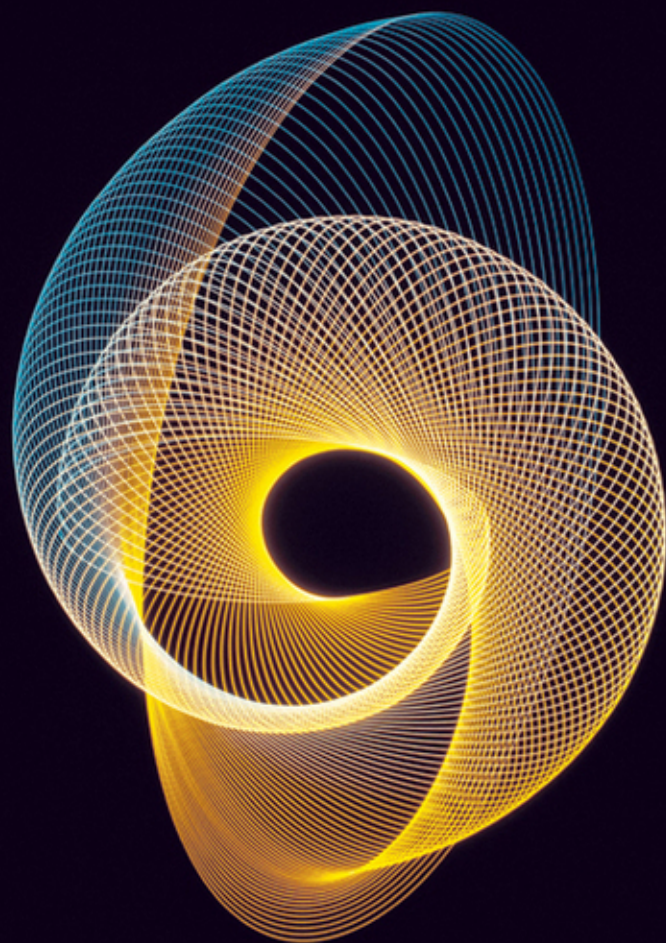


J. R. Parker

Algorithms for Image Processing and Computer Vision



SECOND EDITION

2

Algorithms for Image Processing and Computer Vision

Second Edition



Algorithms for Image Processing and Computer Vision

Second Edition

J.R. Parker



WILEY

Wiley Publishing, Inc.

Algorithms for Image Processing and Computer Vision, Second Edition

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2011 by J.R. Parker

Published by Wiley Publishing, Inc., Indianapolis, Indiana
Published simultaneously in Canada

ISBN: 978-0-470-64385-3
ISBN: 978-1-118-02188-0 (ebk)
ISBN: 978-1-118-02189-7 (ebk)
ISBN: 978-1-118-01962-7 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Web site is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2010939957

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in this book.

*"Sin lies only in hurting other people unnecessarily.
All other 'sins' are invented nonsense.
(Hurting yourself is not a sin – just stupid.)"*

— Robert A. Heinlein

Thanks, Bob.



Credits

Executive Editor

Carol Long

Project Editor

John Sleeva

Technical Editor

Kostas Terzidis

Production Editor

Daniel Scribner

Copy Editor

Christopher Jones

Editorial Director

Robyn B. Siesky

Editorial Manager

Mary Beth Wakefield

**Freelancer Editorial
Manager**

Rosemarie Graham

Marketing Manager

Ashley Zurcher

Production Manager

Tim Tate

**Vice President and Executive
Group Publisher**

Richard Swadley

**Vice President and Executive
Publisher**

Barry Pruett

Associate Publisher

Jim Minatel

Project Coordinator, Cover

Lynsey Stanford

Proofreaders

Nancy Hanger, Paul Sagan

Indexer

Ron Strauss

Cover Image

Ryan Sneed

Cover Designer

© GYRO PHOTOGRAPHY/
amanaimagesRB/Getty Images



About the Author

J.R. Parker is a computer expert and teacher, with special interests in image processing and vision, video game technologies, and computer simulations. With a Ph.D. in Informatics from the State University of Gent, Dr. Parker has taught computer science, art, and drama at the University of Calgary in Canada, where he is a full professor. He has more than 150 technical papers and four books to his credit, as well as video games such as the *Booze Cruise*, a simulation of impaired driving designed to demonstrate its folly, and a number of educational games. Jim lives on a small ranch near Cochrane, Alberta, Canada with family and a host of legged and winged creatures.



About the Technical Editor

Kostas Terzidis is an Associate Professor at the Harvard Graduate School of Design. He holds a Ph.D. in Architecture from the University of Michigan (1994), a Masters of Architecture from Ohio State University (1989), and a Diploma of Engineering from the Aristotle University of Thessaloniki (1986). His most recent work is in the development of theories and techniques for the use of algorithms in architecture. His book *Expressive Form: A Conceptual Approach to Computational Design*, published by London-based Spon Press (2003), offers a unique perspective on the use of computation as it relates to aesthetics, specifically in architecture and design. His book *Algorithmic Architecture* (Architectural Press/Elsevier, 2006) provides an ontological investigation into the terms, concepts, and processes of algorithmic architecture and provides a theoretical framework for design implementations. His latest book, *Algorithms for Visual Design* (Wiley, 2009), provides students, programmers, and researchers the technical, theoretical, and design means to develop computer code that will allow them to experiment with design problems.



Acknowledgments

Thanks this time to Sonny Chan, for the inspiration for the parallel computing chapter, to Jeff Boyd, for introducing me repeatedly to OpenCV, and to Ralph Huntsinger and Ghislain C. Vansteenkiste, for getting me into and successfully out of my Ph.D. program.

Almost all the images used in this book were created by me, using an IBM PC with a frame grabber and a Sony CCD camera, an HP scanner, and a Sony Eyetoy as a webcam. Credits for the few images that were not acquired in this way are as follows:

Corel Corporation made available the color image of the grasshopper on a leaf shown in Figure 3.33, and also was the origin of the example search images in Figure 10.5.

The sample images in Figure 10.1 were a part of the ALOI data set, use of which was allowed by J. M. Geusebroek.

Thanks to Big Hill Veterinary Clinic in Cochrane, Alberta, Canada, for the X-ray image shown in Figure 3.10e.

Finally, thanks to Dr. N. Wardlaw, of the University of Calgary Department of Geology, for the geological micropore image of Figure 3.16.

Most importantly, I need to thank my family: my wife, Katrin, and children, Bailey and Max. They sacrificed time and energy so that this work could be completed. I appreciate it and hope that the effort has been worthwhile.



Contents at a Glance

Preface	xxi
Chapter 1 Practical Aspects of a Vision System – Image Display, Input/Output, and Library Calls	1
Chapter 2 Edge-Detection Techniques	21
Chapter 3 Digital Morphology	85
Chapter 4 Grey-Level Segmentation	137
Chapter 5 Texture and Color	177
Chapter 6 Thinning	209
Chapter 7 Image Restoration	251
Chapter 8 Classification	285
Chapter 9 Symbol Recognition	321
Chapter 10 Content-Based Search – Finding Images by Example	395
Chapter 11 High-Performance Computing for Vision and Image Processing	425
Index	465



Contents

Preface	xxi
Chapter 1 Practical Aspects of a Vision System – Image Display, Input/Output, and Library Calls	1
OpenCV	2
The Basic OpenCV Code	2
The IplImage Data Structure	3
Reading and Writing Images	6
Image Display	7
An Example	7
Image Capture	10
Interfacing with the AIPCV Library	14
Website Files	18
References	18
Chapter 2 Edge-Detection Techniques	21
The Purpose of Edge Detection	21
Traditional Approaches and Theory	23
Models of Edges	24
Noise	26
Derivative Operators	30
Template-Based Edge Detection	36
Edge Models: The Marr-Hildreth Edge Detector	39
The Canny Edge Detector	42
The Shen-Castan (ISEF) Edge Detector	48
A Comparison of Two Optimal Edge Detectors	51

	Color Edges	53
	Source Code for the Marr-Hildreth Edge Detector	58
	Source Code for the Canny Edge Detector	62
	Source Code for the Shen-Castan Edge Detector	70
	Website Files	80
	References	82
Chapter 3	Digital Morphology	85
	Morphology Defined	85
	Connectedness	86
	Elements of Digital Morphology — Binary Operations	87
	Binary Dilation	88
	Implementing Binary Dilation	92
	Binary Erosion	94
	Implementation of Binary Erosion	100
	Opening and Closing	101
	MAX — A High-Level Programming Language for Morphology	107
	The “Hit-and-Miss” Transform	113
	Identifying Region Boundaries	116
	Conditional Dilation	116
	Counting Regions	119
	Grey-Level Morphology	121
	Opening and Closing	123
	Smoothing	126
	Gradient	128
	Segmentation of Textures	129
	Size Distribution of Objects	130
	Color Morphology	131
	Website Files	132
	References	135
Chapter 4	Grey-Level Segmentation	137
	Basics of Grey-Level Segmentation	137
	Using Edge Pixels	139
	Iterative Selection	140
	The Method of Grey-Level Histograms	141
	Using Entropy	142
	Fuzzy Sets	146
	Minimum Error Thresholding	148
	Sample Results From Single Threshold Selection	149

	The Use of Regional Thresholds	151
	Chow and Kaneko	152
	Modeling Illumination Using Edges	156
	Implementation and Results	159
	Comparisons	160
	Relaxation Methods	161
	Moving Averages	167
	Cluster-Based Thresholds	170
	Multiple Thresholds	171
	Website Files	172
	References	173
Chapter 5	Texture and Color	177
	Texture and Segmentation	177
	A Simple Analysis of Texture in Grey-Level Images	179
	Grey-Level Co-Occurrence	182
	Maximum Probability	185
	Moments	185
	Contrast	185
	Homogeneity	185
	Entropy	186
	Results from the GLCM Descriptors	186
	Speeding Up the Texture Operators	186
	Edges and Texture	188
	Energy and Texture	191
	Surfaces and Texture	193
	Vector Dispersion	193
	Surface Curvature	195
	Fractal Dimension	198
	Color Segmentation	201
	Color Textures	205
	Website Files	205
	References	206
Chapter 6	Thinning	209
	What Is a Skeleton?	209
	The Medial Axis Transform	210
	Iterative Morphological Methods	212
	The Use of Contours	221
	Choi/Lam/Siu Algorithm	224
	Treating the Object as a Polygon	226
	Triangulation Methods	227

Force-Based Thinning	228
Definitions	229
Use of a Force Field	230
Subpixel Skeletons	234
Source Code for Zhang-Suen/Stentiford/Holt Combined	
Algorithm	235
Website Files	246
References	247
Chapter 7 Image Restoration	251
Image Degradations — The Real World	251
The Frequency Domain	253
The Fourier Transform	254
The Fast Fourier Transform	256
The Inverse Fourier Transform	260
Two-Dimensional Fourier Transforms	260
Fourier Transforms in OpenCV	262
Creating Artificial Blur	264
The Inverse Filter	270
The Wiener Filter	271
Structured Noise	273
Motion Blur — A Special Case	276
The Homomorphic Filter — Illumination	277
Frequency Filters in General	278
Isolating Illumination Effects	280
Website Files	281
References	283
Chapter 8 Classification	285
Objects, Patterns, and Statistics	285
Features and Regions	288
Training and Testing	292
Variation: In-Class and Out-Class	295
Minimum Distance Classifiers	299
Distance Metrics	300
Distances Between Features	302
Cross Validation	304
Support Vector Machines	306
Multiple Classifiers — Ensembles	309
Merging Multiple Methods	309
Merging Type 1 Responses	310
Evaluation	311
Converting Between Response Types	312

Merging Type 2 Responses	313
Merging Type 3 Responses	315
Bagging and Boosting	315
Bagging	315
Boosting	316
Website Files	317
References	318
Chapter 9 Symbol Recognition	321
The Problem	321
OCR on Simple Perfect Images	322
OCR on Scanned Images — Segmentation	326
Noise	327
Isolating Individual Glyphs	329
Matching Templates	333
Statistical Recognition	337
OCR on Fax Images — Printed Characters	339
Orientation — Skew Detection	340
The Use of Edges	345
Handprinted Characters	348
Properties of the Character Outline	349
Convex Deficiencies	353
Vector Templates	357
Neural Nets	363
A Simple Neural Net	364
A Backpropagation Net for Digit Recognition	368
The Use of Multiple Classifiers	372
Merging Multiple Methods	372
Results From the Multiple Classifier	375
Printed Music Recognition — A Study	375
Staff Lines	376
Segmentation	378
Music Symbol Recognition	381
Source Code for Neural Net Recognition System	383
Website Files	390
References	392
Chapter 10 Content-Based Search — Finding Images by Example	395
Searching Images	395
Maintaining Collections of Images	396
Features for Query by Example	399
Color Image Features	399
Mean Color	400
Color Quad Tree	400

Hue and Intensity Histograms	401
Comparing Histograms	402
Requantization	403
Results from Simple Color Features	404
Other Color-Based Methods	407
Grey-Level Image Features	408
Grey Histograms	409
Grey Sigma — Moments	409
Edge Density — Boundaries Between Objects	409
Edge Direction	410
Boolean Edge Density	410
Spatial Considerations	411
Overall Regions	411
Rectangular Regions	412
Angular Regions	412
Circular Regions	414
Hybrid Regions	414
Test of Spatial Sampling	414
Additional Considerations	417
Texture	418
Objects, Contours, Boundaries	418
Data Sets	418
Website Files	419
References	420
Systems	424
Chapter 11 High-Performance Computing for Vision and Image Processing	425
Paradigms for Multiple-Processor Computation	426
Shared Memory	426
Message Passing	427
Execution Timing	427
Using <i>clock()</i>	428
Using <code>QueryPerformanceCounter</code>	430
The Message-Passing Interface System	432
Installing MPI	432
Using MPI	433
Inter-Process Communication	434
Running MPI Programs	436
Real Image Computations	437
Using a Computer Network — Cluster Computing	440

A Shared Memory System — Using the PC Graphics Processor	444
GLSL	444
OpenGL Fundamentals	445
Practical Textures in OpenGL	448
Shader Programming Basics	451
Vertex and Fragment Shaders	452
Required GLSL Initializations	453
Reading and Converting the Image	454
Passing Parameters to Shader Programs	456
Putting It All Together	457
Speedup Using the GPU	459
Developing and Testing Shader Code	459
Finding the Needed Software	460
Website Files	461
References	461
Index	465



Preface

Humans still obtain the vast majority of their sensory input through their visual system, and an enormous effort has been made to artificially enhance this sense. Eyeglasses, binoculars, telescopes, radar, infrared sensors, and photomultipliers all function to improve our view of the world and the universe. We even have telescopes in orbit (eyes outside the atmosphere) and many of those “see” in other spectra: infrared, ultraviolet, X-rays. These give us views that we could not have imagined only a few years ago, and in colors that we’ll never see with the naked eye. The computer has been essential for creating the incredible images we’ve all seen from these devices.

When the first edition of this book was written, the Hubble Space Telescope was in orbit and producing images at a great rate. It and the European Hipparcos telescope were the only optical instruments above the atmosphere. Now there is COROT, Kepler, MOST (Canada’s space telescope), and Swift Gamma Ray Burst Explorer. In addition, there is the Spitzer (infrared), Chandra (X-ray), GALEX (ultraviolet), and a score of others. The first edition was written on a 450-Mhz Pentium III with 256 MB of memory. In 1999, the first major digital SLR camera was placed on the market: the Nikon D1. It had only 2.74 million pixels and cost just under \$6,000. A typical PC disk drive held 100–200 MB. Webcams existed in 1997, but they were expensive and low-resolution. Persons using computer images needed to have a special image acquisition card and a relatively expensive camera to conduct their work, generally amounting to \$1–2,000 worth of equipment. The technology of personal computers and image acquisition has changed a lot since then.

The 1997 first edition was inspired by my numerous scans through the Internet news groups related to image processing and computer vision. I noted that some requests appeared over and over again, sometimes answered and sometimes not, and wondered if it would be possible to answer the more

frequently asked questions in book form, which would allow the development of some of the background necessary for a complete explanation. However, since I had just completed a book (*Practical Computer Vision Using C*), I was in no mood to pursue the issue. I continued to collect information from the Net, hoping to one day collate it into a sensible form. I did that, and the first edition was very well received. (Thanks!)

Fifteen years later, given the changes in technology, I'm surprised at how little has changed in the field of vision and image processing, at least at the accessible level. Yes, the theory has become more sophisticated and three-dimensional vision methods have certainly improved. Some robot vision systems have accomplished rather interesting things, and face recognition has been taken to a new level. However, cheap character recognition is still, well, cheap, and is still not up to a level where it can be used reliably in most cases. Unlike other kinds of software, vision systems are not ubiquitous features of daily life. Why not? Possibly because the vision problem is really a hard one. Perhaps there is room for a revision of the original book?

My goal has changed somewhat. I am now also interested in "democratization" of this technology — that is, in allowing it to be used by anyone, at home, in their business, or at schools. Of course, you need to be able to program a computer, but that skill is more common than it was. All the software needed to build the programs in this edition is freely available on the Internet. I have used a free compiler (Microsoft Visual Studio Express), and OpenCV is also a free download. The only impediment to the development of your own image-analysis systems is your own programming ability.

Some of the original material has not changed very much. Edge detection, thinning, thresholding, and morphology have not been hot areas of research, and the chapters in this edition are quite similar to those in the original. The software has been updated to use Intel's OpenCV system, which makes image IO and display much easier for programmers. It is even a simple matter to capture images from a webcam in real time and use them as input to the programs. Chapter 1 contains a discussion of the basics of OpenCV use, and all software in this book uses OpenCV as a basis.

Much of the mathematics in this book is still necessary for the detailed understanding of the algorithms described. Advanced methods in image processing and vision require the motivation and justification that only mathematics can provide. In some cases, I have only scratched the surface, and have left a more detailed study for those willing to follow the references given at the ends of chapters. I have tried to select references that provide a range of approaches, from detailed and complex mathematical analyses to clear and concise exposition. However, in some cases there are very few clear descriptions in the literature, and none that do not require at least a university-level math course. Here I have attempted to describe the situation in an intuitive manner, sacrificing rigor (which can be found almost anywhere else) for as

clear a description as possible. The software that accompanies the descriptions is certainly an alternative to the math, and gives a step-by-step description of the algorithms.

I have deleted some material completely from the first edition. There is no longer a chapter on wavelets, nor is there a chapter on genetic algorithms. On the other hand, there is a new chapter on classifiers, which I think was an obvious omission in the first edition. A key inclusion here is the chapter on the use of parallel programming for solving image-processing problems, including the use of graphics cards (GPUs) to accelerate calculations by factors up to 200. There's also a completely new chapter on content-based searches, which is the use of image information to retrieve other images. It's like saying, "Find me another image that looks like this." Content-based search will be an essential technology over the next two decades. It will enable the effective use of modern large-capacity disk drives; and with the proliferation of inexpensive high-resolution digital cameras, it makes sense that people will be searching through large numbers of big images (huge numbers of pixels) more and more often.

Most of the algorithms discussed in this edition can be found in source code form on the accompanying web page. The chapter on thresholding alone provides 17 programs, each implementing a different thresholding algorithm. Thinning programs, edge detection, and morphology are all now available on the Internet.

The chapter on image restoration is still one of the few sources of practical information on that subject. The symbol recognition chapter has been updated; however, as many methods are commercial, they cannot be described and software can't be provided due to patent and copyright concerns. Still, the basics are there, and have been connected with the material on classifiers.

The chapter on parallel programming for vision is, I think, a unique feature of this book. Again using downloadable tools, this chapter shows how to link all the computers on your network into a large image-processing cluster. Of course, it also shows how to use all the CPUs on your multi-core and, most importantly, gives an introductory and very practical look at how to program the GPU to do image processing and vision tasks, rather than just graphics.

Finally, I have provided a chapter giving a selection of methods for use in searching through images. These methods have code showing their implementation and, combined with other code in the book, will allow for many hours of experimenting with your own ideas and algorithms for organizing and searching image data sets.

Readers can download all the source code and sample images mentioned in this book from the book's web page — www.wiley.com/go/jrpkarker. You can also link to my own page, through which I will add new code, new images, and perhaps even new written material to supplement and update the printed matter. Comments and mistakes (how likely is that?) can be communicated

through that web page, and errata will be posted, as will reader contributions to the software collection and new ideas for ways to use the code methods for compiling on other systems and with other compilers.

I invite you to make suggestions through the website for subjects for new chapters that you would like to read. It is my intention to select a popular request and to post a new chapter on that subject on the site at a future date. A book, even one primarily released on paper, need not be a completely static thing!

Jim Parker
Cochrane, Alberta, Canada
October 2010

Practical Aspects of a Vision System—Image Display, Input/Output, and Library Calls

When experimenting with vision- and image-analysis systems or implementing one for a practical purpose, a basic software infrastructure is essential. Images consist of pixels, and in a typical image from a digital camera there will be 4–6 million pixels, each representing the color at a point in the image. This large amount of data is stored as a file in a format (such as GIF or JPEG) suitable for manipulation by commercial software packages, such as Photoshop and Paint. Developing new image-analysis software means first being able to read these files into an internal form that allows access to the pixel values. There is nothing exciting about code that does this, and it does not involve any actual image processing, but it is an essential first step. Similarly, image-analysis software will need to display images on the screen and save them in standard formats. It's probably useful to have a facility for image capture available, too. None of these operations modify an image but simply move it about in useful ways.

These bookkeeping tasks can require most of the code involved in an imaging program. The procedure for changing all red pixels to yellow, for example, can contain as few as 10 lines of code; yet, the program needed to read the image, display it, and output of the result may require an additional 2,000 lines of code, or even more.

Of course, this infrastructure code (which can be thought of as an *application programming interface*, or *API*) can be used for all applications; so, once it is developed, the API can be used without change until updates are required. Changes in the operating system, in underlying libraries, or in additional functionalities can require new versions of the API. If properly done, these

new versions will require little or no modification to the vision programs that depend on it. Such an API is the *OpenCV* system.

1.1 OpenCV

OpenCV was originally developed by Intel. At the time of this writing, version 2.0 is current and can be downloaded from <http://sourceforge.net/projects/opencvlibrary/>.

However, Version 2.0 is relatively new, yet it does not install and compile with all of the major systems and compilers. All the examples in this book use Version 1.1 from http://sourceforge.net/projects/opencvlibrary/files/opencv-win/1.1pre1/OpenCV_1.1pre1a.exe/download, and compile with the Microsoft Visual C++ 2008 Express Edition, which can be downloaded from www.microsoft.com/express/Downloads/#2008-Visual-CPP.

The *Algorithms for Image Processing and Computer Vision* website (www.wiley.com/go/jrparker) will maintain current links to new versions of these tools. The website shows how to install both the compiler and OpenCV. The advantage of using this combination of tools is that they are still pretty current, they work, and they are free.

1.2 The Basic OpenCV Code

OpenCV is a library of C functions that implement both infrastructure operations and image-processing and vision functions. Developers can, of course, add their own functions into the mix. Thus, any of the code described here can be invoked from a program that uses the OpenCV paradigm, meaning that the methods of this book are available in addition to those of OpenCV. One simply needs to know how to call the library, and what the basic data structures of open CV are.

OpenCV is a large and complex library. To assist everyone in starting to use it, the following is a basic program that can be modified to do almost anything that anyone would want:

```
// basic.c : A 'wrapper' for basic vision programs.
#include "stdafx.h"
#include "cv.h"
#include "highgui.h"
int main (int argc, char* argv[])
{
    IplImage *image = 0;
```

```

image = cvLoadImage("C:\\AIPCV\\image1.jpg", 1 );
if( image )
{
    cvNamedWindow( "Input Image", 1 );
    cvShowImage( "Input Image", image );
    printf( "Press a key to exit\n");
    cvWaitKey(0);
    cvDestroyWindow("String");
}
else
    fprintf( stderr, "Error reading image\n" );
return 0;
}

```

This is similar to many example programs on the Internet. It reads in an image (C:\\AIPCV\\image1.jpg is a string giving the path name of the image) and displays it in a window on the screen. When the user presses a key, the program terminates after destroying the display window.

Before anyone can modify this code in a knowledgeable way, the data structures and functions need to be explained.

1.2.1 The IplImage Data Structure

The `IplImage` structure is the in-memory data organization for an image. Images in `IplImage` form can be converted into arrays of pixels, but `IplImage` also contains a lot of structural information about the image data, which can have many forms. For example, an image read from a GIF file could be 256 grey levels with an 8-bit pixel size, or a JPEG file could be read into a 24-bit per pixel color image. Both files can be represented as an `IplImage`.

An `IplImage` is much like other internal image representations in its basic organization. The essential fields are as follows:

<code>width</code>	An integer holding the width of the image in pixels
<code>height</code>	An integer holding the height of the image in pixels
<code>imageData</code>	A pointer to an array of characters, each one an actual pixel or color value

If each pixel is one byte, this is really all we need. However, there are many data types for an image within OpenCV; they can be bytes, ints, floats, or doubles in type, for instance. They can be greys (1 byte) or 3-byte color (RGB), 4 bytes, and so on. Finally, some image formats may have the origin at the upper left (most do, in fact) and some use the lower left (only Microsoft).

Other useful fields to know about include the following:

<code>nChannels</code>	An integer specifying the number of colors per pixel (1–4).
<code>depth</code>	An integer specifying the number of bits per pixel.
<code>origin</code>	The origin of the coordinate system. An integer: 0=upper left, 1=lower left.
<code>widthStep</code>	An integer specifying, in bytes, the size of one row of the image.
<code>imageSize</code>	An integer specifying, in bytes, the size of the image (= <code>widthStep</code> * height).
<code>imageDataOrigin</code>	A pointer to the origin (root, base) of the image.
<code>roi</code>	A pointer to a structure that defines a region of interest within this image that is being processed.

When an image is created or read in from a file, an instance of an `IplImage` is created for it, and the appropriate fields are given values. Consider the following definition:

```
IplImage* img = 0;
```

As will be described later in more detail, an image can be read from a file by the following code:

```
img = cvLoadImage(filename);
```

where the variable `filename` is a string holding the name of the image file. If this succeeds, then

```
img->imageData
```

points to the block of memory where the pixels can be found. Figure 1.1 shows a JPEG image named `marchA062.jpg` that can be used as an example.

Reading this image creates a specific type of internal representation common to basic RGB images and will be the most likely variant of the `IplImage` structure to be encountered in real situations. This representation has each pixel represented as three bytes: one for red, one for green, and one for blue. They appear in the order b, g, r, starting at the first row of the image and stepping through columns, and then rows. Thus, the data pointed to by `img->imageData` is stored in the following order:

```
b0,0  g0,0  r0,0  b0,1  g0,1  r0,1  b0,2  g0,2  r0,2 ...
```

This means that the RGB values of the pixels in the first row (row 0) appear in reverse order (b, g, r) for all pixels in that row. Then comes the next row, starting over at column 0, and so on, until the final row.