensor-driven, 363
  variable conductance, 363
diffusion equation, 49, 51
dilation, 387
dimension matrix, 18
dimensional analysis, 15
dimensionless, coordinate, 218
  numbers, 20
  quantities, 18
diopter, 156
Dirac delta function, 29, 41
direction, of vector, 330
directional derivative, 102, 129
Dirichlet problem, 67
discrete Gaussian kernels, 64
discriminant, 131
disparity map, 285
display a group of images, 424
dissimilarity measure, 231
dissimilarity scale-space, 233

distribution, 30
distribution theory, 147
distributions, regular, tempered, 144
dithering, 5
  color, 6
Div, divergence operator, 294
divergence of flow, 365
divergence theorem, 257
Dorst, Leo, 390
Dot, 100
draw a contour, 423
drift velocity, of singularities, 265
duality, 95
  PDE and curve evolution, 383
dumb-bell example, 255
dye injection, 170

eccentricity, retinal, 159, 170
edge focusing, 221
  implementation, 224
edge hierarchy, 78
edge location, 223
edge-preserving smoothing, 184, 362
EEG, 191
Eigenvalues, 119
Einstein summation convention, 135
electro-encephalography (EEG), 191
elementary catastrophes, 246
end-stopped cells, 187
energy minimization, 144
ensemble measurements, 154
entropy, 26
entropy scale-space, 383
epileptic foci, 192
Epilog, 423–424
erosion, 387
error function, 41
Euclidean shortening flow, 378, 383
Euclidean tranformation, 101
Euler gamma function, 62
Euler-Lagrange equations, 26, 148
EuroRad, 76, 102
Evaluate, 88
evolution, of images, 361
evolutionary computing, 361
exact representation, 86
excitatory orientation coupling, 200
extremal line, 125
extremal mesh, 126
extremality, 124
eye physics, 155–156

Fast Fourier Transform, 89
Fast summation, 424
features, 91
  differential, 29
feedback, 183
feedback loops, 67
field of view, 2
fill-in effect, 201
filter, Castan, 149
  Deriche, 150
  normalized, 25
  unity, 25
filterbank of oriented filters, 184
filterbank representation, 197
fingerprint, 253
  of fold catastrophe, 249
fingerprint enhancement, 239
Fit, 229
fitting spherical harmonics to 3D points, 229
fixing the gauge, 103
flowline, 92
  curvature, 110
fluid density, 18
fluid viscosity, 18
fluorescence, 194
fMRI, 193
  high resolution, 193
Fold, 235
fold catastrophe, 264
fold catastrophe, evolution, 247
follicle, 225
follicle boundary, 228
follicle detection in 3D ultrasound, 225
font style, 424
foreground-background illusion, 182
forward=Euler approximation, 365
Fourier, Jean-Baptiste, 15
Fourier domain, boundary effects, 83
Fourier transform, Gaussian derivative, 28
  leakage, folding, 139
fourth order structure, 131
fractals, 35
frames, 345
Freeman's lab, 162
friction factor, 19
Frobenius norm, 307
front-end, 4
  axioms, 15
  multi-scale geometry engine, 155
  uncommitted, 9
front-end visual system,
  cortical columns, 197
Froude's number, 20
function fitting, 144
functional, 148
functional imaging, 155
functional MRI (fMRI), 193

fundamental equation, 109

Gabor kernels, 65, 147
Gamma function, 139
ganglion cells, midget, 169
  parasol, 169, 288
Ganzfeld illumination, 160
gauge, first order, 103
gauge coordinates, first order, 103
  for discrete images, 107
gauge frame, plot, 106
gauge invariants, 134
Gauß, Carl Friedrich, 37
Gaussian derivative, implementation,
  N-dimensional, 78
Gaussian derivatives, correlation, 60
  Fourier domain, 57
  implementation, Fourier domain, 79
  separability, 67
  zero crossings, 56
Gaussian extremality, 125
Gaussian curvature, 123, 327
  minimal surfaces, 126
Gaussian derivative family, 27
Gaussian derivatives, 24, 53
  algebraic structure, 53
  as bandpass filters, 58
  as smooth test functions, 145
  functions graphs, 74
  implementation, recursive, 85
  implementation, separable, 73
  implementation, spatial domain, 71
  of infinite order, 60
  orthogonality, 57
  Szego's formula, 60
  Zernickes formula, 60
  zero crossings, 59
Gaussian envelop, 55
Gaussian kernel, 37, 51
  axiomatic derivation, 20
  discrete, 64
  Fourier transform, 44
  Fourier spectra, 46
  half width at half maximum, 38
  normalization, 38
  normalization factor, 55
  orientation, 331
  spectral, 315
  temporally skewed, 353
  variance, 37
  width, 37
Gaussian noise, 282, 424
Gaussian weighted extent, 40
gDc, 72
gDf, 80
gDn, 79
gDϕ, 336
gDϕN, 337
generalized functions, 40
generalized anisotropic non-linear
  diffusion, 240
generate an animation, 423
generic event, 246
geometric information, 29
geometric reasoning, 92, 127, 130, 361
geometrical structure, 154
geometry-driven diffusion, 109, 361
Giuseppe Arcimboldo, 9
global structure, 271
Grad, gradient operator, 294
gradient, integral curve, 95
  magnitude, 101
gradient magnitude, 221
gradient magnitude squared, 231
gradient scale-space, 222
gradient squared, generic events, 251
gradient vector, 100
graduated convexity, 30, 144
graph theory, 274
Gray, Alfred, 126
grayscale invariance, 379
grayscale transformation, 93
Green's function, 24
group, of transformations, 97
group theory, 351

Hadamard, Jacques, 143
half axis, 349
Hausdorff dimension, 34, 220, 263
Heavyside function, 42
heightline, 94
helical shell shape, 404
help browser, 423
heltocat surface, 127
Hering color basis, 316
Hermann grid illusion, 175
Hermite, Charles, 54
  polynomial, 55
Hermitian matrix, 101
Hessian matrix, 49
Hessian matrix, 119, 131
  determinant, 123
  Eigenvectors, 122
  trace, 124
hierachy, of structures, 223
Hilbert, David, 131, 134
Hildreth, 112
hills and dales, 117
histogram, 366
Histogram function, 424

histogram stretching, 78
histogram equalization, 77
histology, 325
horizontal cells, 157
Horn, Berthold, 290
horseradish peroxidase, 199
horseshoe crab, 4
Hubel, David, 160
human visual system, bibiography, 153
hypercolumn, 190, 197
interconnectivity, 199
hypercomplex cells, 187
hypersurface, 258
hypo-echoic, 225

identity matrix, 99
Iijima, 13
ill-posed, 30, 143, 370
illusion, Craik-O'Brien-Cornsweet, 175
  curvature of straight lines, 115
  foreground-background, 182
  Hermann grid, 175
  parallel lines, 330
  step function, 175
  Thatcher, 102
  visual fading, 200
image read, 423
image database indexing, 275
image differential structure, 91
image mosaic, 7
image partitioning, 231
image restoration, 277
implicit function theorem, 245
ImplicitPlot3D, 249
Import, 74
incommensurable dimensions, 345
inner scale, 3, 33, 37
installing the book, instructions, 419
  Windows 95/98/2000/NT/XP, 420
integration time, 345
integration area, 25
intensity surface, 95
interactive 3D display, 424
interactive segmentation, 231
interpolation, 144, 424
  cubic spline, 289
nearest neighbour, 11
interrupt, 423
intrinsic geometry, 103
introduction to Mathematica, 395
invariance, 100
  grayscale, 379
  scale, 15
invariant, complete family, 107
  expressed in gauge coordinates, 103
  name, 107
  to coordinate transformations, 92
invariant theory, 31
invariants, meaning of, 100
  tensor notation, 135
inverse heat equation, 370
inverse Fourier transform, 44
InverseFourier, 89
ipsi-lateral, 181
irreducible invariants, 134
iso-orientation contour, 199
isophote, 92
  3D, 95
  critical, 262
  curvature, 108, 110
  density, 112
  self-intersection, 94
isophote curvature, gradient of, 129
isophote curvature, 134
  generic events, 255
  MR image, 110
isophote imaging, 194
isotropic, 48, 68
isotropy, 15

Jacobi, Carl, 98
  polynomials, 57
Jacobian, matrix, 98, 294
Japan, 35
Jordan curve, 93
Josephson junction, 192
junction detection, 131

kernel, family, 27
  normalized, 22, 38
  width, 13
kernels, complete family, 35
keyhole observation, 34
knowledge incorporation, 361
Kuffler, Stephen, 160

Lagrangian, 26, 148
Laguerre polynomials, 57
Laplace's equation, 229
Laplacian, 216
  in gauge coordinates, 105
  in natural coordinates, 133
  of Gaussian, 172
  operator, 24, 101
  repeated, 278
  zero crossings, 112
Laplacian operator, Δ, 49
lateral inhibition, 175
Lateral Geniculate Nucleus, 162, 169, 179, 181

law of scale invariance, 16
least square approximation, 229
Legendre polynomials, 57
lemniscate of Bernouilli, 94
level sets, 382
Levi-Civita tensor, 99
LGN, Lateral Geniculate Nucleus, 169
LGN pathway, 186
Lie derivative, 293
limits on differentiation, 137
Lindeberg, Tony, 13
linear isotropic diffusion equation, 24
linearity, 15
linking, of contours, 268
linking of regions, 234
linking over scale, 215, 234
ListContourPlot3D, 95
ListConvolve, 71, 79
ListPlot3D, 74
LiveGraphics3D, 318, 424
local generic features, 33
localization scale, 236
log-polar mapping, 186
logarithmic half-axis sampling, 349
longevity, 223
low-pass filter, 46
Lvv, ridge detector, 109

macaque monkey, 170, 185
magnetic fields, 192
magneto-encephalography (MEG), 192
magno-cellular layers, LGN, 169, 181
Magritte, René, 83, 128
Malik, Jitendra, 363
manifolds, 91
manual shortcuts, 423
MapThread, 84
marching lines, 126
Marr, 112
matching, of images, 289
Mathematica, front-end, 395
  introduction, tutorial, 395
  kernel, 395
mathematical notation, 423
  speed concerns, 85
  startup suggestions, 423
  suggested reading, 410
  tips and tricks, 423
  web resources, 412
mathematical notation, 423
mathematical point, 40
mathematical morphology, 386
  grayvalued images, 389
  relation to gaussian scale-space, 390
mathematical point, 29
MathGL3d, OpenGL viewer, 95
MathSource, 412
matrix, inverse, 101
  transpose, 101
maximum principle, 50
maximum-length sequence, 163
MDL, minimum description length, 272
mean curvature, 123
MEG, 192
mesh, by isophotes and flowlines, 95
Mexican hat function, 172
midbrain, 179
midget ganglion cells, 169
minimal surface, 126
minimization, in L^2 sense, 152
minimum description length (MDL), 272
mirrored tiling, 84
model, 34, 92
modulation transfer function, 46
Moebius strip, 126
Mona Lisa, 7
monkey saddle, winding number, 259
monkeysaddle, higher order, 260
monosynaptic connection, 179
Morel, Jean-Michel, 362
morphological gradient operator, 388
Morse lemma, 245
Morse theory, 107
motion blur, 284
motion detection, with receptive field pairs, 286
MR brain segmentation, watershed, 238
multi-index, 145
multi-scale derivative operators, 30
multi-scale geometry engine, 155
multi-scale grouping, 273
multi-scale optic flow, 285
Multi-scale Optic Flow Constraint Equation (MOFCE), 286
multi-scale optic flow constraint equation, derivation, 292
multi-scale orientation, texture, 329
multi-scale segmentation, 231
multi-scale signature, of a signal, 223
multi-scale watershed segmentation, 237
multiplanar reconstruction, 277
multiplanar reformatting, 406
Mumford, David, 362

N-jet, 31
nabla operator, 101, 129
natural coordinates, 40, 132, 383
natural scale parameter, 32, 35
Nest, 104

neural activity in the brain, measurement, 191
Nielsen, Mads, 25
Niessen, Wiro, 373
noise, structure, 109
noise images, 424
noise is structure, 10
non-creation, of local extrema, 352
non-linear diffusion, for multi-scale watershed segmentation, 239
nonlinear diffusion, 109
norm, summation, 151
normal constraint, 292
normal distribution, 13
normal flow, constraint, 301
normal motion flow, 383
normalization, 38
normalized filter, 25
normalized derivative operator, 218
normalized derivatives, 218
normalized feature detection, 218
nullspace, 17

observation scale, 216
observations, 1
  critical view, 9
observed derivative, 137
occlusion boundary, 292
occlusion boundaries, 128
ocular dominance bands, 191, 197
off-center center-surround, 160
Olver, Peter, 384
on-center center-surround, 160
OpenGL viewer MathGL3d, 95
operator, differential, 13
  local, 2
  spatio-temporal, 347
operators, 13
optic radiation, 183
optic chiasm, 179
optic chiasma, 157
optic flow, 285
  scale selection, 307
optic flow constraint equation (OFCE), 290
optic nerve, 5
optical imaging, voltage sensitive dyes, 194
orderlessness, 26
organ of Corti, 161
orientation, column, 199
  of vector, 330
  sensitivity, 197
  sensitivity tuning, 197
  tuning curve, 189
orientation analysis, classical papers, 342
orientation bundle, 342
orthogonal polynomial, 229
orthogonal invariance, 103
orthogonal matrix, 101
orthogonal transformation, 101
orthonormal transformation, 101
Osher, Stanley, 362
Outer, 99
outer scale, 3
ovary, 225

packages, 423
palettes, 423
palindrome, 401
parabolic lines, 124
parallel lines illusion, 330
parasol ganglion cell, 169, 288
Parceval theorem, 148
partial differential equation, 361
partial volume effect, 12
parvo-cellular layers, 181
  LGN, 169
pattern matching, in Mathematica, 278
pattern matching, 401
  in Mathematica, 322
PDE's in computer vision, 184
perceptual grouping, 273
perceptual grouping, 184, 200
Perona, Pietro, 363
Perona and Malik, equation, 364
Perona-Malik diffusion scheme, 239
PET, 194
photopic vision, 158
physical dimensions, 15
physics of observation, 1, 21
Pi-theorem, 18
pinwheel, 198
  singularity, 199
piramidal datavolume, 225
pitfalls in Mathematica, 407
Planck's formula, 312
plot commands, 397
PlotGradientField, 48
PlotLegend, 64
pointspread function, 48, 74
Poisseuille coefficient, 20
Poisson scale-space, 67
positron, 194
positron emission tomography (PET), 194
post stimulus-time histogram, 164
post-synaptic potential, excitatory (EPSP), inhibitory (IPSP), 287
Powers of Ten, 4
prevent the textual output, 423

primal sketch, 271
primary visual cortex, 162, 185
principal curvature, 119
principal directions, 122
Printing Style Environment, 423
projective transformation, 101
proper print layout, 423
psychophysics, 5, 155
Puccini pressure-sensitive receptors, 161
pure function, 104, 401
pure functions, 424
pyramidal cell, 199

radioactive isotope, 194
radioactive tracer, 190
random numbers, 424
random dot stereogram, 8
read an image, 423
read binary 3D data, 405
RealTime3D, 424
receptive field, retinal, 154
sensitivity pattern, 160
receptive field, 4, 32, 156
  as smooth test function, 146
  center-surround, 173
  cortical, directional selectivity, 189
  diameter, 170
  dynamic, 165
real-time, 353
recording, 354
retina, 160
  sensitivity profile measurement, 162
  skin, 161
  spatio-temporal, 182
  static, 164
  temporal modulation, 188
  time-causal, model, 354
red-green edge detector, 324
region focusing, 235
region label, 233
region linking, 234
regular point, 244
regular tempered distribution, 145
regular tempered distributions, 144
regularization, 35, 143
  example, 147
  methods, 152
  Tikhonov, 148
regularization property, scale-space
  kernels, 277
Reichardt, Werner, 287
Reichardt detector, 286
  generalized, 288
Reichardt motion sensitive cell, 190
resizing graphics, 424
resolution, CT scanner, 2
  infinite, 9
  spatial, 2
  spurious, 11
  temporal, 2, 346
retina, 153
  as multi-scale sampling device, 177
  eccentricity, 170
  layers, 157
  sensitivity, 159
retinal sampling, scale-space model, 167
retinal receptive fields, scale-space model,
  172
retinotopic projection, 181
retrograde projection, 184
reverse correlation, 162
Reynold's number, 18
RGB color space, 317
rhodopsine, 159
ridge, 117
ridge detection, 108
ridgeness, 329
rods, 157
  electron-microscopic cross-section, 159
rods and cones, 32
RotationMatrix2D, 98
RotationMatrix3D, 98
rowing, 19
rut, 120

saddle point, 93, 117
saddle rut, 120
saddle shapes, 117
sagittal plane, 406
sampling property of derivatives , 41
sampling function, 41
sampling points, 423
sampling rate reduction, 33
Sapiro, Guillermo, 384
scalar, 100
scalar image, optic flow, 292
scale, exponential parametrization, 35
  temporal, 346
scale invariance, 15
scale selection, 216
  optic flow, 307
scale invariance, 132, 151
scale parameter, 40
scale selection, 215
scale-adptive system, 215
scale-invariance, 32, 167
scale-space, 8
  foundations, 13
  from causality, 23
  from entropy, 25

history, 13
regularization properties, 148
stack, 168
temporal, 346
scale-space theory, linear, 9
scale-space saddle, 272
scale-space stack, 31
scale-space theory, theory of apertures, 144
scale-step, in nonlinear diffusion schemes, 384
scale-time, 346
scaling laws, 20
Schwartz, 30, 41
Laurent, 144, 146
Schwartz space, 144
scotopic vision, 158
Screen Style Environment, 423
second order invariants, color mapping, 118
second order structure, 117
segmentation, 216
need, 91
self similar functions, 35
self-similar function, 39
self-similarity, 32
selfsimilarity, 39
semi-axis, 349
semicolon, as separator, 92
sensitivity profile, 28
separability, 43
Separability, 43
separatrix, 272
sequence of images, 423
Series, Taylor expansion, 97
Sethian, James, 362
shading, 313
shadows, 78
Shah, Jayant, 362
shape operator, 119
shape graph, 120
shape index, 120
shape-context operator, 274
shear transformation, 113
shift-integral, 415
shocks, 274
shortening flow, 386
shortnotation, 105
signal-to-noise ratio, 368
signature, of a signal, 223
signature function, noisy edge, 223
simple cells, 187
simple cells, 187
simultaneous scales observation, 167
single cell recordings, 191
singularities, 220
singularity, 94, 107, 241
drift velocity, 265
evolution in scale-space, 242
singularity points, annihilation, 223
size of notebooks, 423
skewness, 356
smooth test function, 30, 144
smoothing data, 144
smoothing property, 39
snakes, 144
somatotopic projection, 179
spatial frequency, 21
spatial shift invariance, 15
spatio-temporal operator, 347
special transformation, 99
special orthogonal transformation, 101
special plot forms, 404
spectral reflection function, 312
spectrum, 46
speed concerns, 85
spherical harmonics, to second order, 229
splines, 30
splitting lemma, 245, 248
spurious resolution, 11, 30
stability, of Gaussian linear diffusion, 373
of numerical PDE's, 372
stabilized retinal images, 200
state parameter, 243
steerable filter, 331
steerable kernels, 329
steering, with Cartesian partial derivatives, 336
with self-similar functions, 332
stellate tumore detection, 338
step function illusion, 175
step-edge, 42
stepsize, in numerical PDE evolution, 365
stop, 423
structural equivalence, 244
structure, generation, 23
of noise, 109
second order, 117
structure matrix, 343, 363
structuring element, 386
subimage, 423
submatrix, 423
superficial structure, 215
swallowtail catastrophe, 250
symmetric tensor, 99

T-junction detection, 127
T-junction-likeniness, 128
Tannenbaum, Andy, 384
taxonomy, 187

Taylor expansion, 96, 278
tempered distributions, 144
temporal width, 345
tensor, contraction, 135
indices, 135
tensor analysis, 31
tensor notation, 135
test function, 30
text on graphics, 424
texture, multi-scale orientation, 329
thalamus, 179
Thatcher illusion, 102
theorem, Parceval, 148
theory of distributions, 41
thin plate splines, 30, 144
third order structure, 127
Thom, René, 245
Thom's theorem, 245
Tikhonov regularization, 144, 148
time-resolution, 350
time-sequences, pre-recorded, 346
Toeplitz matrix, 277
Tolansky's curvature illusion, 111
tomographic image, 130
topological organization, 198
topological number, in scale-space, 258
topological numbers, 257
topology, 31, 94
torus, 405
total edge strength, color, 322
transformation, affine, 113
  rotation, 100
  shear, 113
transformation group, 97
transformations, 96
transversal plane, 406
transversality, 247
tree structure, 215
triangle function, 47
trough, 120

ultrasound, 382
umbilical point, 119, 124, 263
uncommitted, 9, 238
uncommittment, 27
unit vector, 100
unitstep function, 42
unity filter, 25

V1, primary visual cortex, 185
van den Boomgaard, Rein, 390
variance, 40
variational calculus, 148
vector, orthogonal, 25
vectorfield, 293
vicinity, on- and off-center receptive fields, 170
visual system, self-organization, 190
visual acuity, 171
visual column, 198
visual context, 155
visual fading illusion, 200
visual front-end, 4
visual pathway, 4, 156, 179
visual system, color processing, 190
voltage sensitive dyes, 191, 194
Von Neumann stability criterion, 372
voting scheme, 235

warping, 289
watershed segmentation, MR brain, 238
watershed segmentation, 215, 232
Weickert, Joachim, 362
well-posed, 143
well-posed derivatives, 147
wetware, 154
Weyl, Herman, 100
Wiesel, Torsten, 160
winding number, 257, 259
3D, 226
  on 2D images, 260
Wolfram, Stephen, 395

yellow-blue edge detector, 324

zero crossings, Laplacian, 112
zero-crossing detection, 222
zooming, 33

δ-operator, 99
$\delta(x)$, 41
ϵ-operator, 99
λ-normalized scale selection, 220

Computational Imaging and Vision

1. B.M. ter Haar Romeny (ed.): *Geometry-Driven Diffusion in Computer Vision.* 1994 ISBN 0-7923-3087-0
2. J. Serra and P. Soille (eds.): *Mathematical Morphology and Its Applications to Image Processing.* 1994 ISBN 0-7923-3093-5
3. Y. Bizais, C. Barillot, and R. Di Paola (eds.): *Information Processing in Medical Imaging.* 1995 ISBN 0-7923-3593-7
4. P. Grangeat and J.-L. Amans (eds.): *Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine.* 1996 ISBN 0-7923-4129-5
5. P. Maragos, R.W. Schafer and M.A. Butt (eds.): *Mathematical Morphology and Its Applications to Image and Signal Processing.* 1996 ISBN 0-7923-9733-9
6. G. Xu and Z. Zhang: *Epipolar Geometry in Stereo, Motion and Object Recognition.* A Unified Approach. 1996 ISBN 0-7923-4199-6
7. D. Eberly: *Ridges in Image and Data Analysis.* 1996 ISBN 0-7923-4268-2
8. J. Sporring, M. Nielsen, L. Florack and P. Johansen (eds.): *Gaussian Scale-Space Theory.* 1997 ISBN 0-7923-4561-4
9. M. Shah and R. Jain (eds.): *Motion-Based Recognition.* 1997 ISBN 0-7923-4618-1
10. L. Florack: *Image Structure.* 1997 ISBN 0-7923-4808-7
11. L.J. Latecki: *Discrete Representation of Spatial Objects in Computer Vision.* 1998 ISBN 0-7923-4912-1
12. H.J.A.M. Heijmans and J.B.T.M. Roerdink (eds.): *Mathematical Morphology and its Applications to Image and Signal Processing.* 1998 ISBN 0-7923-5133-9
13. N. Karssemeijer, M. Thijssen, J. Hendriks and L. van Erning (eds.): *Digital Mammography.* 1998 ISBN 0-7923-5274-2
14. R. Highnam and M. Brady: *Mammographic Image Analysis.* 1999 ISBN 0-7923-5620-9
15. I. Amidror: *The Theory of the Moiré Phenomenon.* 2000 ISBN 0-7923-5949-6; Pb: ISBN 0-7923-5950-x
16. G.L. Gimel'farb: *Image Textures and Gibbs Random Fields.* 1999 ISBN 0-7923-5961
17. R. Klette, H.S. Stiehl, M.A. Viergever and K.L. Vincken (eds.): *Performance Characterization in Computer Vision.* 2000 ISBN 0-7923-6374-4
18. J. Goutsias, L. Vincent and D.S. Bloomberg (eds.): *Mathematical Morphology and Its Applications to Image and Signal Processing.* 2000 ISBN 0-7923-7862-8
19. A.A. Petrosian and F.G. Meyer (eds.): *Wavelets in Signal and Image Analysis.* From Theory to Practice. 2001 ISBN 1-4020-0053-7
20. A. Jaklič, A. Leonardis and F. Solina: *Segmentation and Recovery of Superquadrics.* 2000 ISBN 0-7923-6601-8
21. K. Rohr: *Landmark-Based Image Analysis.* Using Geometric and Intensity Models. 2001 ISBN 0-7923-6751-0
22. R.C. Veltkamp, H. Burkhardt and H.-P. Kriegel (eds.): *State-of-the-Art in Content-Based Image and Video Retrieval.* 2001 ISBN 1-4020-0109-6
23. A.A. Amini and J.L. Prince (eds.): *Measurement of Cardiac Deformations from MRI: Physical and Mathematical Models.* 2001 ISBN 1-4020-0222-X

Computational Imaging and Vision

24. M.I. Schlesinger and V. Hlaváč: *Ten Lectures on Statistical and Structural Pattern Recognition*. 2002 ISBN 1-4020-0642-X
25. F. Mokhtarian and M. Bober: *Curvature Scale Space Representation: Theory, Applications, and MPEG-7 Standardization*. 2003 ISBN 1-4020-1233-0
26. N. Sebe and M.S. Lew: *Robust Computer Vision: Theory and Applications*. 2003 ISBN 1-4020-1293-8
27. B.M. ter Haar Romeny: *Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, written in Mathematica*. 2008 ISBN 978-1-4020-1503-8 (HB); 978-1-4020-1507-6 (PB)