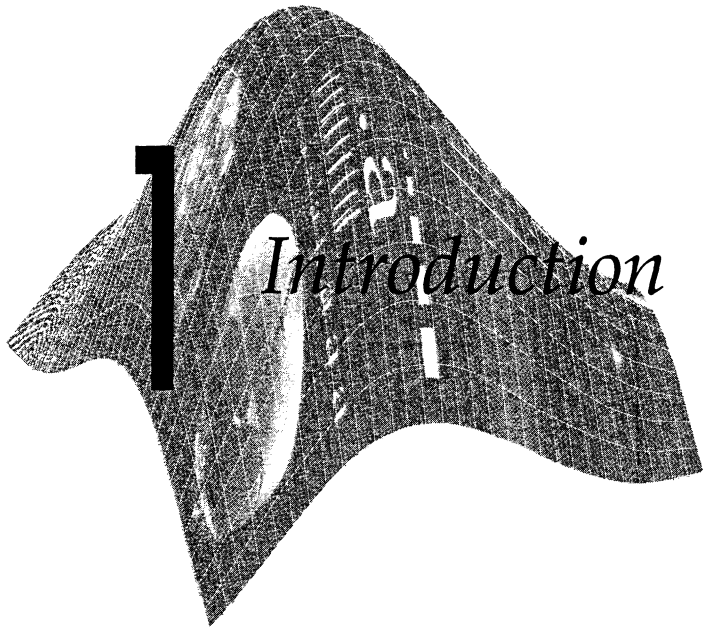

1 Introduction

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 base 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. 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 mainly from the leading textbook in the field: *Digital Image Processing*, by Gonzalez and Woods, published by Prentice Hall. The software code and supporting tools are based on the leading software package in the field: The *MATLAB Image Processing Toolbox*,†

†In the following discussion and in subsequent chapters we sometimes refer to *Digital Image Processing* by Gonzalez and Woods as "the Gonzalez-Woods book," and to the Image Processing Toolbox as "IPT" or simply as the "toolbox."

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

The book is self-contained. To master its contents, the 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. It is assumed also that the reader has some familiarity with MATLAB, as well as rudimentary knowledge of the basics of computer programming, such as that acquired in a sophomore- or junior-level course on programming in a technically oriented language. Because MATLAB is an array-oriented language, basic knowledge of matrix analysis also 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, IPT, 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 IPT 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.

Over 60 new functions are developed in the chapters that follow. These functions complement and extend by 35% the set of about 175 functions in IPT. In addition to addressing specific applications, the new functions are clear examples of how to combine existing MATLAB and IPT functions with new code to develop prototypic 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 most widely used 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 are not accustomed to associating 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, start. 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 earliest stages of infancy in terms of development, 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, one 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 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, higher-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 the area 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." 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 graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. 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 noninteractive language such as C or Fortran.

The name MATLAB stands for *matrix laboratory*. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (Linear System Package) and EISPACK (Eigen System Package) projects. Today, MATLAB engines incorporate the LAPACK (Linear Algebra Package) and BLAS (Basic Linear Algebra Subprograms) libraries, constituting the state of the art in software for matrix computation.

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 IPT are the Signal Processing, Neural Network, Fuzzy Logic, and Wavelet Toolboxes.

The MATLAB Student Version includes a full-featured version of MATLAB. The Student Version can be purchased at significant discounts at university bookstores and at the MathWorks' Web site (www.mathworks.com). Student versions of add-on products, including the Image Processing Toolbox, also are available.

1.4 Areas of Image Processing Covered in the Book

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

Chapter 2: Fundamentals. This chapter covers the fundamentals of MATLAB notation, 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 IPT to implement intensity transformation functions. Linear and nonlinear spatial filters are covered and illustrated in detail.

Chapter 4: Processing in the Frequency Domain. The material in this chapter shows how to use IPT functions for computing the forward and inverse 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. Geometric corrections and image registration also are covered.

Chapter 6: Color Image Processing. This chapter deals with pseudocolor and full-color image processing. Color models applicable to digital image processing are discussed, and IPT functionality in color processing is extended via implementation of additional color models. The chapter also covers applications of color to edge detection and region segmentation.

Chapter 7: Wavelets. In its current form, IPT does not have any wavelet transforms. A set of wavelet-related functions compatible with the Wavelet Toolbox is developed in this chapter that will allow the reader to implement all the wavelet-transform concepts discussed in the Gonzalez-Woods book.

Chapter 8: 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 9: Morphological Image Processing. The broad spectrum of functions available in IPT for morphological image processing are explained and illustrated in this chapter using both binary and gray-scale images.

Chapter 10: Image Segmentation. The set of IPT functions available for image segmentation are explained and illustrated in this chapter. New functions for Hough transform processing and region growing also are developed.

Chapter 11: 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 IPT.

Chapter 12: 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: Contains a summary of all IPT and new image-processing functions developed in the book. Relevant MATLAB function also are included. This is a useful reference that provides a global overview of all functions in the toolbox and the book.

Appendix B: Contains a discussion on how to implement graphical user interfaces (GUIs) in MATLAB. GUIs are a useful complement to the material in the book because they simplify and make more intuitive the control of interactive functions.

Appendix C: New function listings are included in the body of a chapter when a new concept is explained. Otherwise the listing is included in Appendix C. This is true also for listings of functions that are lengthy. Deferring the listing of some functions to this appendix was done primarily to avoid breaking the flow of explanations in text material.

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.prenhall.com/gonzalezwoodseddins

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

- Downloadable M-files, including all M-files in the book
- Tutorials

- Projects
- Teaching materials
- Links to databases, including all images in the book
- Book updates
- Background publications

The site is integrated with the Web site of the Gonzalez-Woods book:

www.prenhall.com/gonzalezwoods

which offers additional 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 $\phi(u, v) = \tan^{-1}[I(u, v)/R(u, v)]$. 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 IPT function is highlighted by use of the following icon on the page margin:

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

`function name`

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 Working Environment

In this section we give a brief overview of some important operational aspects of using MATLAB.

1.7.1 The MATLAB Desktop

The MATLAB *desktop* is the main MATLAB application window. As Fig. 1.1 shows, the desktop contains five subwindows: the Command Window, the Workspace Browser, the Current Directory Window, the Command History Window, and one or more Figure Windows, which are shown only when the user displays a graphic.

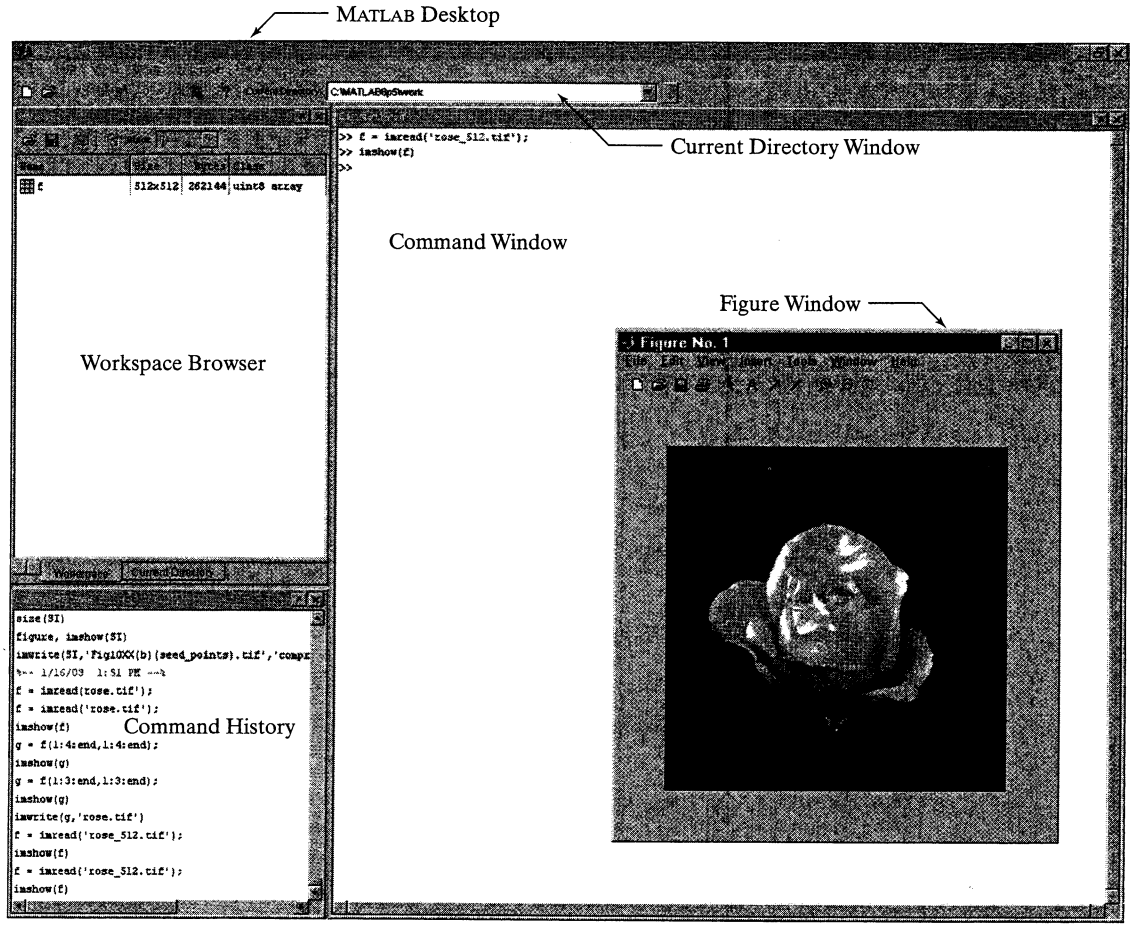

FIGURE 1.1 The MATLAB desktop and its principal components.

The *Command Window* is where the user types MATLAB commands and expressions at the prompt (>>) and where the outputs of those commands are displayed. MATLAB defines the *workspace* as the set of variables that the user creates in a work session. The *Workspace Browser* shows these variables and some information about them. Double-clicking on a variable in the Workspace Browser launches the *Array Editor*, which can be used to obtain information and in some instances edit certain properties of the variable.

The Current Directory tab above the Workspace tab shows the contents of the *current directory*, whose *path* is shown in the *Current Directory Window*. For example, in the Windows operating system the path might be as follows: C:\MATLAB\Work, indicating that directory "Work" is a subdirectory of the main directory "MATLAB," which is installed in drive C. Clicking on the arrow in the Current Directory Window shows a list of recently used paths. Clicking on the button to the right of the window allows the user to change the current directory.

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 any commonly used directories to the search path to avoid repeatedly having the change the current directory.

The *Command History Window* contains a record of the commands a user has entered in the Command Window, including both current and previous MATLAB sessions. Previously entered MATLAB commands can be selected and re-executed from the Command History Window by right-clicking on a command or sequence of commands. This action launches a menu from which to select various options in addition to executing the commands. This is a useful feature when experimenting with various commands in a work session.

1.7.2 Using the MATLAB Editor to Create M-Files

The MATLAB *editor* is both a text editor specialized for creating M-files and a graphical MATLAB debugger. The editor can appear in a window by itself, or it can be a subwindow in the desktop. M-files are denoted by the extension `.m`, as in `pixeldup.m`. The MATLAB editor window has numerous pull-down menus for tasks such as saving, viewing, and debugging files. Because it performs some simple checks and also uses color to differentiate between various elements of code, this text editor is recommended as the tool of choice for writing and editing M-functions. 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. As noted earlier, the file must be in the current directory, or in a directory in the search path.

1.7.3 Getting Help

The principal way to get help online[†] 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 `helpbrowser` at the prompt in the Command Window. The Help Browser is a Web browser integrated into the MATLAB desktop that displays Hypertext Markup Language (HTML) documents. 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, selecting **Function Name** as the **Search Type**, and then typing in the function name in the **Search for** field. It is good practice to open the Help Browser

[†]Use of the term *online* in this book refers to information, such as help files, available in a local computer system, not on the Internet.

at the beginning of a MATLAB session to have help readily available during code development or other MATLAB task.

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 format` displays documentation for the function called `format` in the display pane of the Help Browser. This command opens the browser if it is not already open.

M-functions have two types of information that can be displayed by the user. The first is called the *H1 line*, which contains the function name and a one-line description. The second is a block of explanation called the *Help text block* (these are discussed in detail in Section 2.10.1). Typing `help` at the prompt followed by a function name displays both the H1 line and the Help text for that function in the Command Window. Occasionally, this information can be more up to date than the information in the Help browser because it is extracted directly from the documentation of the M-function in question. Typing `lookfor` followed by a keyword displays all the H1 lines that contain that keyword. This function is useful when looking for a particular topic without knowing the names of applicable functions. For example, typing `lookfor edge` at the prompt displays all the H1 lines containing that keyword. Because the H1 line contains the function name, it then becomes possible to look at specific functions using the other help methods. Typing `lookfor edge -all` at the prompt displays the H1 line of all functions that contain the word `edge` in either the H1 line or the Help text block. Words that contain the characters `edge` also are detected. For example, the H1 line of a function containing the word `polyedge` in the H1 line or Help text would also be displayed.

It is common MATLAB terminology to use the term *help page* when referring to the information about an M-function displayed by any of the preceding approaches, excluding `lookfor`. It is highly recommended that the reader become familiar with all these methods for obtaining information because in the following chapters we often give only representative syntax forms for MATLAB and IPT functions. 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. By being comfortable with online search methods, the reader 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 online documentation contains insufficient information about a desired topic.

1.7.4 Saving and Retrieving a Work Session

There are several ways to save and 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, simply 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 simply click **Save**. To save a selected variable from the Workspace, select the variable with a left click and then right-click on the highlighted area. Then select **Save Selection As** from the menu that appears. This again 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 double-precision, binary format with the extension `.mat`. These saved files commonly are referred to as *MAT-files*. For example, a session named, say, mywork_2003_02_10, would appear as the MAT-file mywork_2003_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 file 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. As an exercise, the reader is 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 [2002]. 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 IPT functions and introduce a set of fundamental programming concepts that are used throughout the book. The material in Chapters 3 through 12 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 IPT functions with new code can be used to solve a broad spectrum of image-processing problems.