

Wilhelm Burger · Mark J. Burge

Principles of Digital Image Processing

Advanced Methods

With 129 figures, 6 tables and 46 algorithms

Springer-Verlag

Berlin Heidelberg New York

London Paris Tokyo

Hong Kong Barcelona

Budapest

Preface

This is the 3rd volume of the authors' textbook series on *Principles of Digital Image Processing* that is predominantly aimed at undergraduate study and teaching:

Vol. 1: *Fundamental Techniques*,

Vol. 2: *Core Algorithms*,

Vol. 3: *Advanced Methods* (this volume).

While it builds on the previous two volumes and relies on their proven format, it contains all new material published by the authors for the first time. The topics covered in this volume are slightly more advanced and should thus be well suited for a follow-up undergraduate or Master-level course and as a solid reference for experienced practitioners in the field.

The topics of this volume range over a variety of image processing applications, with a general focus on “classic” techniques that are in wide use but are at the same time challenging to explore with the existing scientific literature. In choosing these topics, we have also considered input received from students, lecturers and practitioners over several years, for which we are very grateful. While it is almost unfeasible to cover all recent developments in the field, we focused on popular “workhorse” techniques that are available in many image processing systems but are often used without a thorough understanding of their inner workings. This particularly applies to the contents of the first five chapters on automatic thresholding, filters and edge detectors for color images, and edge-preserving smoothing. Also, an extensive part of the book is devoted to David Lowe's popular *SIFT* method for invariant local feature detection, which has found its way into so many applications and has become a standard tool in the industry, despite (as the text probably shows) its inherent sophistication and complexity. An additional “bonus chapter” on *Synthetic*

Gradient Noise, which could not be included in the print version, is available for download from the book's website.

As in the previous volumes, our main goal has been to provide *accurate*, *understandable* and *complete* algorithmic descriptions that take the reader all the way from the initial idea through the formal description to a working implementation. This may make the text appear bloated or too mathematical in some places, but we expect that interested readers will appreciate the high level of detail and the decision not to omit the (sometimes essential) intermediate steps. Wherever reasonable, general prerequisites and more specific details are summarized in the Appendix, which should also serve as a quick reference that is supported by a carefully compiled index. While space constraints did not permit the full source code to be included in print, complete (Java) implementations for each chapter are freely available on the book's website (see below). Again we have tried to make this code maximally congruent with the notation used in the text, such that readers should be able to easily follow, execute and extend the described steps.

Software

The implementations in this book series are all based on Java and ImageJ, a widely used programmer-extensible imaging system developed, maintained, and distributed by Wayne Rasband of the National Institutes of Health (NIH).¹ ImageJ is implemented completely in Java and therefore runs on all major platforms. It is widely used because its “plugin”-based architecture enables it to be easily extended. Although all examples run in ImageJ, they have been specifically designed to be easily ported to other environments and programming languages. We chose Java as an implementation language because it is elegant, portable, familiar to many computing students, and more efficient than commonly thought. Note, however, that we incorporate Java purely as an instructional vehicle because precise language semantics are needed eventually to achieve ultimate clarity. Since we stress the simplicity and readability of our programs, this should not be considered production-level but “instructional” software that naturally leaves vast room for improvement and performance optimization. Consequently, this book is not primarily on Java programming nor is it intended to serve as a reference manual for ImageJ.

Online Resources

In support of this book series, the authors maintain a dedicated website that provides supplementary materials, including the complete Java source code,

¹ <http://rsb.info.nih.gov/ij/>.

the test images used in the examples, and corrections. Readers are invited to visit this site at

www.imagingbook.com

It also makes available additional materials for educators, including a complete set of figures, tables and mathematical elements shown in the text, in a format suitable for easy inclusion in presentations and course notes. Also, as a free add-on to this volume, readers may download a supplementary “bonus chapter” on synthetic noise generation. Any comments, questions, and corrections are welcome and should be addressed to

imagingbook@gmail.com

Acknowledgements

As with its predecessors, this volume would not have been possible without the understanding and steady support of our families. Thanks go to Wayne Rasband (NIH) for continuously improving ImageJ and for his outstanding service to the imaging community. We appreciate the contributions from the many careful readers who have contacted us to suggest new topics, recommend alternative solutions, or to suggest corrections. A special debt of gratitude is owed to Stefan Stavrev for his detailed, technical editing of this volume. Finally, we are grateful to Wayne Wheeler for initiating this book series and Simon Rees and his colleagues at Springer’s UK and New York offices for their professional support, for the high quality (full-color) print production and the enduring patience with the authors.

Hagenberg, Austria / Washington DC, USA
January 2013

Contents

Preface	v
1. Introduction	1
2. Automatic Thresholding	5
2.1 Global histogram-based thresholding	6
2.1.1 Statistical information from the histogram	8
2.1.2 Simple threshold selection	10
2.1.3 Iterative threshold selection (ISODATA algorithm)	11
2.1.4 Otsu's method	14
2.1.5 Maximum entropy thresholding	18
2.1.6 Minimum error thresholding	22
2.2 Local adaptive thresholding	30
2.2.1 Bernsen's method	30
2.2.2 Niblack's method	34
2.3 Java implementation	45
2.4 Summary and further reading	49
2.5 Exercises	50
3. Filters for Color Images	51
3.1 Linear filters	51
3.1.1 Using monochromatic linear filters on color images	52
3.1.2 Color space considerations	55
3.2 Non-linear color filters	66
3.2.1 Scalar median filter	66
3.2.2 Vector median filter	67

3.2.3	Sharpening vector median filter	69
3.3	Java implementation	76
3.4	Further reading	80
3.5	Exercises	80
4.	Edge Detection in Color Images	83
4.1	Monochromatic techniques	84
4.2	Edges in vector-valued images	88
4.2.1	Multi-dimensional gradients	88
4.2.2	The Jacobian matrix	93
4.2.3	Squared local contrast	94
4.2.4	Color edge magnitude	95
4.2.5	Color edge orientation	97
4.2.6	Grayscale gradients revisited	99
4.3	Canny edge operator	103
4.3.1	Canny edge detector for grayscale images	103
4.3.2	Canny edge detector for color images	105
4.4	Implementation	115
4.5	Other color edge operators	116
5.	Edge-Preserving Smoothing Filters	119
5.1	Kuwahara-type filters	120
5.1.1	Application to color images	123
5.2	Bilateral filter	126
5.2.1	Domain vs. range filters	128
5.2.2	Bilateral filter with Gaussian kernels	131
5.2.3	Application to color images	132
5.2.4	Separable implementation	136
5.2.5	Other implementations and improvements	141
5.3	Anisotropic diffusion filters	143
5.3.1	Homogeneous diffusion and the heat equation	144
5.3.2	Perona-Malik filter	146
5.3.3	Perona-Malik filter for color images	149
5.3.4	Geometry-preserving anisotropic diffusion	156
5.3.5	Tschumperlé-Deriche algorithm	157
5.4	Measuring image quality	161
5.5	Implementation	164
5.6	Exercises	165

6. Fourier Shape Descriptors	169
6.1 2D boundaries in the complex plane	169
6.1.1 Parameterized boundary curves	169
6.1.2 Discrete 2D boundaries	170
6.2 Discrete Fourier transform	171
6.2.1 Forward transform	173
6.2.2 Inverse Fourier transform (reconstruction)	173
6.2.3 Periodicity of the DFT spectrum	177
6.2.4 Truncating the DFT spectrum	177
6.3 Geometric interpretation of Fourier coefficients	179
6.3.1 G_0 corresponds to the contour's centroid	180
6.3.2 Coefficient G_1 corresponds to a circle	181
6.3.3 Coefficient G_m corresponds to a circle with frequency m	182
6.3.4 Negative frequencies	183
6.3.5 Fourier descriptor pairs correspond to ellipses	183
6.3.6 Shape reconstruction from truncated Fourier descriptors	187
6.3.7 Fourier descriptors from arbitrary polygons	193
6.4 Effects of geometric transformations	195
6.4.1 Translation	197
6.4.2 Scale change	199
6.4.3 Shape rotation	199
6.4.4 Shifting the contour start position	200
6.4.5 Effects of phase removal	201
6.4.6 Direction of contour traversal	203
6.4.7 Reflection (symmetry)	203
6.5 Making Fourier descriptors invariant	203
6.5.1 Scale invariance	204
6.5.2 Start point invariance	205
6.5.3 Rotation invariance	208
6.5.4 Other approaches	209
6.6 Shape matching with Fourier descriptors	214
6.6.1 Magnitude-only matching	214
6.6.2 Complex (phase-preserving) matching	218
6.7 Java implementation	219
6.8 Summary and further reading	225
6.9 Exercises	225
7. SIFT—Scale-Invariant Local Features	229
7.1 Interest points at multiple scales	230
7.1.1 The Laplacian-of-Gaussian (LoG) filter	231
7.1.2 Gaussian scale space	237

7.1.3	LoG/DoG scale space	240
7.1.4	Hierarchical scale space	242
7.1.5	Scale space implementation in SIFT	248
7.2	Key point selection and refinement	252
7.2.1	Local extrema detection	255
7.2.2	Position refinement	257
7.2.3	Suppressing responses to edge-like structures	260
7.3	Creating Local Descriptors	263
7.3.1	Finding dominant orientations	263
7.3.2	Descriptor formation	267
7.4	SIFT algorithm summary	276
7.5	Matching SIFT Features	276
7.5.1	Feature distance and match quality	285
7.5.2	Examples	287
7.6	Efficient feature matching	289
7.7	SIFT implementation in Java	294
7.7.1	SIFT feature extraction	294
7.7.2	SIFT feature matching	295
7.8	Exercises	296

Appendix

A.	Mathematical Symbols and Notation	299
B.	Vector Algebra and Calculus	305
B.1	Vectors	305
B.1.1	Column vectors and row vectors	306
B.1.2	Vector length	306
B.2	Eigenvectors and eigenvalues	306
B.2.1	Eigenvectors of a 2×2 matrix	307
B.3	Parabolic fitting	309
B.3.1	Fitting a parabolic function to three sample points	309
B.3.2	Parabolic interpolation	310
B.4	Vector fields	311
B.4.1	Jacobian matrix	312
B.4.2	Gradient	312
B.4.3	Maximum gradient direction	313
B.4.4	Divergence	314
B.4.5	Laplacian	314
B.4.6	The Hessian matrix	315
B.5	Operations on multi-variable, scalar functions (scalar fields)	316

B.5.1	Estimating the derivatives of a discrete function	316
B.5.2	Taylor series expansion of functions	316
B.5.3	Finding the continuous extremum of a multi-variable discrete function	320
C.	Statistical Prerequisites	325
C.1	Mean, variance and covariance	325
C.2	Covariance matrices	326
C.2.1	Example	327
C.3	The Gaussian distribution	328
C.3.1	Maximum likelihood	329
C.3.2	Gaussian mixtures	330
C.3.3	Creating Gaussian noise	331
C.4	Image quality measures	331
D.	Gaussian Filters	333
D.1	Cascading Gaussian filters	333
D.2	Effects of Gaussian filtering in the frequency domain	334
D.3	LoG-approximation by the difference of two Gaussians (DoG)	335
E.	Color Space Transformations	337
E.1	RGB/sRGB transformations	337
E.2	CIELAB/CIELUV transformations	338
E.2.1	CIELAB	339
E.2.2	CIELUV	340
	Bibliography	343
	Index	357