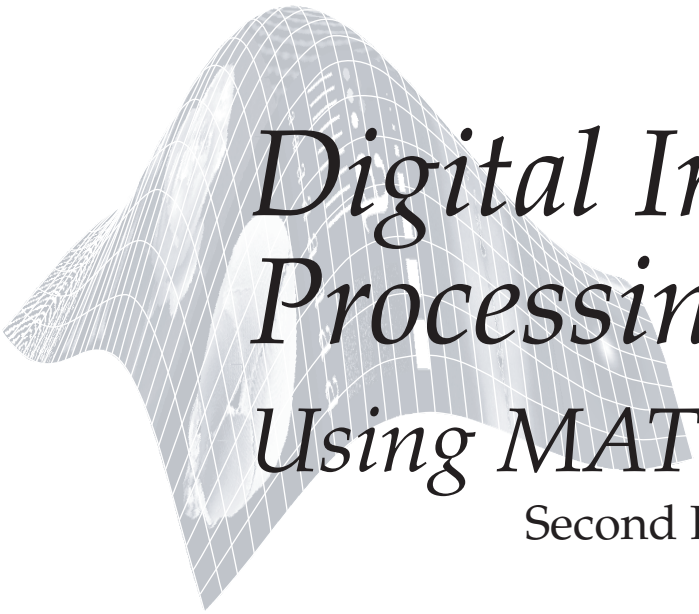


Digital Image Processing Using MATLAB®

Second Edition

Rafael C. Gonzalez
University of Tennessee

Richard E. Woods
MedData Interactive

Steven L. Eddins
The MathWorks, Inc.

Gatesmark Publishing®
A Division of Gatesmark,® LLC
www.gatesmark.com

Library of Congress Cataloging-in-Publication Data on File

Library of Congress Control Number: 2009902793

Gatesmark Publishing
A Division of Gatesmark, LLC
www.gatesmark.com

© 2009 by Gatesmark, LLC

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, without written permission from the publisher.

Gatesmark Publishing® is a registered trademark of Gatesmark, LLC, www.gatesmark.com.

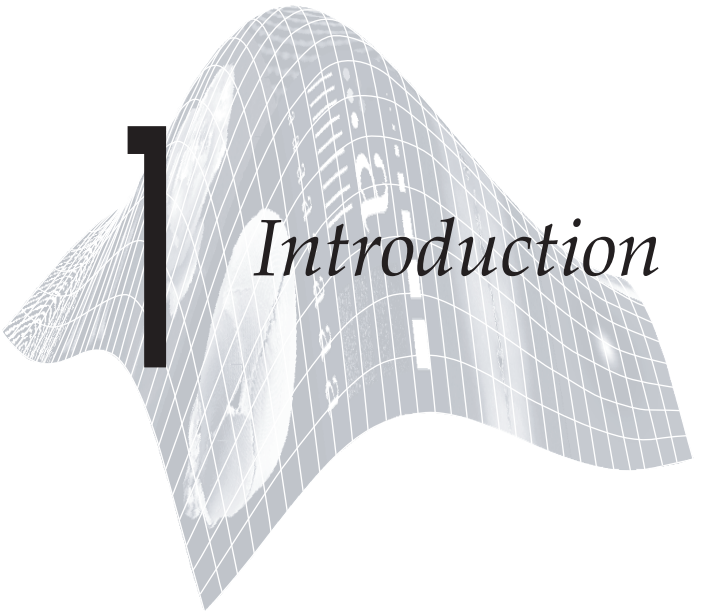
Gatesmark® is a registered trademark of Gatesmark, LLC, www.gatesmark.com.

MATLAB® is a registered trademark of The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098

The authors and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The authors and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 978-0-9820854-0-0



1 Introduction

Preview

Digital image processing is an area characterized by the need for extensive experimental work to establish the viability of proposed solutions to a given problem. In this chapter, we outline how a theoretical foundation and state-of-the-art software can be integrated into a prototyping environment whose objective is to provide a set of well-supported tools for the solution of a broad class of problems in digital image processing.

1.1 Background

An important characteristic underlying the design of image processing systems is the significant level of testing and experimentation that normally is required before arriving at an acceptable solution. This characteristic implies that the ability to formulate approaches and quickly prototype candidate solutions generally plays a major role in reducing the cost and time required to arrive at a viable system implementation.

Little has been written in the way of instructional material to bridge the gap between theory and application in a well-supported software environment for image processing. The main objective of this book is to integrate under one cover a broad base of theoretical concepts with the knowledge required to implement those concepts using state-of-the-art image processing software tools. The theoretical underpinnings of the material in the following chapters are based on the leading textbook in the field: *Digital Image Processing*, by Gonzalez and Woods.[†] The software code and supporting tools are based on the leading software in the field: *MATLAB*® and the *Image Processing Toolbox*™

[†] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed., Prentice Hall, Upper Saddle River, NJ, 2008 .

from The MathWorks, Inc. (see Section 1.3). The material in the book shares the same design, notation, and style of presentation as the Gonzalez-Woods text, thus simplifying cross-referencing between the two.

The book is self-contained. To master its contents, a reader should have introductory preparation in digital image processing, either by having taken a formal course of study on the subject at the senior or first-year graduate level, or by acquiring the necessary background in a program of self-study. Familiarity with MATLAB and rudimentary knowledge of computer programming are assumed also. Because MATLAB is a matrix-oriented language, basic knowledge of matrix analysis is helpful.

The book is based on principles. It is organized and presented in a textbook format, not as a manual. Thus, basic ideas of both theory and software are explained prior to the development of any new programming concepts. The material is illustrated and clarified further by numerous examples ranging from medicine and industrial inspection to remote sensing and astronomy. This approach allows orderly progression from simple concepts to sophisticated implementation of image processing algorithms. However, readers already familiar with MATLAB, the Image Processing Toolbox, and image processing fundamentals can proceed directly to specific applications of interest, in which case the functions in the book can be used as an extension of the family of toolbox functions. All new functions developed in the book are fully documented, and the code for each is included either in a chapter or in Appendix C.

We use the term *custom function* to denote a function developed in the book, as opposed to a "standard" MATLAB or Image Processing Toolbox function.

Over 120 *custom functions* are developed in the chapters that follow. These functions extend by nearly 45% the set of about 270 functions in the Image Processing Toolbox. In addition to addressing specific applications, the new functions are good examples of how to combine existing MATLAB and toolbox functions with new code to develop prototype solutions to a broad spectrum of problems in digital image processing. The toolbox functions, as well as the functions developed in the book, run under most operating systems. Consult the book web site (see Section 1.5) for a complete list.

1.2 What Is Digital Image Processing?

An image may be defined as a two-dimensional function, $f(x, y)$, where x and y are *spatial coordinates*, and the amplitude of f at any pair of coordinates (x, y) is called the *intensity* or *gray level* of the image at that point. When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a *digital image*. The field of digital image processing refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as *picture elements*, *image elements*, *pels*, and *pixels*. *Pixel* is the term used most widely to denote the elements of a digital image. We consider these definitions formally in Chapter 2.

Vision is the most advanced of our senses, so it is not surprising that images play the single most important role in human perception. However, unlike humans, who are limited to the visual band of the electromagnetic (EM)

spectrum, imaging machines cover almost the entire EM spectrum, ranging from gamma to radio waves. They can operate also on images generated by sources that humans do not customarily associate with images. These include ultrasound, electron microscopy, and computer-generated images. Thus, digital image processing encompasses a wide and varied field of applications.

There is no general agreement among authors regarding where image processing stops and other related areas, such as image analysis and computer vision, begin. Sometimes a distinction is made by defining image processing as a discipline in which both the input and output of a process are images. We believe this to be a limiting and somewhat artificial boundary. For example, under this definition, even the trivial task of computing the average intensity of an image would not be considered an image processing operation. On the other hand, there are fields, such as computer vision, whose ultimate goal is to use computers to emulate human vision, including learning and being able to make inferences and take actions based on visual inputs. This area itself is a branch of artificial intelligence (AI), whose objective is to emulate human intelligence. The field of AI is in its infancy in terms of practical developments, with progress having been much slower than originally anticipated. The area of *image analysis* (also called *image understanding*) is in between image processing and computer vision.

There are no clear-cut boundaries in the continuum from image processing at one end to computer vision at the other. However, a useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, and high-level processes. *Low-level* processes involve primitive operations, such as image preprocessing to reduce noise, contrast enhancement, and image sharpening. A low-level process is characterized by the fact that both its inputs and outputs typically are images. *Mid-level* processes on images involve tasks such as segmentation (partitioning an image into regions or objects), description of those objects to reduce them to a form suitable for computer processing, and classification (recognition) of individual objects. A mid-level process is characterized by the fact that its inputs generally are images, but its outputs are attributes extracted from those images (e.g., edges, contours, and the identity of individual objects). Finally, *high-level* processing involves "making sense" of an ensemble of recognized objects, as in image analysis, and, at the far end of the continuum, performing the cognitive functions normally associated with human vision.

Based on the preceding comments, we see that a logical place of overlap between image processing and image analysis is the area of recognition of individual regions or objects in an image. Thus, what we call in this book *digital image processing* encompasses processes whose inputs and outputs are images and, in addition, encompasses processes that extract attributes from images, up to and including the recognition of individual objects. As a simple illustration to clarify these concepts, consider the area of automated analysis of text. The processes of acquiring an image of a region containing the text, preprocessing that image, extracting (segmenting) the individual characters, describing the characters in a form suitable for computer processing, and recognizing those

individual characters, are in the scope of what we call digital image processing in this book. Making sense of the content of the page may be viewed as being in the domain of image analysis and even computer vision, depending on the level of complexity implied by the statement "making sense of." Digital image processing, as we have defined it, is used successfully in a broad range of areas of exceptional social and economic value.

1.3 Background on MATLAB and the Image Processing Toolbox

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include the following:

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including building graphical user interfaces

MATLAB is an interactive system whose basic data element is a matrix. This allows formulating solutions to many technical computing problems, especially those involving matrix representations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C.

The name MATLAB stands for *Matrix Laboratory*. MATLAB was written originally to provide easy access to matrix and linear algebra software that previously required writing FORTRAN programs to use. Today, MATLAB incorporates state of the art numerical computation software that is highly optimized for modern processors and memory architectures.

In university environments, MATLAB is the standard computational tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the computational tool of choice for research, development, and analysis. MATLAB is complemented by a family of application-specific solutions called *toolboxes*. The Image Processing Toolbox is a collection of MATLAB functions (called *M-functions* or *M-files*) that extend the capability of the MATLAB environment for the solution of digital image processing problems. Other toolboxes that sometimes are used to complement the Image Processing Toolbox are the Signal Processing, Neural Networks, Fuzzy Logic, and Wavelet Toolboxes.

As we discuss in more detail in Chapter 2, images may be treated as matrices, thus making MATLAB software a natural choice for image processing applications.

The *MATLAB & Simulink Student Version* is a product that includes a full-featured version of MATLAB, the Image Processing Toolbox, and several other useful toolboxes. The Student Version can be purchased at significant discounts at university bookstores and at the MathWorks web site (www.mathworks.com).

1.4 Areas of Image Processing Covered in the Book

Every chapter in the book contains the pertinent MATLAB and Image Processing Toolbox material needed to implement the image processing methods discussed. When a MATLAB or toolbox function does not exist to implement a specific method, a custom function is developed and documented. As noted earlier, a complete listing of every new function is available. The remaining twelve chapters cover material in the following areas.

Chapter 2: Fundamentals. This chapter covers the fundamentals of MATLAB notation, matrix indexing, and programming concepts. This material serves as foundation for the rest of the book.

Chapter 3: Intensity Transformations and Spatial Filtering. This chapter covers in detail how to use MATLAB and the Image Processing Toolbox to implement intensity transformation functions. Linear and nonlinear spatial filters are covered and illustrated in detail. We also develop a set of basic functions for fuzzy intensity transformations and spatial filtering.

Chapter 4: Processing in the Frequency Domain. The material in this chapter shows how to use toolbox functions for computing the forward and inverse 2-D fast Fourier transforms (FFTs), how to visualize the Fourier spectrum, and how to implement filtering in the frequency domain. Shown also is a method for generating frequency domain filters from specified spatial filters.

Chapter 5: Image Restoration. Traditional linear restoration methods, such as the Wiener filter, are covered in this chapter. Iterative, nonlinear methods, such as the Richardson-Lucy method and maximum-likelihood estimation for blind deconvolution, are discussed and illustrated. Image reconstruction from projections and how it is used in computed tomography are discussed also in this chapter.

Chapter 6: Geometric Transformations and Image Registration. This chapter discusses basic forms and implementation techniques for geometric image transformations, such as affine and projective transformations. Interpolation methods are presented also. Different image registration techniques are discussed, and several examples of transformation, registration, and visualization methods are given.

Chapter 7: Color Image Processing. This chapter deals with pseudocolor and full-color image processing. Color models applicable to digital image processing are discussed, and Image Processing Toolbox functionality in color processing is extended with additional color models. The chapter also covers applications of color to edge detection and region segmentation.

Chapter 8: Wavelets. The Image Processing Toolbox does not have wavelet transform functions. Although the MathWorks offers a Wavelet Toolbox, we develop in this chapter an independent set of wavelet transform functions that allow implementation all the wavelet-transform concepts discussed in Chapter 7 of *Digital Image Processing* by Gonzalez and Woods.

Chapter 9: Image Compression. The toolbox does not have any data compression functions. In this chapter, we develop a set of functions that can be used for this purpose.

Chapter 10: Morphological Image Processing. The broad spectrum of functions available in toolbox for morphological image processing are explained and illustrated in this chapter using both binary and gray-scale images.

Chapter 11: Image Segmentation. The set of toolbox functions available for image segmentation are explained and illustrated in this chapter. Functions for Hough transform processing are discussed, and custom region growing and thresholding functions are developed.

Chapter 12: Representation and Description. Several new functions for object representation and description, including chain-code and polygonal representations, are developed in this chapter. New functions are included also for object description, including Fourier descriptors, texture, and moment invariants. These functions complement an extensive set of region property functions available in the Image Processing Toolbox.

Chapter 13: Object Recognition. One of the important features of this chapter is the efficient implementation of functions for computing the Euclidean and Mahalanobis distances. These functions play a central role in pattern matching. The chapter also contains a comprehensive discussion on how to manipulate strings of symbols in MATLAB. String manipulation and matching are important in structural pattern recognition.

In addition to the preceding material, the book contains three appendices.

Appendix A: This appendix summarizes Image Processing Toolbox and custom image-processing functions developed in the book. Relevant MATLAB functions also are included. This is a useful reference that provides a global overview of all functions in the toolbox and the book.

Appendix B: Implementation of graphical user interfaces (GUIs) in MATLAB are discussed in this appendix. GUIs complement the material in the book because they simplify and make more intuitive the control of interactive functions.

Appendix C: The code for many custom functions is included in the body of the text at the time the functions are developed. Some function listings are deferred to this appendix when their inclusion in the main text would break the flow of explanations.

1.5 The Book Web Site

An important feature of this book is the support contained in the book web site. The site address is

www.ImageProcessingPlace.com

This site provides support to the book in the following areas:

- Availability of M-files, including executable versions of all M-files in the book
- Tutorials
- Projects
- Teaching materials
- Links to databases, including all images in the book
- Book updates
- Background publications

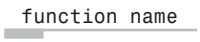
The same site also supports the Gonzalez-Woods book and thus offers complementary support on instructional and research topics.

1.6 Notation

Equations in the book are typeset using familiar italic and Greek symbols, as in $f(x, y) = A\sin(ux + vy)$ and $\phi(u, v) = \tan^{-1}\left[I(u, v)/R(u, v)\right]$. All MATLAB function names and symbols are typeset in monospace font, as in `fft2(f)`, `logical(A)`, and `roipoly(f, c, r)`.

The first occurrence of a MATLAB or Image Processing Toolbox function is highlighted by use of the following icon on the page margin:

Similarly, the first occurrence of a new (custom) function developed in the book is highlighted by use of the following icon on the page margin:



The symbol ▬▬ is used as a visual cue to denote the end of a function listing.

When referring to keyboard keys, we use bold letters, such as **Return** and **Tab**. We also use bold letters when referring to items on a computer screen or menu, such as **File** and **Edit**.

1.7 The MATLAB Desktop

The *MATLAB Desktop* is the main working environment. It is a set of graphics tools for tasks such as running MATLAB commands, viewing output, editing and managing files and variables, and viewing session histories. Figure 1.1 shows the MATLAB Desktop in the default configuration. The Desktop com-

ponents shown are the Command Window, the Workspace Browser, the Current Directory Browser, and the Command History Window. Figure 1.1 also shows a Figure Window, which is used to display images and graphics.

The *Command Window* is where the user types MATLAB commands at the prompt (>>). For example, a user can call a MATLAB function, or assign a value to a variable. The set of variables created in a session is called the *Workspace*, and their values and properties can be viewed in the *Workspace Browser*.

Directories are called *folders* in Windows.

The top-most rectangular window shows the user's *Current Directory*, which typically contains the path to the files on which a user is working at a given time. The current directory can be changed using the arrow or browse button ("...") to the right of the *Current Directory Field*. Files in the Current Directory can be viewed and manipulated using the *Current Directory Browser*.

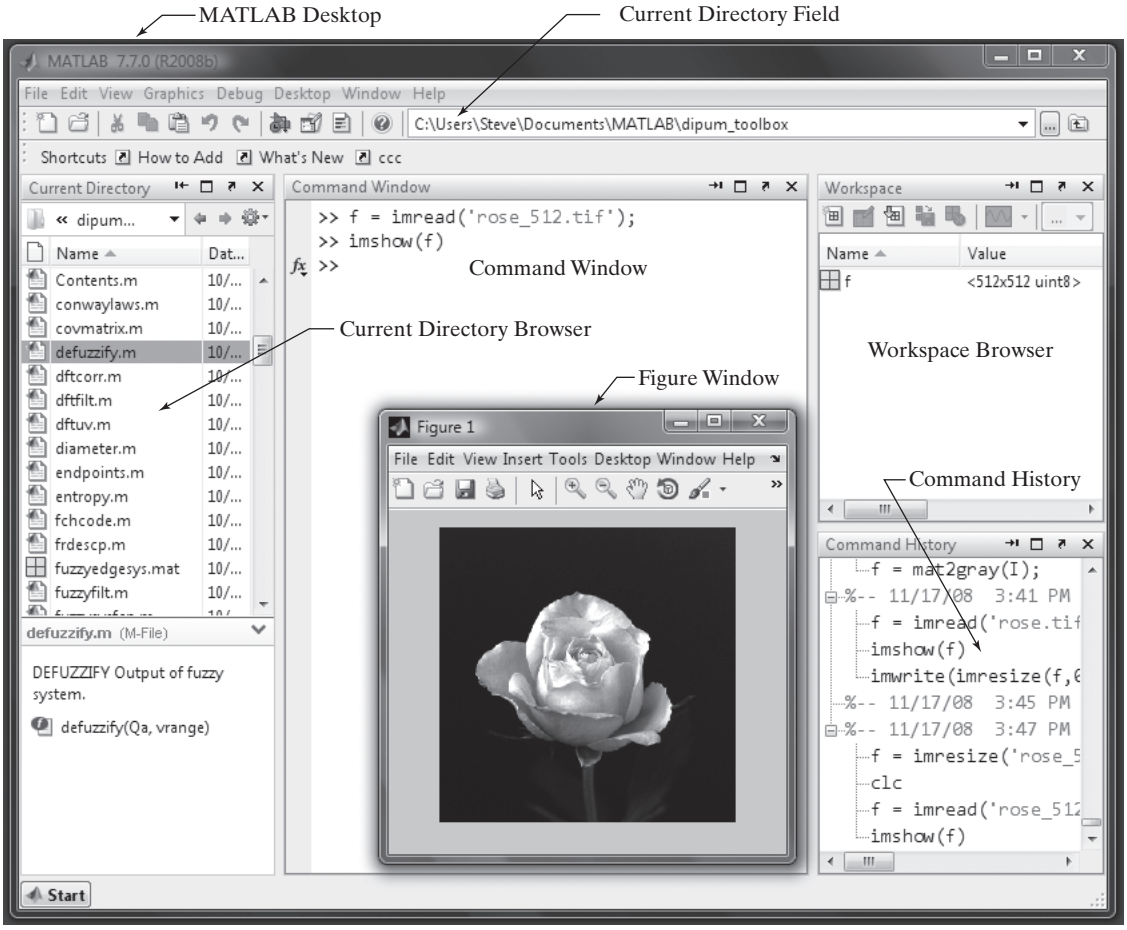


FIGURE 1.1 The MATLAB Desktop with its typical components.

The *Command History Window* displays a log of MATLAB statements executed in the Command Window. The log includes both current and previous sessions. From the Command History Window a user can right-click on previous statements to copy them, re-execute them, or save them to a file. These features are useful for experimenting with various commands in a work session, or for reproducing work performed in previous sessions.

The MATLAB Desktop may be configured to show one, several, or all these tools, and favorite Desktop layouts can be saved for future use. Table 1.1 summarizes all the available Desktop tools.

MATLAB uses a *search path* to find M-files and other MATLAB-related files, which are organized in directories in the computer file system. Any file run in MATLAB must reside in the Current Directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path. The easiest way to see which directories are on the search path, or to add or modify a search path, is to select **Set Path** from the **File** menu on the desktop, and then use the **Set Path** dialog box. It is good practice to add commonly used directories to the search path to avoid repeatedly having to browse to the location of these directories.

Typing `clear` at the prompt removes all variables from the workspace. This frees up system memory. Similarly, typing `clc` clears the contents of the command window. See the help page for other uses and syntax forms.

TABLE 1.1 MATLAB desktop tools.

Tool	Description
Array Editor	View and edit array contents.
Command History Window	View a log of statements entered in the Command Window; search for previously executed statements, copy them, and re-execute them.
Command Window	Run MATLAB statements.
Current Directory Browser	View and manipulate files in the current directory.
Current Directory Field	Shows the path leading to the current directory.
Editor/Debugger	Create, edit, debug, and analyze M-files.
Figure Windows	Display, modify, annotate, and print MATLAB graphics.
File Comparisons	View differences between two files.
Help Browser	View and search product documentation.
Profiler	Measure execution time of MATLAB functions and lines; count how many times code lines are executed.
Start Button	Run product tools and access product documentation and demos.
Web Browser	View HTML and related files produced by MATLAB or other sources.
Workspace Browser	View and modify contents of the workspace.

1.7.1 Using the MATLAB Editor/Debugger

The *MATLAB Editor/Debugger* (or just the *Editor*) is one of the most important and versatile of the Desktop tools. Its primary purpose is to create and edit MATLAB function and script files. These files are called *M-files* because their filenames use the extension `.m`, as in `pixeldup.m`. The Editor highlights different MATLAB code elements in color; also, it analyzes code to offer suggestions for improvements. The Editor is the tool of choice for working with M-files. With the Editor, a user can set debugging breakpoints, inspect variables during code execution, and step through code lines. Finally, the Editor can publish MATLAB M-files and generate output to formats such as HTML, LaTeX, Word, and PowerPoint.

To open the editor, type `edit` at the prompt in the Command Window. Similarly, typing `edit filename` at the prompt opens the M-file `filename.m` in an editor window, ready for editing. The file must be in the current directory, or in a directory in the search path.

1.7.2 Getting Help

The principal way to get help is to use the MATLAB *Help Browser*, opened as a separate window either by clicking on the question mark symbol (**?**) on the desktop toolbar, or by typing `doc` (one word) at the prompt in the Command Window. The Help Browser consists of two panes, the *help navigator pane*, used to find information, and the *display pane*, used to view the information. Self-explanatory tabs on the navigator pane are used to perform a search. For example, help on a specific function is obtained by selecting the **Search** tab and then typing the function name in the **Search for** field. It is good practice to open the Help Browser at the beginning of a MATLAB session to have help readily available during code development and other MATLAB tasks.

Another way to obtain help for a specific function is by typing `doc` followed by the function name at the command prompt. For example, typing `doc file_name` displays the reference page for the function called `file_name` in the display pane of the Help Browser. This command opens the browser if it is not open already. The `doc` function works also for user-written M-files that contain help text. See Section 2.10.1 for an explanation of M-file help text.

When we introduce MATLAB and Image Processing Toolbox functions in the following chapters, we often give only representative syntax forms and descriptions. This is necessary either because of space limitations or to avoid deviating from a particular discussion more than is absolutely necessary. In these cases we simply introduce the syntax required to execute the function in the form required at that point in the discussion. By being comfortable with MATLAB documentation tools, you can then explore a function of interest in more detail with little effort.

Finally, the MathWorks' web site mentioned in Section 1.3 contains a large database of help material, contributed functions, and other resources that

should be utilized when the local documentation contains insufficient information about a desired topic. Consult the book web site (see Section 1.5) for additional MATLAB and M-function resources.

1.7.3 Saving and Retrieving Work Session Data

There are several ways to save or load an entire work session (the contents of the Workspace Browser) or selected workspace variables in MATLAB. The simplest is as follows: To save the entire workspace, right-click on any blank space in the Workspace Browser window and select **Save Workspace As** from the menu that appears. This opens a directory window that allows naming the file and selecting any folder in the system in which to save it. Then click **Save**. To save a selected variable from the Workspace, select the variable with a left click and right-click on the highlighted area. Then select **Save Selection As** from the menu that appears. This opens a window from which a folder can be selected to save the variable. To select multiple variables, use shift-click or control-click in the familiar manner, and then use the procedure just described for a single variable. All files are saved in a binary format with the extension `.mat`. These saved files commonly are referred to as *MAT-files*, as indicated earlier. For example, a session named, say, mywork_2009_02_10, would appear as the MAT-file mywork_2009_02_10.mat when saved. Similarly, a saved image called `final_image` (which is a single variable in the workspace) will appear when saved as final_image.mat.

To load saved workspaces and/or variables, left-click on the folder icon on the toolbar of the Workspace Browser window. This causes a window to open from which a folder containing the MAT-files of interest can be selected. Double-clicking on a selected MAT-file or selecting **Open** causes the contents of the file to be restored in the Workspace Browser window.

It is possible to achieve the same results described in the preceding paragraphs by typing `save` and `load` at the prompt, with the appropriate names and path information. This approach is not as convenient, but it is used when formats other than those available in the menu method are required. Functions `save` and `load` are useful also for writing M-files that save and load workspace variables. As an exercise, you are encouraged to use the Help Browser to learn more about these two functions.

1.8 How References Are Organized in the Book

All references in the book are listed in the Bibliography by author and date, as in Soille [2003]. Most of the background references for the theoretical content of the book are from Gonzalez and Woods [2008]. In cases where this is not true, the appropriate new references are identified at the point in the discussion where they are needed. References that are applicable to all chapters, such as MATLAB manuals and other general MATLAB references, are so identified in the Bibliography.

Summary

In addition to a brief introduction to notation and basic MATLAB tools, the material in this chapter emphasizes the importance of a comprehensive prototyping environment in the solution of digital image processing problems. In the following chapter we begin to lay the foundation needed to understand Image Processing Toolbox functions and introduce a set of fundamental programming concepts that are used throughout the book. The material in Chapters 3 through 13 spans a wide cross section of topics that are in the mainstream of digital image processing applications. However, although the topics covered are varied, the discussion in those chapters follows the same basic theme of demonstrating how combining MATLAB and toolbox functions with new code can be used to solve a broad spectrum of image-processing problems.