

# **IMAQ™**

---

## **IMAQ Vision Concepts Manual**



October 2000 Edition  
Part Number 322916A-01

## **Worldwide Technical Support and Product Information**

ni.com

## **National Instruments Corporate Headquarters**

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

## **Worldwide Offices**

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,  
Canada (Calgary) 403 274 9391, Canada (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,  
China 0755 3904939, Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24,  
Germany 089 741 31 30, Greece 30 1 42 96 427, Hong Kong 2645 3186, India 91805275406,  
Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456, Mexico (D.F.) 5 280 7625,  
Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, New Zealand 09 914 0488, Norway 32 27 73 00,  
Poland 0 22 528 94 06, Portugal 351 1 726 9011, Singapore 2265886, Spain 91 640 0085,  
Sweden 08 587 895 00, Switzerland 056 200 51 51, Taiwan 02 2528 7227, United Kingdom 01635 523545

For further support information, see the *Technical Support Resources* appendix. To comment on the documentation, send e-mail to [techpubs@ni.com](mailto:techpubs@ni.com)

© Copyright 2000 National Instruments Corporation. All rights reserved.

# Important Information

---

## Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

## Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

## Trademarks

IMAQ™, National Instruments™, and ni.com™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

- (1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.
- (2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATION DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HERAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

# Contents

---

## About This Manual

Conventions .....	xv
Related Documentation.....	xvi

## PART I Vision Basics

## Chapter 1 Digital Images

Definition of a Digital Image.....	1-1
Properties of a Digitized Image .....	1-2
Image Resolution.....	1-2
Image Definition.....	1-2
Number of Planes .....	1-3
Image Types.....	1-3
Grayscale Images.....	1-4
Color Images .....	1-5
Complex Images.....	1-5
Image Files.....	1-5
Internal Representation of an IMAQ Vision Image.....	1-6
Image Borders.....	1-8
Image Masks .....	1-10
When to Use .....	1-10
Concepts .....	1-10
Color Spaces .....	1-13
When to Use .....	1-13
Concepts .....	1-14
The RGB Color Space.....	1-15
HSL Color Space .....	1-17
CIE-Lab Color Space .....	1-17
CMY Color Space.....	1-18
YIQ Color Space.....	1-18
In-Depth Discussion .....	1-18
RGB To Grayscale .....	1-18
RGB and HSL .....	1-19
RGB and CIE L*a*b* .....	1-20
RGB and CMY.....	1-21
RGB and YIQ.....	1-21

## **Chapter 2**

### **Display**

Image Display.....	2-1
Image Display Concepts .....	2-1
When to Use.....	2-1
In-Depth Discussion.....	2-2
Display Modes .....	2-2
Mapping Methods for 16-Bit Image Display .....	2-3
Palettes.....	2-4
When to Use.....	2-4
Concepts.....	2-5
In-Depth Discussion.....	2-5
Gray Palette .....	2-5
Temperature Palette .....	2-6
Rainbow Palette .....	2-7
Gradient Palette .....	2-7
Binary Palette .....	2-8
Regions of Interest.....	2-9
When to Use.....	2-9
ROI Concepts .....	2-10
Nondestructive Overlay .....	2-11
When to Use.....	2-11
Nondestructive Overlay Concepts .....	2-11

## **Chapter 3**

### **System Setup and Calibration**

Setting Up Your Imaging System.....	3-1
Acquiring Quality Images.....	3-3
Resolution.....	3-3
Contrast.....	3-5
Depth of Field.....	3-5
Perspective.....	3-5
Distortion .....	3-7
Spatial Calibration.....	3-7
When to Use.....	3-7
Concepts.....	3-8
Calibration Process .....	3-8
Coordinate System.....	3-9
Calibration Algorithms .....	3-11
Calibration Quality Information .....	3-12

Image Correction.....	3-14
Scaling Mode .....	3-14
Correction Region .....	3-15
Simple Calibration .....	3-16
Redefining a Coordinate System.....	3-17

## PART II

## Image Processing and Analysis

### **Chapter 4**

### **Image Analysis**

Histogram.....	4-1
When to Use .....	4-1
Histogram Concepts .....	4-2
Linear Histogram.....	4-3
Cumulative Histogram.....	4-3
Interpretation .....	4-4
Histogram Scale.....	4-4
Histogram of Color Images .....	4-5
Line Profile .....	4-6
When to Use .....	4-6
Intensity Measurements .....	4-6
When to Use .....	4-6
Concepts .....	4-7
Densitometry.....	4-7

### **Chapter 5**

### **Image Processing**

Lookup Tables .....	5-1
When to Use .....	5-1
LUT Transformation Concepts.....	5-1
Example .....	5-2
Predefined Lookup Tables .....	5-3
Logarithmic and Inverse Gamma Correction.....	5-4
Exponential and Gamma Correction.....	5-6
Equalize.....	5-8
Convolution Kernels .....	5-10
Concepts .....	5-10
Spatial Filtering.....	5-13
When to Use .....	5-13

Spatial Filtering Concepts.....	5-14
Spatial Filter Classification Summary .....	5-14
Linear Filters.....	5-15
Nonlinear Filters .....	5-27
In-Depth Discussion.....	5-32
Linear Filters.....	5-32
Nonlinear Prewitt Filter .....	5-33
Nonlinear Sobel Filter .....	5-33
Nonlinear Gradient Filter.....	5-34
Roberts Filter .....	5-34
Differentiation Filter.....	5-34
Sigma Filter .....	5-35
Lowpass Filter .....	5-35
Median Filter .....	5-35
Nth Order Filter .....	5-35
Grayscale Morphology .....	5-36
When to Use.....	5-36
Grayscale Morphology Concepts.....	5-36
Erosion Function.....	5-37
Dilation Function .....	5-37
Erosion and Dilation Examples .....	5-37
Opening Function .....	5-38
Closing Function.....	5-39
Opening and Closing Examples .....	5-39
Proper-Opening Function .....	5-40
Proper-Closing Function.....	5-40
Auto-Median Function .....	5-40
In-Depth Discussion.....	5-41
Erosion Concept and Mathematics .....	5-41
Dilation Concept and Mathematics .....	5-41
Proper-Opening Concept and Mathematics.....	5-42
Proper-Closing Concept and Mathematics .....	5-42
Auto-Median Concept and Mathematics .....	5-43

## Chapter 6

### Operators

Introduction .....	6-1
When to Use.....	6-1
Operator Concepts.....	6-1
Arithmetic Operators .....	6-2
Logic and Comparison Operators.....	6-2
Example 1 .....	6-5
Example 2 .....	6-6

## Chapter 7

### Frequency Domain Analysis

Introduction.....	7-1
When to Use .....	7-3
Fast Fourier Transform Concepts.....	7-3
FFT Representation.....	7-3
Lowpass FFT Filters .....	7-6
Highpass FFT Filters.....	7-9
Mask FFT Filters.....	7-11
In-Depth Discussion .....	7-12
Fourier Transform .....	7-12
FFT Display .....	7-13

## PART III

### Blob Analysis

Introduction.....	III-1
When to Use .....	III-2
Blob Analysis Concepts .....	III-2

## Chapter 8

### Thresholding

Introduction.....	8-1
When to Use .....	8-1
Thresholding Concepts.....	8-2
Intensity Threshold .....	8-2
Thresholding Example .....	8-2
Automatic Threshold.....	8-3
In-Depth Discussion .....	8-6
Auto-Thresholding Techniques.....	8-6
Clustering .....	8-7
Entropy .....	8-7
InterVarianc.....	8-8
Metric .....	8-8
Moments.....	8-9
Color Thresholding.....	8-9
When to Use .....	8-9

# Chapter 9

## Binary Morphology

Introduction .....	9-1
Structuring Elements .....	9-1
When to Use.....	9-1
Structuring Elements Concepts .....	9-2
Structuring Element Size .....	9-2
Structuring Element Values.....	9-3
Pixel Frame Shape .....	9-4
Connectivity .....	9-7
When to Use.....	9-7
Connectivity Concepts .....	9-7
In-Depth Discussion.....	9-9
Connectivity-4 .....	9-9
Connectivity-8 .....	9-9
Primary Morphology Operations.....	9-10
When to Use.....	9-10
Primary Morphology Concepts .....	9-10
Erosion and Dilation Functions .....	9-11
Opening and Closing Functions .....	9-13
Inner Gradient Function.....	9-14
Outer Gradient Function.....	9-14
Hit-Miss Function .....	9-15
Thinning Function .....	9-17
Thickening Function.....	9-19
Proper-Opening Function .....	9-21
Proper-Closing Function.....	9-21
Auto-Median Function .....	9-22
Advanced Morphology Operations .....	9-22
When to Use.....	9-22
Advanced Morphology Transforms Concepts .....	9-23
Border Function .....	9-23
Hole Filling Function.....	9-23
Labeling Function.....	9-23
Lowpass and Highpass Filters .....	9-24
Separation Function .....	9-25
Skeleton Functions .....	9-26
Segmentation Function .....	9-28
Distance Function .....	9-29
Danielsson Function .....	9-30
Circle Function .....	9-30
Convex Function.....	9-31

## Chapter 10

### Particle Measurements

Digital Particles.....	10-1
When to Use .....	10-1
Digital Particle Concepts.....	10-1
Areas .....	10-1
Lengths.....	10-2
In-Depth Discussion .....	10-9
Definitions of Primary Measurements .....	10-9
Derived Measurements .....	10-10

## PART IV

### Machine Vision

## Chapter 11

### Edge Detection

Introduction.....	11-1
When to Use .....	11-1
Gauging.....	11-2
Detection .....	11-3
Alignment.....	11-4
Edge Detection Concepts .....	11-4
Definition of an Edge.....	11-4
Characteristics of an Edge.....	11-5
Edge Detection Methods .....	11-7
Extending Edge Detection to Two-Dimensional Search Regions ....	11-11

## Chapter 12

### Pattern Matching

Introduction.....	12-1
When to Use .....	12-1
Pattern Matching Concepts.....	12-3
What to Expect from a Pattern Matching Tool .....	12-3
Pattern Matching Techniques .....	12-4
Traditional Pattern Matching .....	12-5
New Pattern Matching Techniques .....	12-6
In-Depth Discussion .....	12-8
Cross Correlation .....	12-8

Shape Matching .....	12-10
When to Use.....	12-10
Shape Matching Concepts.....	12-10

## **Chapter 13**

### **Dimensional Measurements**

Introduction .....	13-1
When to Use.....	13-1
Dimensional Measurements Concepts .....	13-2
Locating the Part in the Image.....	13-2
Locating Features .....	13-2
Making Measurements .....	13-3
Qualifying Measurements.....	13-3
Coordinate System .....	13-3
When to Use .....	13-4
Concepts .....	13-4
In-Depth Discussion .....	13-5
Finding Features or Measurement Points.....	13-10
Edge-Based Features .....	13-10
Line and Circular Features .....	13-11
Shape-Based Features.....	13-13
Making Measurements on the Image .....	13-13
Distance Measurements.....	13-13
Analytic Geometry .....	13-14

## **Chapter 14**

### **Color Inspection**

The Color Spectrum .....	14-1
Color Space Used to Generate the Spectrum.....	14-1
Generating the Color Spectrum.....	14-3
Color Matching.....	14-6
When to Use.....	14-6
Color Identification.....	14-6
Color Inspection .....	14-7
Color Matching Concepts .....	14-9
Learning Color Distribution .....	14-9
Comparing Color Distributions .....	14-9
Color Location.....	14-10
When to Use.....	14-10
Inspection.....	14-11
Identification.....	14-12

What to Expect from a Color Location Tool.....	14-14
Pattern Orientation and Multiple Instances.....	14-14
Ambient Lighting Conditions .....	14-15
Blur and Noise Conditions .....	14-15
Color Location Concept .....	14-15
Color Pattern Matching.....	14-18
When to Use .....	14-18
What to Expect from a Color Pattern Matching Tool .....	14-21
Pattern Orientation and Multiple Instances.....	14-21
Ambient Lighting Conditions .....	14-22
Blur and Noise Conditions .....	14-22
Color Pattern Matching Concepts .....	14-22
Color Matching and Color Location .....	14-23
Grayscale Pattern Matching .....	14-23
Combining Color Location and Grayscale Pattern Matching .....	14-24

## Chapter 15 Instrument Readers

Introduction.....	15-1
When to Use .....	15-1
Meter Functions.....	15-1
Meter Algorithm Limits.....	15-2
LCD Functions .....	15-2
LCD Algorithm Limits.....	15-3
Barcode.....	15-3
Barcode Algorithm Limits .....	15-4

## Appendix A Kernels

## Appendix B Technical Support Resources

## Glossary

## Index

# About This Manual

---

The *IMAQ Vision Concepts Manual* helps people with little or no imaging experience learn the basic concepts of machine vision and image processing. This manual also contains in-depth discussions on machine vision and image processing functions for advanced users.

## Conventions

---

The following conventions appear in this manual:

- |   |   |
|---|---|
| <>  | Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, DBIO<3..0>.  |
|  | This icon denotes a tip, which alerts you to advisory information.  |
|  | This icon denotes a note, which alerts you to important information.  |
| <b>bold</b>   | Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.  |
| <i>italic</i>   | Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.  |
| monospace   | Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts. |

## Related Documentation

---

The following documents contain information that you might find helpful as you read this manual:

- *IMAQ Vision for LabVIEW User Manual*
- *IMAQ Vision for Measurement Studio User Manual*
- IMAQ Vision for LabVIEW online VI reference
- IMAQ Vision for Measurement Studio LabWindows/CVI online function reference
- IMAQ Vision for Measurement Studio ActiveX Controls online function reference

# Part I

---

## Vision Basics

This section describes conceptual information about digital images, display, and system calibration.

Part I, *Vision Basics*, contains the following chapters:

Chapter 1, *Digital Images*, contains information about the properties of digital images, image types and file formats, the internal representation of images in IMAQ Vision, image borders, image masks, and color spaces.

Chapter 2, *Display*, contains information about image display, palettes, regions of interest, and nondestructive overlays.

Chapter 3, *System Setup and Calibration*, describes how to setup an imaging system and calibrate the imaging setup so that you can convert pixel coordinates to real-world coordinates.

# Digital Images

This chapter contains information about the properties of digital images, image types and file formats, the internal representation of images in IMAQ Vision, image borders, image masks, and color spaces.

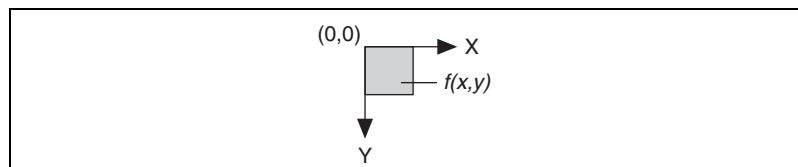
## Definition of a Digital Image

An image is a two-dimensional array of values representing light intensity. For the purposes of image processing, the term *image* refers to a digital image. An image is a function of the light intensity

$$f(x, y)$$

where  $f$  is the brightness of the point  $(x, y)$ , and  $x$  and  $y$  represent the spatial coordinates of a picture element (abbreviated *pixel*).

By convention, the spatial reference of the pixel with the coordinates  $(0, 0)$  is located at the top, left corner of the image. Notice in Figure 1-1 that the value of  $x$  increases moving from left to right, and the value of  $y$  increases from top to bottom.



**Figure 1-1.** Spatial Reference of the  $(0, 0)$  Pixel

In *digital image processing*, an imaging sensor converts an image into a discrete number of pixels. The imaging sensor assigns to each pixel a numeric location and a *gray level* or color value that specifies the brightness or color of the pixel.

# Properties of a Digitized Image

---

A digitized image has three basic properties: resolution, definition, and number of planes.

## Image Resolution

The *spatial resolution* of an image is its number of rows and columns of pixels. An image composed of  $m$  columns and  $n$  rows has a resolution of  $m \times n$ . This image has  $m$  pixels along its horizontal axis and  $n$  pixels along its vertical axis.

## Image Definition

The definition of an image indicates the number of shades that you can see in the image. The *bit depth* of an image is the number of bits used to encode the value of a pixel. For a given bit depth of  $n$ , the image has an image definition of  $2^n$ , meaning a pixel can have  $2^n$  different values. For example, if  $n$  equals 8 bits, a pixel can take 256 different values ranging from 0 to 255. If  $n$  equals 16 bits, a pixel can take 65,536 different values ranging from 0 to 65,535 or from -32,768 to 32,767. Currently, IMAQ Vision only supports a range of -32,768 to 32,767 for 16-bit images.

IMAQ Vision can process images with 8-bit, 10-bit, 12-bit, 14-bit, 16-bit, floating point, or color encoding. The manner in which you encode your image depends on the nature of the image, the type of image processing you need to use, and the type of analysis you need to perform. For example, 8-bit encoding is sufficient if you need to obtain the shape information of objects in an image. However, if you need to precisely measure the light intensity of an image or region in an image, you must use 16-bit or floating-point encoding.

Use color encoded images when your machine vision or image processing application depends on the color content of the objects you are inspecting or analyzing.

IMAQ Vision does not directly support other types of image encoding, particularly images encoded as 1-bit, 2-bit, or 4-bit images. In these cases, IMAQ Vision automatically transforms the image into an 8-bit image (the minimum bit depth for IMAQ Vision) when opening the image file.

## Number of Planes

The number of planes in an image corresponds to the number of arrays of pixels that compose the image. A grayscale or pseudo-color image is composed of one plane, while a true-color image is composed of three planes—one for the red component, blue component, and green component.

In true-color images, the color component intensities of a pixel are coded into three different values. The color image is the combination of three arrays of pixels corresponding to the red, green, and blue components in an RGB image. HSL images are defined by their hue, saturation, and luminance values.

## Image Types

---

The IMAQ Vision libraries can manipulate three types of images: grayscale, color, and complex images. Although IMAQ Vision supports all three image types, certain operations on specific image types are not possible (for example, applying the logic operator AND to a complex image).

Table 1-1 shows how many bytes per pixel grayscale, color, and complex images use. For an identical spatial resolution, a color image occupies four times the memory space of an 8-bit grayscale image, and a complex image occupies eight times the memory of an 8-bit grayscale image.

**Table 1-1.** Bytes Per Pixel

Image Type	Number of Bytes Per Pixel Data
<b>8-bit (Unsigned) Integer grayscale</b>  (1 byte or 8-bit)	 8-bit for the grayscale intensity
<b>16-bit (Signed) Integer grayscale</b>  (2 bytes or 16-bit)	 16-bit for the grayscale intensity

**Table 1-1.** Bytes Per Pixel (Continued)

<b>32-bit Floating-Point grayscale</b> (4 bytes or 32-bit)	 32-bit floating for the grayscale intensity			
<b>RGB Color</b> (4 bytes or 32-bit)	 8-bit for the alpha value (not used)      8-bit for the Red intensity      8-bit for the Green intensity      8-bit for the Blue intensity			
<b>HSL Color</b> (4 bytes or 32-bit)	 8 bits not used      8-bit for the Hue      8-bit for the Saturation      8-bit for the Luminance			
<b>Complex</b> (8 bytes or 64-bit)	 32-bit floating for the Real part      32-bit floating for the Imaginary part			

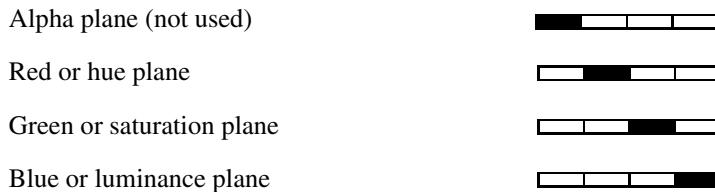
## Grayscale Images

A *grayscale image* is composed of a single plane of pixels. Each pixel is encoded using a single number. This number can be:

- An 8-bit unsigned integer representing grayscale values between 0 and 255
- A 16-bit signed integer representing grayscale values between -32768 and +32767
- A single precision floating point number (encoded using four bytes) representing grayscale values ranging from  $-\infty$  to  $\infty$

## Color Images

A *color image* is encoded in memory as either an RGB or HSL image. Color image pixels are a composite of four values. RGB images store color information using 8 bits each for the red, green, and blue planes. HSL images store color information using 8 bits each for hue, saturation, and luminance. In all of the color models, an additional 8-bit value goes unused. This representation is known as  $4 \times 8$ -bit or 32-bit encoding.



## Complex Images

A *complex image* contains the frequency information of a grayscale image. Create a complex image by applying a Fast Fourier transform (FFT) to a grayscale image. Once you transform a grayscale image into a complex image, you can perform frequency domain operations on the image.

Each pixel in a complex image is encoded as two single-precision floating-point values, which represent the real and imaginary components of the complex pixel. You can extract the following four components from a complex image: the real part, imaginary part, magnitude, and phase.

## Image Files

---

An *image file* is composed of a header followed by pixel values. Depending on the file format, the header contains image information about the horizontal and vertical resolution, pixel definition, and the original palette. Image files may also store information about calibration, pattern matching templates, and overlays. The following are some common image file formats:

- Bitmap ([BMP](#))
- Tagged image file format ([TIFF](#))
- Portable network graphics ([PNG](#))—offers the capability of storing image information about spatial calibration, pattern matching templates, and overlays

- Joint Photographic Experts Group format ([JPEG](#))
- National Instruments internal image file format ([AIPD](#))—used for saving floating-point, complex, and HSL images

Standard 8-bit grayscale formats are BMP, TIFF, PNG, JPEG, and AIPD.

Standard 16-bit grayscale formats are PNG and AIPD.

Standard color file formats for RGB images are BMP, TIFF, PNG, JPEG, and AIPD.

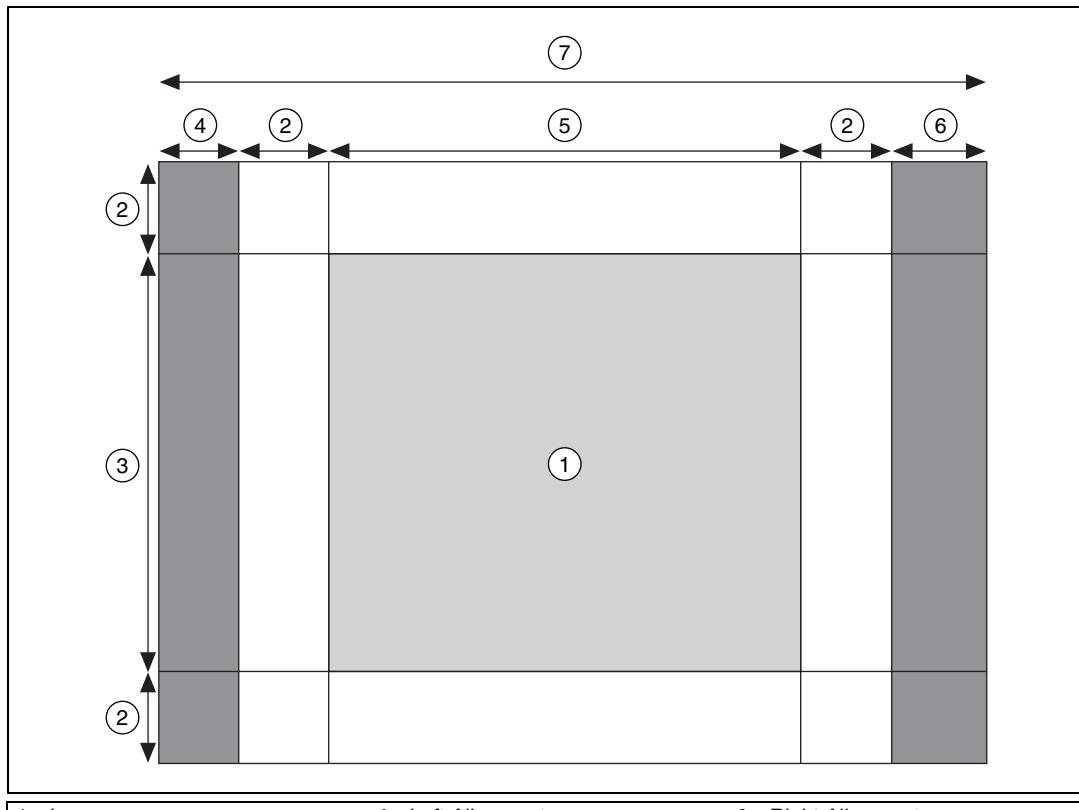
Standard complex image file formats are PNG and AIPD.

## Internal Representation of an IMAQ Vision Image

---

Figure 1-2 illustrates how an IMAQ Vision image is represented in your system memory. In addition to the pixels of the image, the stored image includes additional rows and columns of pixels called the image border and the left and right alignments. Specific processing functions involving pixel neighborhood operations use image borders. The alignment regions ensure that the first pixel of the image is 8-byte aligned in memory. The size of the alignment blocks depend on the image width and border size. Aligning the image increases processing speed by as much as 30%.

The line width is the total number of pixels in a horizontal line of an image. The line width is the sum of the horizontal resolution, the image borders, and the left and right alignments. The horizontal resolution and line width may be the same length if the horizontal resolution is a multiple of 8 bytes and the border size is 0.



1 Image  
2 Image Border  
3 Vertical Resolution

4 Left Alignment  
5 Horizontal Resolution

6 Right Alignment  
7 Line Width

**Figure 1-2.** Internal Image Representation

# Image Borders

---

Many image processing functions process a pixel by using the values of its neighbors. A *neighbor* is a pixel whose value affects the value of a nearby pixel when an image is processed. Pixels along the edge of an image do not have neighbors on all four sides. If you need to use a function that processes pixels based on the value of their neighboring pixels, specify an *image border* that surrounds the image to account for these outlying pixels. You define the image border by specifying a border size and the values of the border pixels.

The size of the border should accommodate the largest pixel neighborhood required by the function you are using. The size of the neighborhood is specified by the size of a 2D array. For example, if a function uses the eight adjoining neighbors of a pixel for processing, the size of the neighborhood is  $3 \times 3$ , indicating an array with three columns and three rows. Set the border size to be greater than or equal to half the number of rows or columns of the 2D array rounded down to the nearest integer value. For example, if a function uses a  $3 \times 3$  neighborhood, the image should have a border size of at least 1; if a function uses a  $5 \times 5$  neighborhood, the image should have a border size of at least 2. In IMAQ Vision, an image is created with a default border size of 3. This supports any function using up to a  $7 \times 7$  neighborhood.

IMAQ Vision provides three ways to specify the pixel values of the image border. Figure 1-3 illustrates these options. Figure 1-3a shows the pixel values of an image. You can set all the image border pixels to zero (this is the default case), as shown in Figure 1-3b. You can copy the values of the pixels along the edge of the image into the border pixels, as shown in Figure 1-3c, or you can mirror the pixels values along the edge of the image into the border pixels, as shown in Figure 1-3d.

**a.**

10	9	15	11	12	20	16	12	11	8
11	13	11	12	9	16	17	11	13	14
12	8	12	14	12	13	12	14	11	13
10	9	13	31	30	32	33	12	13	11
15	11	10	30	42	45	31	15	12	10
13	12	14	29	40	41	33	13	12	13
14	15	12	33	34	36	32	12	14	11
10	12	13	14	12	16	12	15	10	9
10	8	11	13	15	17	13	14	12	10
9	10	12	11	8	15	14	12	11	7

**b.**

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	10	9	15	11	12	20	16	12
0	0	11	13	11	12	9	16	17	11
0	0	12	8	12	14	12	13	12	14
0	0	10	9	13	31	30	32	33	12
0	0	15	11	10	30	42	45	31	15
0	0	13	12	14	29	40	41	33	13
0	0	14	15	12	33	34	36	32	12
0	0	10	12	13	14	12	16	12	15
0	0	10	8	11	13	15	17	13	14
0	0	9	10	12	11	8	15	14	12
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

**c.**

10	10	10	9	15	11	12	20	16	22
10	10	10	9	15	11	12	20	16	22
10	10	10	9	15	11	12	20	16	22
11	11	11	13	11	12	9	16	17	11
12	12	12	8	12	14	12	13	12	14
10	10	10	9	13	31	30	32	33	12
15	15	15	11	10	30	42	45	31	15
13	13	13	12	14	29	40	41	33	13
14	14	14	15	12	33	34	36	32	12
10	10	10	12	13	14	12	16	12	15
10	10	10	8	11	13	15	17	13	14
9	9	9	10	12	11	8	15	14	12
9	9	9	10	12	11	8	15	14	12
9	9	9	10	12	11	8	15	14	12

**d.**

13	11	11	13	11	12	9	16	17	11
10	9	10	9	15	11	12	20	16	12
10	9	10	9	15	11	12	20	16	12
13	11	11	13	11	12	9	16	17	11
8	12	12	8	12	14	12	13	12	14
9	10	10	9	13	31	30	32	33	12
11	15	15	11	10	30	42	45	31	15
12	13	13	12	14	29	40	41	33	13
15	14	14	15	12	33	34	36	32	12
12	10	10	12	13	14	12	16	12	15
8	10	10	8	11	13	15	17	13	14
10	9	9	10	12	11	8	15	14	12
10	9	9	10	12	11	8	15	14	12
8	10	10	8	11	13	15	17	13	14

**Figure 1-3.** Setting the Pixel Values of an Image Border

The method you use to fill the border pixels depends on the processing function you require for your application. Review how the function works before choosing a border-filling method, since your choice can drastically affect the processing results. For example, if you are using a function that detects edges in an image based on the difference between a pixel and its neighbors, do not set the border pixel values to zero. As shown in Figure 1-3b, an image border containing zero values introduces significant differences between the pixel values in the border and the image pixels along the border, which causes the function to detect erroneous edges along the border of the image. If you need to use an edge detection function, copy or mirror the pixel values along the border into the border region to obtain more accurate results.

In IMAQ Vision, most image processing functions that use neighbors automatically set pixel values in the image border using neighborhoods. The grayscale filtering operations low pass, Nth order, and edge detection use the mirroring method to set pixels in the image border. The binary morphology, grayscale morphology, and segmentation functions copy the pixel values along the border into the border region. The correlate, circles, reject border, remove particles, skeleton, and label functions set the pixel values in the border to zero.



**Note** The border of an image is taken into account only for processing. The border is never displayed or stored to a file.

## Image Masks

---

An *image mask* isolates parts of an image for processing. If a function has an image mask parameter, the function process or analysis depends on both the source image and the image mask.

An image mask is an 8-bit binary image that is the same size as or smaller than the inspection image. Pixels in the image mask determine whether corresponding pixels in the inspection image are processed. If a pixel in the image mask has a non-zero value, the corresponding pixel in the inspection image is processed. If a pixel in the image mask has a value of 0, the corresponding pixel in the inspection image is not processed.

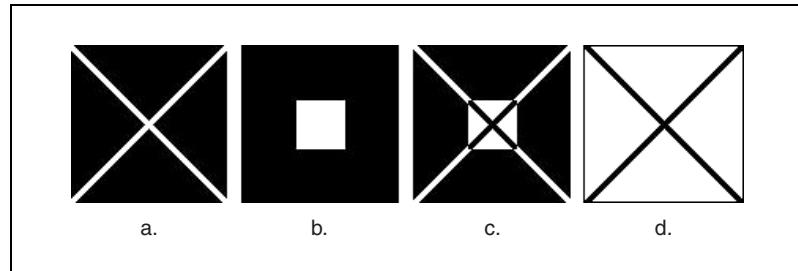
### When to Use

Use image masks when you want to focus your processing or inspection on particular regions in the image.

### Concepts

Pixels in the source image are processed if corresponding pixels in the image mask have values other than zero. Figure 1-4 shows how a mask affects the output of the function that inverts the pixel values in an image. Figure 1-4a shows the inspection image. Figure 1-4b shows the image mask. Figure 1-4c shows the inverse of the inspection image using the

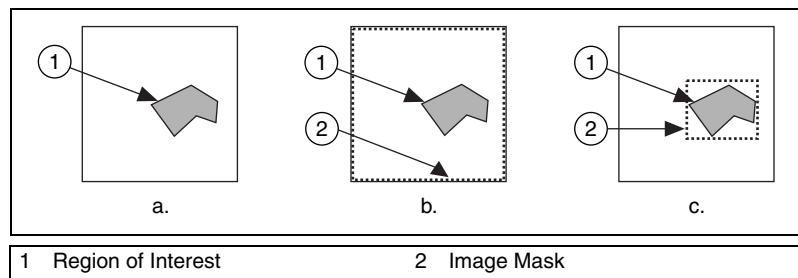
image mask. Figure 1-4d shows the inverse of the inspection image without the image mask.



**Figure 1-4.** The Effect of an Image Mask

You can limit the area in which your function applies an image mask to the bounding rectangle of the region you want to process. This technique saves memory by limiting the image mask to only the part of the image containing significant information. To keep track of the location of this *region of interest* (ROI) in regard to the original image, IMAQ Vision sets an *offset*. An offset defines the coordinate position in the original image where you want to place the origin of the image mask.

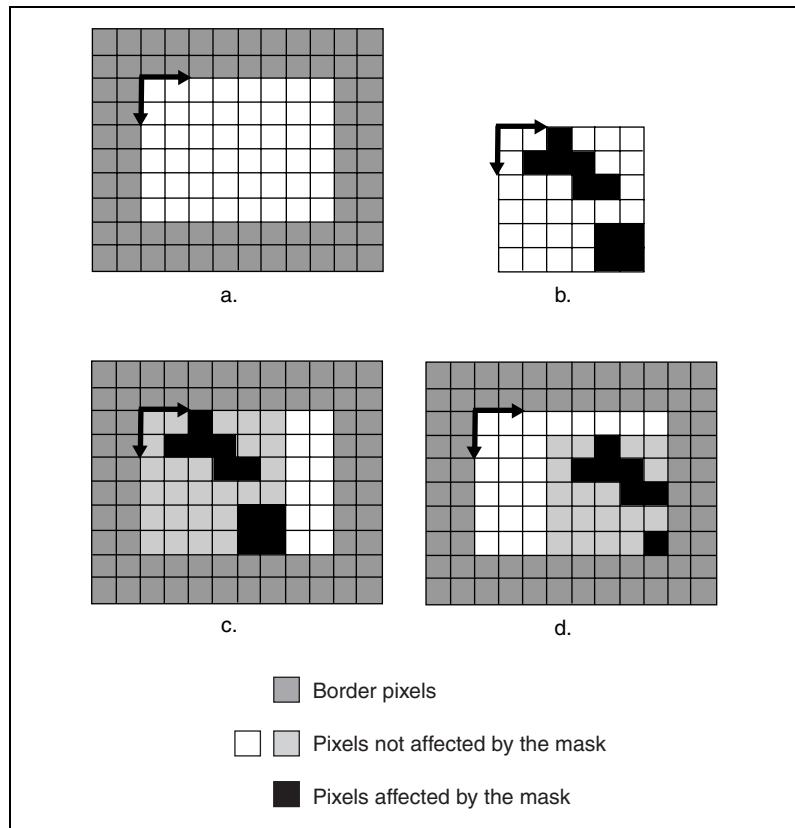
Figure 1-5 illustrates the different methods of applying image masks. Figure 1-5a shows the ROI in which you want to apply an image mask. Figure 1-5b shows an image mask with the same size as the inspection image. In this case, the offset is set to (0,0). A mask image can also be the size of the bounding box of the ROI, as shown in Figure 1-5c, where the offset specifies the location of the mask image in the reference image. You can define this offset to apply the mask image to different regions in the inspection image.



**Figure 1-5.** Using an Offset to Limit an Image Mask

Figure 1-6 illustrates the use of a mask with two different offsets.

Figure 1-6a shows the inspection image, and Figure 1-6b shows the image mask. Figure 1-6c and Figure 1-6d show the results of a function using the image mask given the offsets of [0, 0] and [3, 1], respectively.



**Figure 1-6.** Effect of Applying a Mask with Different Offsets

For more information about ROIs, see the *Regions of Interest* section of Chapter 2, *Display*.

# Color Spaces

---

*Color spaces* allow you to represent a color. A color space is a subspace within a three-dimensional coordinate system where each color is represented by a point. You can use color spaces to facilitate the description of colors between persons, machines, or software programs.

Various industries and applications use a number of different color spaces. Humans perceive color according to parameters such as brightness, hue, and intensity, while computers perceive color as a combination of red, green, and blue. The printing industry uses cyan, magenta, and yellow to specify color. The following is a list of common color spaces:

- RGB—Based on red, green, and blue. Used by computers to display images.
- HSL—Based on hue, saturation, and luminance. Used in image processing applications.
- CIE—Based on brightness, hue, and colorfulness. Defined by the Commission Internationale de l'Eclairage (International Commission on Illumination) as the different sensations of color that the human brain perceives.
- CMY—Based on cyan, magenta, and yellow. Used by the printing industry.
- YIQ—Separates the luminance information (Y) from the color information (I and Q). Used for TV broadcasting.

## When to Use

You must define a color space every time you process color images. With IMAQ Vision, you specify the color space associated with an image when you create the image. IMAQ Vision supports the RGB and HSL color spaces.

If you expect the lighting conditions to vary considerably during your color machine vision application, use the HSL color space. The HSL color space provides more accurate color information than the RGB space when running color processing functions, such as color matching, color location, and color pattern matching. IMAQ Vision's advanced algorithms for color processing—which perform under various lighting and noise conditions—process images in the HSL color space.

If you do not expect the lighting conditions to vary considerably during your application, and you can easily define the colors you are looking for using red, green, and blue, use the RGB space. Also, use the RGB space if you only want to display color images, but not process them, in your application. The RGB space reproduces an image as you would expect to see it. IMAQ Vision always displays color images in the RGB space. If you create an image in the HSL space, IMAQ Vision automatically converts the image to the RGB space before displaying it.

For more information about using color images, see Chapter 14, [Color Inspection](#).

## Concepts

Because color is the brain's reaction to a specific visual stimulus, color is best described by the different sensations of color that the human brain perceives. The color-sensitive cells in the eye's retina sample color using three bands that correspond to red, green, and blue light. The signals from these cells travel to the brain where they combine to produce different sensations of colors. The Commission Internationale de l'Eclairage has defined the following sensations:

- Brightness—The sensation of an area exhibiting more or less light
- Hue—The sensation of an area appearing similar to a combination of red, green, and blue
- Colorfulness—The sensation of an area appearing to exhibit more or less of its hue
- Lightness—The sensation of an area's brightness relative to a reference white in the scene
- Chroma—The colorfulness of an area with respect to a reference white in the scene
- Saturation—The colorfulness of an area relative to its brightness

The tri-chromatic theory describes how three separate lights—red, green, and blue—can be combined to match any visible color. This theory is based on the three color sensors that the eye uses. Printing and photography use the tri-chromatic theory as the basis for combining three different colored dyes to reproduce colors in a scene. Computer color spaces also use three parameters to define a color.

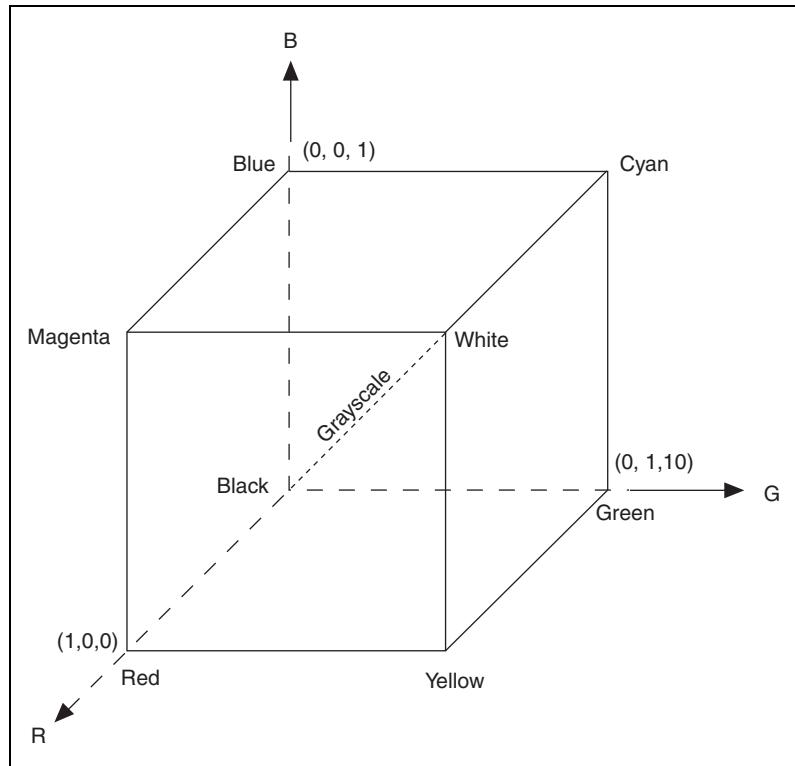
Most color spaces are geared toward displaying images via hardware, such as color monitors and printers, or toward applications that manipulate color information, such as computer graphics and image processing. Color CRT monitors, the majority of color-video cameras, and most computer graphics

systems use the RGB color space. The HSL space, combined with RGB and YIQ, is frequently used in applications that manipulate color, such as image processing. The color picture publishing industry uses the Cyan, Magenta, and Yellow (CMY) color space, also known as CMYK. The YIQ space is the standard for color TV broadcast.

## The RGB Color Space

The RGB color space is the most commonly used color space. The human eye receives color information in separate red, green, and blue components through cones—the color receptors present in the human eye. These three colors are known as additive primary colors. In an additive color system, the human brain processes the three primary light sources and combines them to compose a single color “image.” The three primary color components can combine to reproduce almost all other possible colors.

You can visualize the RGB space as a 3-dimensional cube with red, green, and blue at the corners of each axis, as shown in Figure 1-7. Black is at the origin, while white is at the opposite corner of the cube. Each side of the cube has a value between 0 and 1. Along each axis of the RGB cube, the colors range from no contribution of that component to a fully saturated color. Any point, or color, within the cube is specified by three numbers: an R, G, B triple. The diagonal line of the cube from black (0, 0, 0) to white (1, 1, 1) represents all the grayscale values or where all of the red, green, and blue components are equal. Different computer hardware and software combinations use different ranges for the colors. Common combinations are 0–255 and 0–65,535 for each component. To map color values within these ranges to values in the RGB cube, divide the color values by the maximum value that the range can take.



**Figure 1-7.** RGB Cube

The RGB color space lies within the perceptual space of humans. In other words, the RGB cube represents fewer colors than we can see.

The RGB space simplifies the design of computer monitors, but it is not ideal for all applications. In the RGB color space, the red, green, and blue color components are all necessary to describe a color. Therefore, RGB is not as intuitive as other color spaces. The HSL color space describes color using only the hue component, which makes HSL the best choice for many image processing applications, such as color matching.

## HSL Color Space

The HSL color space was developed to put color in terms that are easy for humans to quantify. Hue, saturation, and brightness are characteristics that distinguish one color from another in the HSL space. *Hue* corresponds to the dominant wavelength of the color. The hue component is a color, such as orange, green, or violet. You can visualize the range of hues as a rainbow. *Saturation* refers to the amount of white added to the hue and represents the relative purity of a color. A color without any white is fully saturated. The degree of saturation is inversely proportional to the amount of white light added. Colors such as pink (red and white) and lavender (purple and white) are less saturated than red and purple. *Brightness* embodies the chromatic notion of luminance, or the amplitude or power of light. *Chromacity* is the combination of hue and saturation, and the relationship between chromacity and brightness characterizes a color. Systems that manipulate hue use the HSL color space, which also can be written as HSI (Hue, Saturation, and Intensity) or HSV (Hue, Saturation, and Value).

The coordinate system for the HSL color space is cylindrical. Colors are defined inside a hexcone, as shown in Figure 14-1 of Chapter 14, *Color Inspection*. The hue value runs from 0 to 360°. The saturation S ranges from 0 to 1, where 1 represents the purest color (no white). Luminance also ranges from 0 to 1, where 0 is black and 1 is white.

Overall, two principal factors—the de-coupling of the intensity component from the color information and the close relationship between chromacity and human perception of color—make the HSL space ideal for developing machine vision applications.

## CIE-Lab Color Space

The CIE color space system classifies colors according to the human vision system. This system specifies colors in CIE coordinates and is a standard for comparing one color in the CIE coordinates with another.

CIE 1976 L\*a\*b\* is one of the CIE-based color spaces and is a way to linearize the perceptibility of color differences. The nonlinear relations for L\*, a\*, and b\* mimic the logarithmic response of the eye.

## CMY Color Space

CMY is another set of familiar primary colors: cyan, magenta, and yellow. CMY is a subtractive color space in which these primary colors are subtracted from white light to produce the desired color. The CMY color space is the basis of most color printing and photography processes. CMY is the complement of the RGB color space because cyan, magenta, and yellow are the complements of red, green, and blue.

## YIQ Color Space

The YIQ space is the primary color space adopted by the National Television System Committee (NTSC) for color TV broadcasting. It is a linear transformation of the RGB cube for transmission efficiency and for maintaining compatibility with monochrome television standards. The Y component of the YIQ system provides all the video information that a monochrome television set requires. The main advantage of the YIQ space for image processing is that the luminance information (Y) is de-coupled from the color information (I and Q). Because luminance is proportional to the amount of light perceived by the eye, modifications to the grayscale appearance of the image do not affect the color information.

## In-Depth Discussion

There are standard ways to convert RGB to grayscale and to convert one color space to another. The transformation from RGB to grayscale is linear. However, some transformations from one color space to another are nonlinear because some color spaces represent colors that cannot be represented in other spaces.

## RGB To Grayscale

The following equations convert an RGB image into a grayscale image on a pixel-by-pixel basis.

$$\text{grayscale value} = 0.299R + .587G + 0.114B$$

This equation is part of the NTSC standard for luminance. An alternative conversion from RGB to grayscale is a simple average:

$$\text{grayscale value} = (R + G + B) / 3$$

## RGB and HSL

There is no matrix operation that allows you to convert from the RGB color space to the HSL color space. The following equations describe the nonlinear transformation that maps the RGB color space to the HSL color space.

$$L = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

$$V2 = \sqrt{3} \times (G - B)$$

$$V1 = 2 \times R - G - B$$

$$H = 256 \times \tan^{-1}(V2 / V1) / (2 \times \Pi)$$

$$S = 255 \times (1 - 3 \times \min(R, G, B) / (R + G + B))$$

The following equations map the HSL color space to the RGB color space:

$$h = H \cdot \frac{2\pi}{256}$$

$$s = S / 255$$

$$s' = (1 - s) / 3$$

$$f(h) = (1 - s \cdot \cos(h)) / \cos(\pi / 3 - h) / 3$$

$$\left. \begin{array}{l} b = s' \\ r = f(h) \\ g = 1 - r - b \end{array} \right\} [0 < h \leq 2\pi / 3]$$

$$\left. \begin{array}{l} h' = h - 2\pi / 3 \\ r = s' \\ g = f(h') \\ b = 1 - r - g \end{array} \right\} [2\pi / 3 < h \leq 4\pi / 3]$$

$$\left. \begin{array}{l} h' = h - 4\pi / 3 \\ g = s' \\ b = f(h') \\ r = 1 - g - b \end{array} \right\} [4\pi / 3 < h \leq 2\pi]$$

$$l = 0.299.r + 0.587.g + 0.114.b$$

$$l' = L / l$$

$$R = r.l'$$

$$G = g.l'$$

$$B = b.l'$$

## RGB and CIE L\*a\*b\*

Before transforming RGB to CIE L\*a\*b\*, you must transform the RGB values into an intermediate color space—the CIE XYZ space. The following  $3 \times 3$  matrix converts RGB to CIE XYZ.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Then use the following equations to convert the CIE XYZ values into the CIE L\*a\*b\* values:

$$L^* = 116 \times (Y/Yn)^{1/3} - 16 \text{ for } Y/Yn > 0.008856$$

$$L^* = 903.3 \times Y/Yn \text{ otherwise}$$

$$a^* = 500 \times (f(X/Xn) - f(Y/Yn))$$

$$b^* = 200 \times (f(Y/Yn) - f(Z/Zn))$$

where  $f(t) = t^{1/3}$  for  $t > 0.008856$

$f(t) = 7.787 \times t + 16/116$  otherwise

Here  $Xn$ ,  $Yn$  and  $Zn$  are the tri-stimulus values of the reference white.

To transform CIE L\*a\*b\* values to RGB, first convert the CIE L\*a\*b\* values to CIE XYZ using the following equations:

$$X = Xn \times (P + a^* / 500)^3$$

$$Y = Yn \times P^3$$

$$Z = Zn \times (P - b^* / 200)^3$$

where  $P = (L^* + 16) / 116$

Then use the following matrix operation to convert the CIE XYZ values to RGB values:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

## RGB and CMY

The following matrix operation converts the RGB color space to the CMY color space:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Normalize all color values to lie between 0 and 1 before using this conversion equation. To obtain RGB values from a set of CMY values, subtract the individual CMY values from 1.

## RGB and YIQ

The following matrix operation converts the RGB color space to the YIQ color space:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The following matrix operation converts the YIQ color space to the RGB color space:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.956 & 0.621 \\ 1.0 & -0.272 & -0.647 \\ 1.0 & -1.105 & 1.702 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

# Display

This chapter contains information about image display, palettes, regions of interest, and nondestructive overlays.

## Image Display

Displaying images is an important component of a vision application because it gives you the ability to visualize your data. *Image processing* and *image visualization* are distinct and separate elements. Image processing refers to the creation, acquisition, and analysis of images. Image visualization refers to how image data is presented and how you can interact with the visualized images. A typical imaging application uses many more images in memory than the number of display windows.

## Image Display Concepts

Display functions display images in external image windows, set the attributes of the windows, assign color palettes to image windows, close windows, and setup and use an image browser in an external window. Some *region of interest* (ROI) functions—a subset of the display functions—interactively define ROIs in external image windows. These ROI functions configure and display different drawing tools, detect draw events in windows, retrieve information about the region drawn on the window, and move and rotate ROIs. Nondestructive overlays display important information on top of an image without changing the values of the image pixels.

## When to Use

Use display functions to visualize your image data, retrieve generated events and the associated data from an image window, select ROIs from an image interactively, and annotate the image with additional information.

## In-Depth Discussion

This section describes the display modes available in IMAQ Vision and the 16-bit grayscale display mapping methods.

### Display Modes

One of the key components of displaying images is the display mode that the video adaptor operates. The display mode indicates how many bits specify the color of a pixel on the display screen. Generally, the display mode available from a video adaptor ranges from 8 bits to 32 bits per pixel, depending the amount of video memory available on the video adaptor and the screen resolution you choose.

If you have an 8-bit display mode, a pixel can be one of 256 different colors. If you have a 16-bit display mode, a pixel can be one of 65,536 colors. In 24-bit or 32-bit display mode, the color of a pixel on the screen is encoded using 3 or 4 bytes, respectively. In these modes, information is stored using 8 bits each for the red, green, and blue components of the pixel. These modes offer the possibility to display about 16.7 million colors.

Understanding your display mode is important to understanding how IMAQ Vision displays the different image types on a screen. Image processing functions often use grayscale images. Because display screen pixels are made of red, green, and blue components, the pixels of a grayscale image cannot be rendered directly.

In 24-bit or 32-bit display mode, the display adaptor uses 8 bits to encode a grayscale value, offering 256 gray shades. This color resolution is sufficient to display 8-bit grayscale images. However, higher bit depth images, such as 16-bit grayscale images, are not accurately represented in 24-bit or 32-bit display mode. To display a 16-bit grayscale image, either ignore the least significant bits or use a mapping function to convert 16 bits to 8 bits.

## Mapping Methods for 16-Bit Image Display

The following techniques describe how IMAQ Vision converts 16-bit images to 8-bit images and displays them using mapping functions.

Mapping functions evenly distribute the dynamic range of the 16-bit image to an 8-bit image.

- Full Dynamic—The minimum intensity value of the 16-bit image is mapped to 0, and the maximum intensity value is mapped to 255. All other values in the image are mapped to lie between 0 and 255 using the equation shown below. This mapping method is general purpose because it insures the display of the complete dynamic range of the image. Because the minimum and maximum pixel values in an image are used to determine the full dynamic range of that image, the presence of noisy or defective pixels (for non-Class A sensors) with minimum or maximum values can affect the appearance of the displayed image. IMAQ Vision uses this technique by default.

$$z = \frac{x - y}{v - y} \times 255$$

where  $z$  is the 8-bit pixel value

$x$  is the 16-bit value

$y$  is the minimum intensity value

$v$  is the maximum intensity value

- 90% Dynamic—The intensity corresponding to 5% of the cumulative histogram is mapped to 0, the intensity corresponding to 95% of the cumulated histogram is mapped to 255. Values in the 0 to 5% range are mapped to 0, while values in the 95 to 100% range are mapped to 255. This mapping method is more robust than the full dynamic method and is not sensitive to small aberrations in the image. This method requires the computation of the cumulative histogram or an estimate of the histogram. See Chapter 4, *Image Analysis*, for more information on histograms.
- Given Percent Range—This method is similar to the 90% Dynamic method, except that the minimum and maximum percentages of the cumulative histogram that the software maps to 8-bit are user-defined.
- Given Range—This technique is similar to the Full Dynamic method, except that the minimum and maximum values to be mapped to 0 and 255 are user-defined. You can use this method to enhance the contrast of some regions of the image by finding the minimum and maximum values of those regions and computing the histogram of those regions. A histogram of this region shows the minimum and maximum

intensities of the pixels. Those values are used to stretch the dynamic range of the entire image.

- Downshifts—This technique is based on shifts of the pixel values. This method applies a given number of right shifts to the 16-bit pixel value and displays the least significant byte. This technique truncates some of the lowest bits, which are not displayed. This method is very fast, but it reduces the real dynamic of the sensor to 8-bit sensor capabilities. It requires knowledge of the bit-depth of the imaging sensor that has been used. For example, an image acquired with a 12-bit camera should be visualized using 4 right shifts in order to display the 8 most-significant bits acquired with the camera. If you are using an IMAQ image acquisition device, this technique is the default used by Measurement and Automation Explorer.

## Palettes

---

At the time a grayscale image is displayed on the screen, IMAQ Vision converts the value of each pixel of the image into red, green, and blue intensities for the corresponding pixel displayed on the screen. This process uses a color table, called a *palette*, which associates a color to each possible grayscale value of an image. IMAQ Vision provides the capability to customize the palette used to display an 8-bit grayscale image.

### When to Use

With palettes, you can produce different visual representations of an image without altering the pixel data. Palettes can generate effects, such as photonegative displays or color-coded displays. In the latter case, palettes are useful for detailing particular image constituents in which the total number of colors is limited.

Displaying images in different palettes helps emphasize regions with particular intensities, identify smooth or abrupt gray-level variations, and convey details that might be difficult to perceive in a grayscale image. For example, the human eye is much more sensitive to small intensity variations in a bright area than in a dark area. Using a color palette may help you distinguish these slight changes.

## Concepts

A palette is a pre-defined or user-defined array of RGB values. It defines for each possible gray-level value a corresponding color value to render the pixel. The gray-level value of a pixel acts as an address that is indexed into the table, returning three values corresponding to a red, green, and blue (RGB) intensity. This set of RGB values defines a palette in which varying amounts of red, green, and blue are mixed to produce a color representation of the value range.

In the case of 8-bit grayscale images, pixels can take  $2^8$  or 256 values ranging from 0 to 255. Color palettes are composed of 256 RGB elements. A specific color is the result of applying a value between 0 and 255 for each of the three color components (red, green, and blue). If the red, green, and blue components have an identical value, the result is a gray level pixel value.

A gray palette associates different shades of gray to each value so as to produce a linear and continuous gradation of gray, from black to white. You can set up the palette to assign the color black to the value 0 and white to 255, or vice versa. Other palettes can reflect linear or nonlinear gradations going from red to blue, light brown to dark brown, and so on.

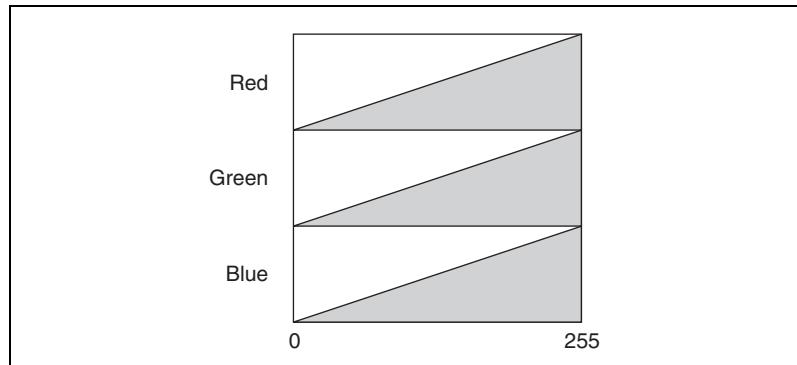
IMAQ Vision has five predefined color palettes. Each palette emphasizes different shades of gray.

## In-Depth Discussion

The following sections introduce the five predefined palettes available in IMAQ Vision. The graphs in each section represent the color tables used by each palette. The horizontal axes of the graphs represent the input gray-level range [0, 255], and the vertical axes represent the RGB intensities assigned to a given gray-level value.

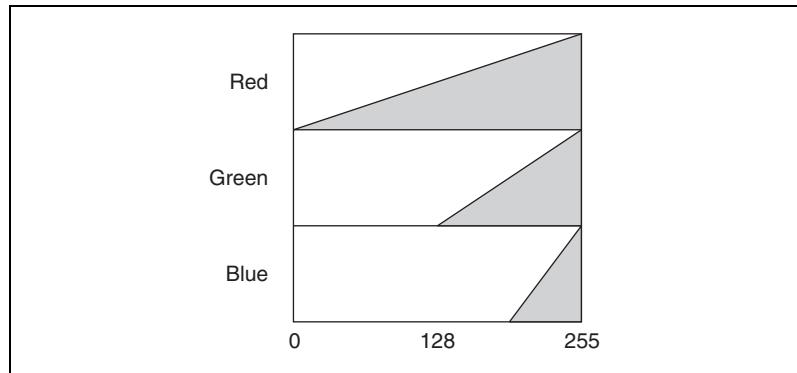
### Gray Palette

This palette has a gradual gray-level variation from black to white. Each value is assigned to an equal amount of Red, Green, and Blue in order to produce a gray-level.



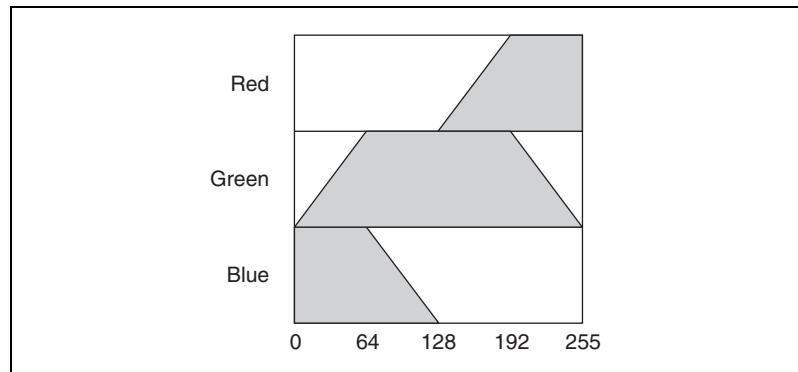
## Temperature Palette

This palette has a gradation from light brown to dark brown. 0 is black and 255 is white.



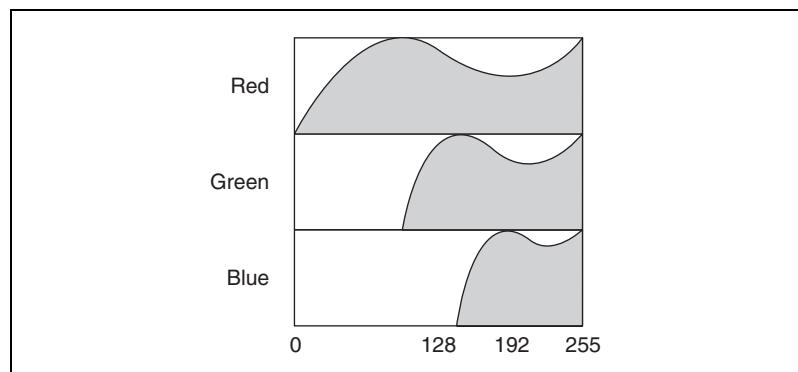
## Rainbow Palette

This palette has a gradation from blue to red with a prominent range of greens in the middle value range. 0 is blue and 255 is red.



## Gradient Palette

This palette has a gradation from red to white with a prominent range of light blue in the upper value range. 0 is black and 255 is white.

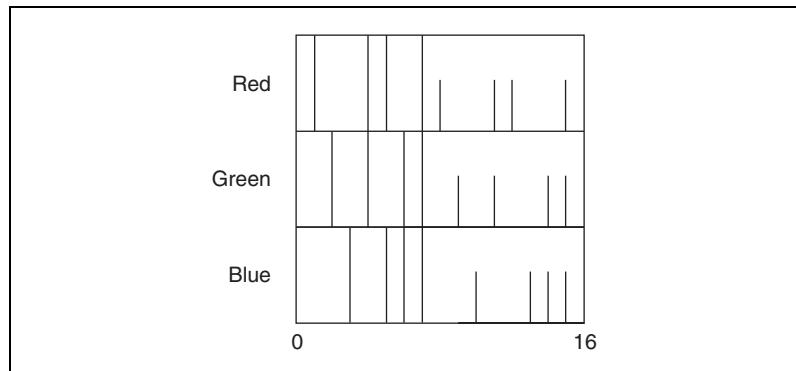


## Binary Palette

This palette has 16 cycles of 16 different colors. Table 2-1 illustrates where  $g$  is the gray-level value.

**Table 2-1.** Gray-Level Values in the Binary Palette

<b><math>g=</math></b>	<b>R</b>	<b>G</b>	<b>B</b>	<b>Resulting Color</b>
0	0	0	0	Black
1	255	0	0	Red
2	0	255	0	Green
3	0	0	255	Blue
4	255	255	0	Yellow
5	255	0	255	Purple
6	0	255	255	Aqua
7	255	127	0	Orange
8	255	0	127	Magenta
9	127	255	0	Bright green
10	127	0	255	Violet
11	0	127	255	Sky blue
12	0	255	127	Sea green
13	255	127	127	Rose
14	127	255	127	Spring green
15	127	127	255	Periwinkle



This periodic palette is appropriate for the display of binary and labeled images.

## Regions of Interest

---

A *region of interest* (ROI) is an area of an image in which you want to perform your image analysis.

### When to Use

Use ROIs to focus your processing and analysis on part of an image. You can define an ROI using standard contours, such as an oval or rectangle, or freehand contours. You can also perform any of the following options:

- Construct an ROI in an image window
- Associate an ROI with an image window
- Extract an ROI associated with an image window
- Erase the current ROI from an image window
- Transform an ROI into an image mask
- Transform an image mask into an ROI

## ROI Concepts

An ROI describes a region or multiple regions of an image in which you want to focus your processing and analysis. These regions are defined by specific contours. IMAQ Vision supports the following contour types:

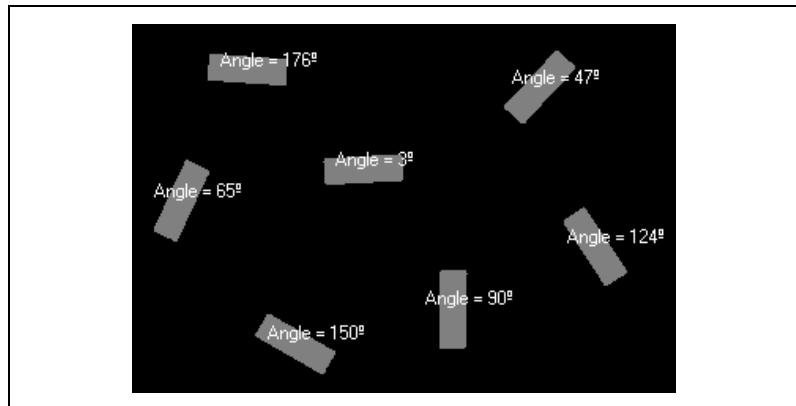
**Table 2-2.** Types of Contours an ROI May Contain

Icon	Contour Name
	Point
	Line
	Rectangle
	Rotated Rectangle
	Oval
	Annulus
	Broken Line
	Polygon
	Freehand Line
	Freehand

You can define an ROI interactively, programmatically, or with an image mask. Define an ROI interactively by using the tools from the tools palette to draw an ROI on a displayed image. For more information about defining ROIs programmatically or with an image mask, see the *IMAQ Vision for LabVIEW User Manual* or the *IMAQ Vision for Measurement Studio User Manual*.

# Nondestructive Overlay

A nondestructive overlay enables you to annotate the display of an image with useful information without actually modifying the image. You can overlay text, lines, points, complex geometric shapes, and bitmaps on top of your image without changing the underlying pixel values in your image; only the display of the image is affected. Figure 2-1 shows how you can use the overlay to depict the orientation of each particle in the image.



**Figure 2-1.** Nondestructive Overlay

## When to Use

You can use nondestructive overlays for many purposes, such as the following:

- Highlighting the location in an image where objects have been detected
- Adding quantitative or qualitative information to the displayed image—like the match score from a pattern matching function
- Displaying ruler grids or alignment marks

## Nondestructive Overlay Concepts

Overlays do not affect the results of any analysis or processing functions—they affect only the display. The overlay is associated with an image, so there are no special overlay data types. You only need to add the overlay to your image. IMAQ Vision clears the overlay anytime you change the size or orientation of the image because the overlay ceases to have meaning. You can save overlays with images using the PNG file format.

---

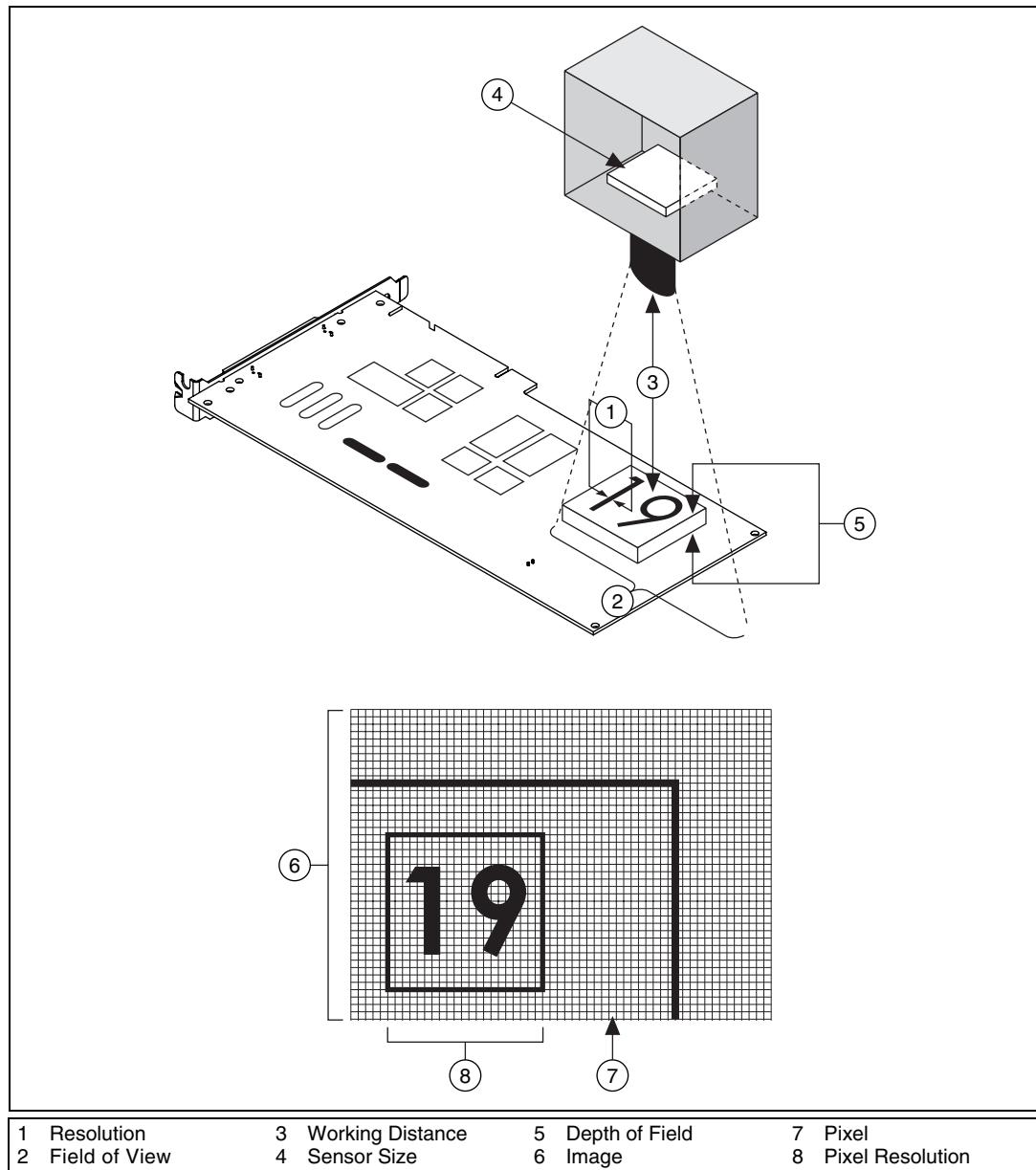
# System Setup and Calibration

This chapter describes how to setup an imaging system and calibrate the imaging setup so that you can convert pixel coordinates to real-world coordinates. Converting pixel coordinates to real-world coordinates is useful when you need to make accurate measurements from inspection images using real-world units.

## Setting Up Your Imaging System

---

Before you acquire, analyze, and process images, you must set up your imaging system. Five factors comprise a imaging system: field of view, working distance, resolution, depth of field, and sensor size. Figure 3-1 illustrates these concepts.



**Figure 3-1.** Fundamental Parameters of an Imaging System

- Resolution—the smallest feature size on your object that the imaging system can distinguish
- Pixel resolution—the minimum number of pixels needed to represent the object under inspection
- Field of view—the area of the object under inspection that the camera can acquire
- Working distance—the distance from the front of the camera lens to the object under inspection
- Sensor size—the size of a sensor's active area, typically defined by the sensor's horizontal dimension
- Depth of field—the maximum object depth that remains in focus

For additional information about the fundamental parameters of an imaging system, see the *Application Notes* section of the Edmund Industrial Optics *Electronic Imaging Resource Guide*, or visit Edmund Industrial Optics at [edmundoptics.com](http://edmundoptics.com)

## Acquiring Quality Images

The manner in which you set up your system depends on the type of analysis and processing you need to do. Your imaging system should produce images with high enough quality so that you can extract the information you need from the images. Five factors contribute to overall image quality: resolution, contrast, depth of field, perspective, and distortion.

### Resolution

There are two kinds of resolution to consider when setting up your imaging system: pixel resolution and resolution. Pixel resolution refers to the minimum number of pixels you need to represent the object under inspection. You can determine the pixel resolution you need by the smallest feature you need to inspect. Try to have at least two pixels represent the smallest feature. You can use the following equation to determine the minimum pixel resolution required by your imaging system:

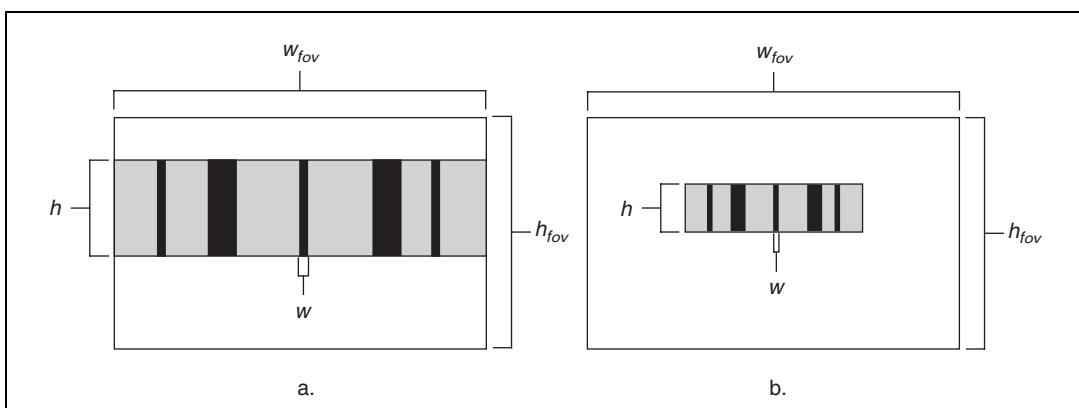
$$(length\ of\ object's\ longest\ axis / size\ of\ object's\ smallest\ feature) \times 2$$

If the object does not occupy the entire field of view, the image size will be greater than the pixel resolution.

Resolution indicates the amount of object detail that the imaging system can reproduce. Images with low resolution lack detail and often appear blurry. Three factors contribute to the resolution of your imaging system: field of view, the camera sensor size, and number of pixels in the sensor. Once you know these three factors, you can determine the focal length of your camera lens.

## Field of View

The field of view is the area of the object under inspection that the camera can acquire. Figure 3-2 describes the relationship between pixel resolution and the field of view.



**Figure 3-2.** Relationship Between Pixel Resolution and Field of View

Figure 3-2a shows an object that occupies the field of view. Figure 3-2b shows an object that occupies less space than the field of view. If  $w$  is the size of the smallest feature in the  $x$  direction and  $h$  is the size of the smallest feature in the  $y$  direction, the minimum  $x$  pixel resolution is

$$\frac{w_{fov}}{w} \times 2$$

and the minimum  $y$  pixel resolution is

$$\frac{h_{fov}}{h} \times 2.$$

Choose the larger pixel resolution of the two for your imaging application.



**Note** In Figure 3-2b, the image size is larger than the pixel resolution.

## Sensor Size and Number of Pixels in the Sensor

The camera sensor size is important in determining your field of view, which is a key element in determining your minimum resolution requirement. The sensor's diagonal length specifies the size of the sensor's active area. The number of pixels in your sensor should be greater than or equal to the pixel resolution. Choose a camera with a sensor that satisfies your minimum resolution requirement.

## Lens Focal Length

Once you determine the field of view and appropriate sensor size, you can decide which type of camera lens meets your imaging needs. A lens is defined primarily by its focal length. The relationship between the lens, field of view, and sensor size is as follows:

$$\text{focal length} = (\text{sensor size} \times \text{working distance}) / \text{field of view}$$

If you cannot change the working distance, you are limited in choosing a focal length for your lens. If you have a fixed working distance and your focal length is short, your images may appear distorted. However, if you have the flexibility to change your working distance, modify the distance so that you can select a lens with the appropriate focal length and minimize distortion.

## Contrast

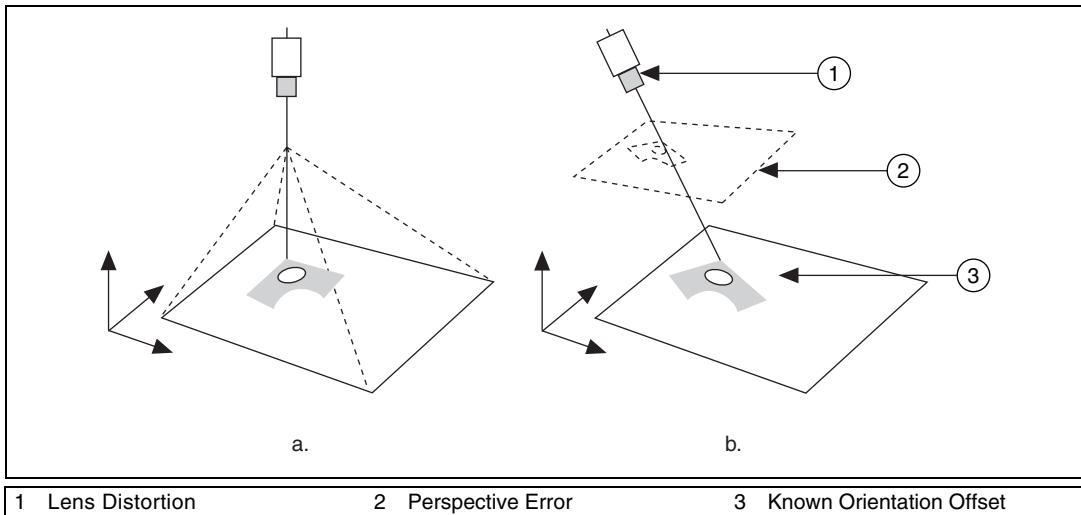
Resolution and contrast are closely related factors contributing to image quality. Contrast defines the differences in intensity values between the object under inspection and the background. Your imaging system should have enough contrast to distinguish objects from the background. Proper lighting techniques can enhance the contrast of your system.

## Depth of Field

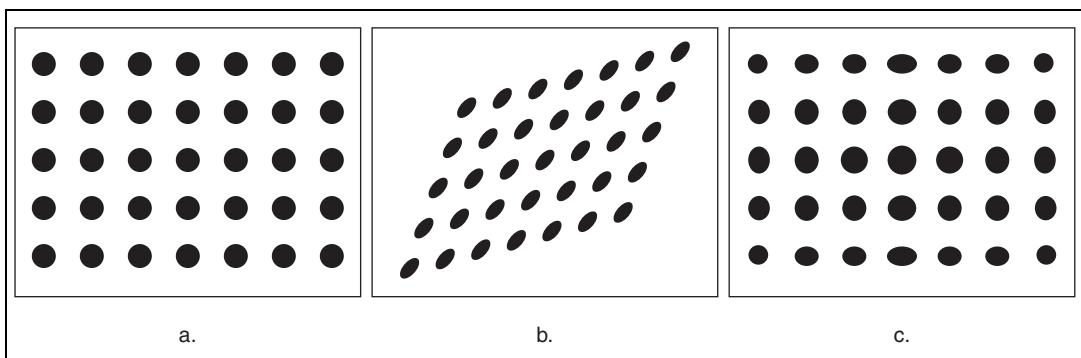
The depth of field of a lens is its ability to keep objects of varying heights in focus. If you need to inspect objects with various heights, chose a lens that can maintain the image quality you need as the objects move closer to and further from the lens.

## Perspective

Perspective errors often occur when the camera axis is not perpendicular to the object you are inspecting. Figure 3-3a shows an ideal camera position. Figure 3-3b shows a camera imaging an object from an angle.

**Figure 3-3.** Camera Angle Relative to the Object Under Inspection

Perspective errors appear as changes in the object's magnification depending on the object's distance from the lens. Figure 3-4a illustrates perspective errors caused by a camera imaging the grid from an angle.

**Figure 3-4.** Perspective and Distortion Errors

Try to position your camera perpendicular to the object you are trying to inspect to reduce perspective errors. If you need to take precise measurements from your image, correct perspective error by applying calibration techniques to your image.

## Distortion

Nonlinear distortion is a geometric aberration caused by optical errors in the camera lens. A typical camera lens introduces radial distortion. This causes points that are away from the lens's optical center to appear further away from the center than they really are. Figure 3-4c illustrates the effect of distortion on a grid of dots. When distortion occurs, information in the image is misplaced relative to the center of the field of view, but the information is not necessarily lost. Therefore, you can undistort your image through spatial calibration.

# Spatial Calibration

---

Spatial calibration is the process of computing pixel to real-world unit transformations while accounting for many errors inherent to the imaging setup. Calibrating your imaging setup is important when you need to make accurate measurements in real-world units.

An image contains information in the form of pixels. Spatial calibration allows you to translate a measurement from pixel units into another unit, such as inches or centimeters. This conversion is easy if you know a conversion ratio between pixels and real-world units. For example, if 1 pixel equals 1 inch, a length measurement of 10 pixels equals 10 inches.

This conversion may not be straightforward since perspective projection and lens distortion affect the measurement in pixels. Calibration accounts for possible errors by constructing mappings that you can use to convert between pixel and real world units. You can also use the calibration information to correct perspective or nonlinear distortion errors for image display and shape measurements.

## When to Use

Calibrate your imaging system when you need to make accurate and reliable measurements. Use the IMAQ Vision calibration tools to do the following:

- Calibrate your imaging setup automatically by imaging a standard pattern (calibration template) or by providing reference points
- Convert measurements (lengths, areas, widths) from real-world units to pixel units and back

- Apply a learned calibration mapping to correct an image acquired through a calibrated setup
- Assign an arbitrary coordinate system to measure positions in real world units



**Note** If the camera axis is as close to 90° from the object plane as possible, you can ignore errors caused by perspective and estimate your measurements.

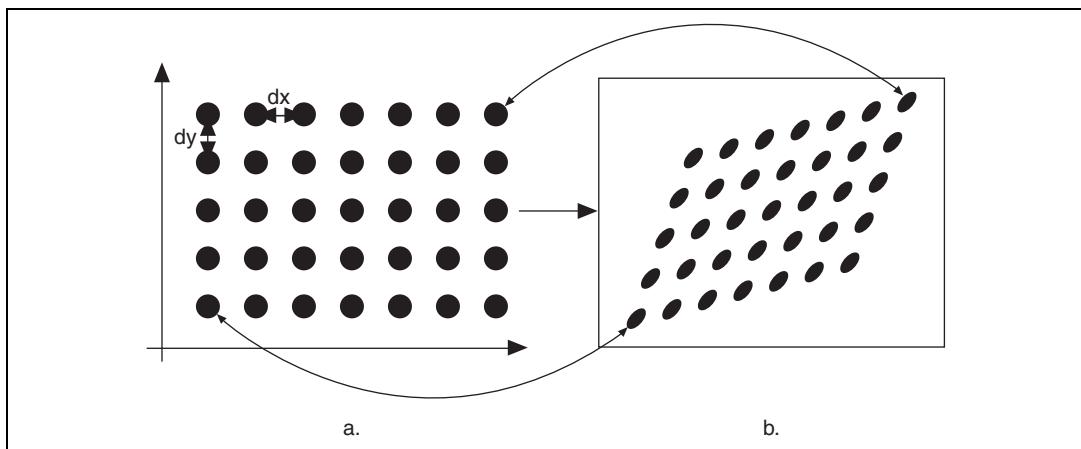
## Concepts

To calibrate an imaging setup, the calibration software uses a set of known mappings between points in the image and their corresponding locations in the real world. The calibration software uses these known mappings to compute the pixel to real-world mapping for the entire image. The resulting calibration information is valid only for the imaging setup that you used to create the mapping. Any change in the imaging setup that violates the mapping information compromises the accuracy of the calibration information.

## Calibration Process

The calibration software requires a list of known pixel to real-world mappings to compute calibration information for the entire image. You can specify the list in two ways:

- Image a grid of dots similar to the one shown in Figure 3-5a. Input the dx and dy spacing between the dots in real-world units. The calibration software uses the image of the grid, shown in Figure 3-5b, and the spacing between the dots in the grid to generate the list of pixel to real-world mappings required for the calibration process.
- Input a list of real world points and the corresponding pixel coordinates directly to the calibration software.

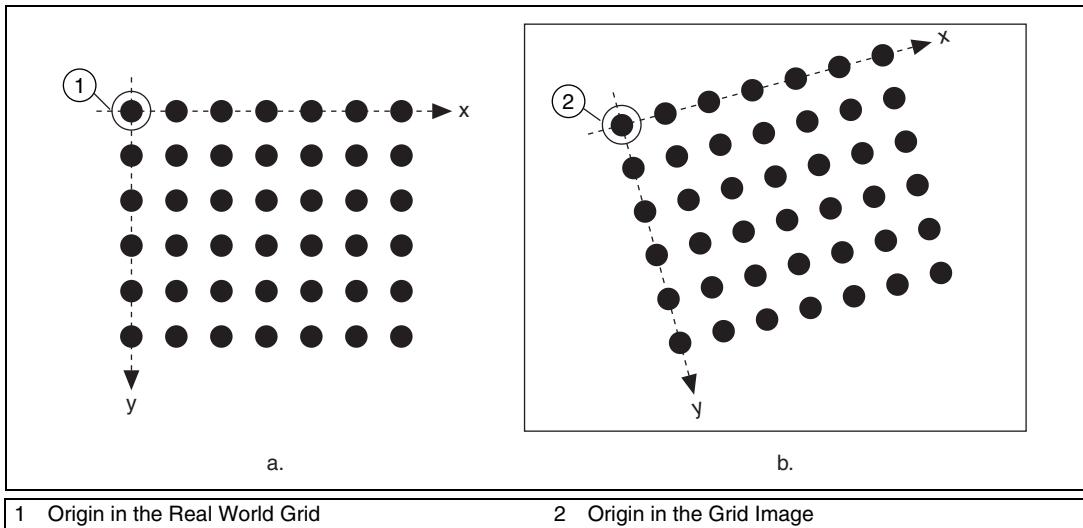
**Figure 3-5. Calibration Setup**

The calibration process uses the list of pixel to real-world mappings and a user-defined algorithm to create a mapping for the entire image. The calibration software also generates an error map. An error map returns an estimate of the worst case error when a pixel coordinate is transformed into a real-world coordinate.

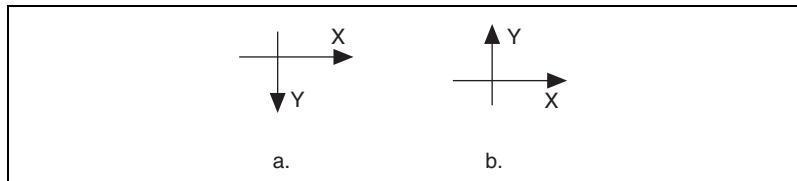
Use the calibration information obtained from the calibration process to convert any pixel coordinate to its real-world coordinate and back.

## Coordinate System

To express measurements in real-world units, you must define a coordinate system. Define a coordinate system by its origin, angle, and axis direction. Figure 3-6a shows the coordinate system of a calibration grid in the real world. Figure 3-6b shows the coordinate system of an image of the corresponding calibration grid. The origin, expressed in pixels, defines the center of your coordinate system. The origins of the coordinate systems depicted in Figure 3-6 lie at the center of the circled dots. The angle specifies the orientation of your coordinate system with respect to the horizontal axis in the real world. Notice in Figure 3-6b that the horizontal axis automatically aligns to the top row of dots in the image of the grid. The calibration procedure determines the direction of the horizontal axis in the real world, which is along the topmost row of dots in the image of the grid.

**Figure 3-6.** Origin and Angle of a Coordinate System

The second axis direction can either be indirect, as shown in Figure 3-7a, or direct, as shown in Figure 3-7b. The indirect axis orientation corresponds to the way a coordinate system is present in digital images. The direct axis orientation corresponds to the way a coordinate system is present in the real world.

**Figure 3-7.** Axis Direction of a Coordinate System

If you do not specify a coordinate system, the calibration process defines a default coordinate system, as follows:

1. The origin is placed at the center of the left, topmost dot in the calibration grid.
2. The angle is set to zero. This aligns the x-axis with the topmost row of dots in the grid.
3. The axis direction is set to indirect. This aligns the y-axis to the leftmost column of the dots in the grid.

If you specify a list of points instead of a grid for the calibration process, the software defines a default coordinate system, as follows:

1. The origin is placed at the point in the list with the lowest x-coordinate value and then the lowest y-coordinate value.
2. The angle is set to zero.
3. The axis direction is set to indirect.

If you define a coordinate system yourself, remember the following:

- Express the origin in pixels. Always choose an origin location that lies within the calibration grid so that you can convert the location to real-world units.
- Specify the angle as the angle between the new coordinate system and the horizontal direction in the real world. If your imaging system has perspective errors but no lens distortion, this angle can be visualized as shown in Figure 3-11. However, if your images exhibit nonlinear distortion, visualizing the coordinate system in the image is not trivial.

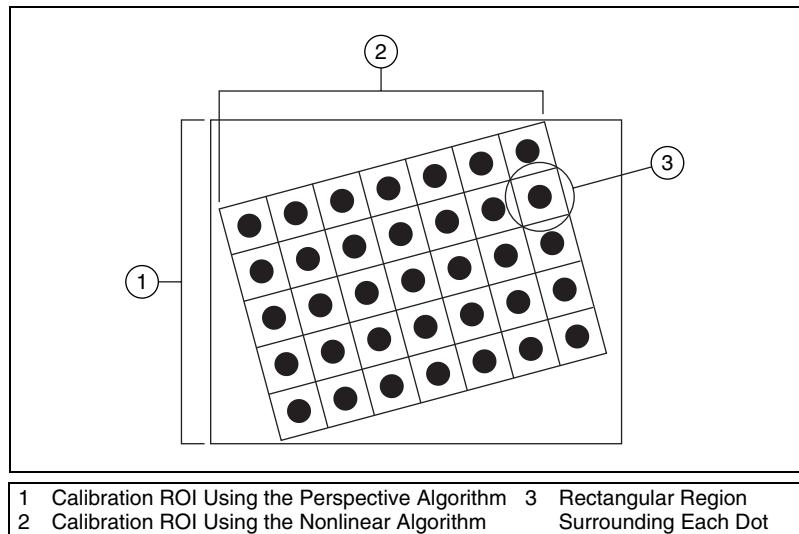
## Calibration Algorithms

IMAQ Vision has two algorithms for calibration: *perspective* and *nonlinear*. Perspective calibration corrects for perspective errors, and nonlinear calibration corrects for perspective errors and nonlinear lens distortion. Learning for perspective is faster than learning for nonlinear distortion.

The perspective algorithm computes one pixel to real-world mapping for the entire image. You can use this mapping to convert the coordinates of any pixel in the image to real-world units.

The nonlinear algorithm computes pixel to real-world mappings in a rectangular region centered around each dot in the calibration grid, as shown in Figure 3-8. IMAQ Vision estimates the mapping information around each dot based on its neighboring dots. You can convert pixel units to real-world units within the area covered by the grid dots. Because IMAQ Vision computes the mappings around each dot, only the area in the image covered by the grid dots is calibrated accurately.

The *calibration ROI* output of the calibration function defines the region of the image in which the calibration information is accurate. The calibration ROI in the perspective method encompasses the entire image. The calibration ROI in the nonlinear method encompasses the bounding rectangle that encloses all the rectangular regions around the grid dots. Figure 3-8 illustrates the calibration ROI concept.



**Figure 3-8.** Calibrating ROIs



**Note** You can convert pixels that lie outside the calibration ROI to real-world units, but these conversions may be inaccurate.

## Calibration Quality Information

The *quality score* and *error map* outputs of the calibration function indicate how well your system is calibrated.

The quality score, which ranges from 0 to 1000, reflects how well the calibration function learned the grid or set of points you provided. The quality score does not reflect the accuracy of the calibration, but describes how well the calibration mapping adapts to the learned grid or feature points.

Use the quality score to determine whether the calibration algorithm you chose is adequate. IMAQ Vision returns a low quality score if you calibrate an image with high nonlinear distortion using the perspective method or if you use a sparse grid to calibrate an image with high nonlinear distortion.

You can also use this score to gauge whether the setup is behaving as expected. For example, if you are using a lens with very little lens distortion, perspective calibration should produce accurate results. However, system setup problems, such as a physically distorted calibration template, may cause a low quality score regardless of your lens quality.

The error map is an estimate of the positional error that you can expect when you convert a pixel coordinate into a real-world coordinate. The error map is a 2D array that contains the expected positional error for each pixel in the image. The error value of the pixel coordinate  $(i, j)$  indicates the largest possible location error for the estimated real-world coordinate  $(x, y)$  as compared to the true real-world location. The following equation shows how to calculate the error value.

$$e(i, j) = \sqrt{(x - x_{true})^2 + (y - y_{true})^2}$$

The error value indicates the radical distance from the true real world position in which the estimated real world coordinates can live. The error value has a confidence interval of 95%, which implies that the positional error of the estimated real-world coordinate is equal to or smaller than the error value 95% of the time. A pixel coordinate with a small error value indicates that its estimated real-world coordinate is computed very accurately. A large error value indicates that the estimated real-world coordinate for a pixel may not be accurate.

Use the error map to determine whether your imaging setup and calibration information satisfy the accuracy requirements of your inspection application. If the error values are greater than the positional errors that your application can tolerate, you need to improve your imaging setup. An imaging system with high lens distortion usually results in an error map with high values. If you are using a lens with considerable distortion, you can use the error map to determine the position of the pixels that satisfy your application's accuracy requirements. Because the effect of lens distortion increases towards the image borders, pixels close to the center of the image have lower error values than the pixels at the image borders.

## Image Correction

Image correction involves transforming a distorted image acquired in a calibrated setup into an image where perspective errors and lens distortion are corrected. IMAQ Vision corrects an image by applying the transformation from pixel to real-world coordinates for each pixel in the input image. Then IMAQ Vision applies simple shift and scaling transformations to position the real-world coordinates into a new image. IMAQ Vision uses interpolation during the scaling process to generate the new image.

When you learn for correction, you have the option of constructing a correction table. The correction table is a lookup table stored in memory that contains the real-world location information of all the pixels in the image. The lookup table greatly increases the speed of image correction but requires more memory and increases your learning time. Use this option when you want to correct several images at a time in your vision application.

## Scaling Mode

The scaling mode defines how to scale a corrected image. Two scaling mode options are available: scale to fit and scale to preserve area.

Figure 3-9 illustrates the scaling modes. Figure 3-9a shows the original image. With the scale to fit option, the corrected image is scaled to fit in an image the same size as the original image, as shown in Figure 3-9b. With the scale to preserve area option, the corrected image is scaled such that features in the image retain the same area as they did in the original image, as shown in Figure 3-9c. Images that are scaled to preserve area are usually larger than the original image. Because scaling to preserve the area increases the size of the image, the processing time for the function may increase.

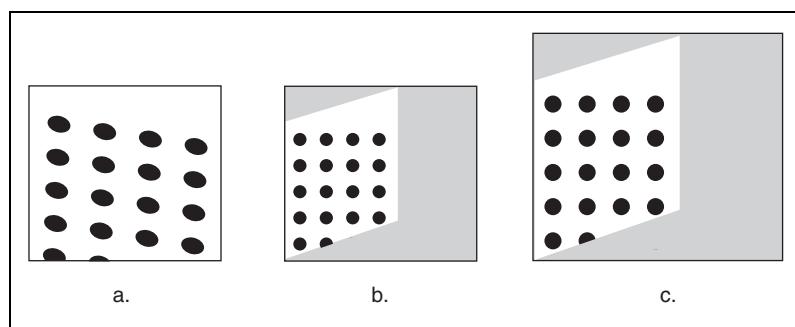
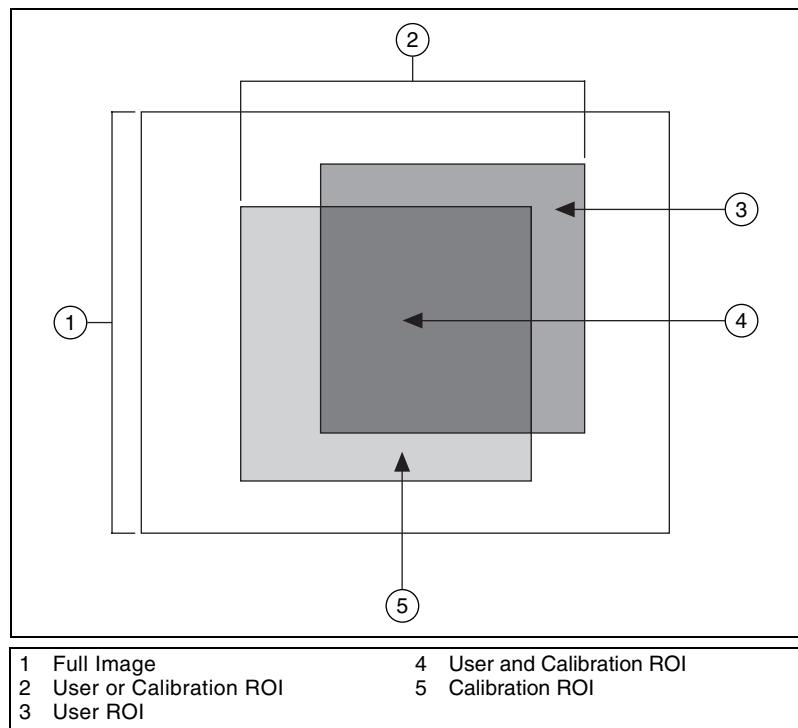


Figure 3-9. Scaling Modes

The scaling mode you choose depends on your application. Scale to preserve the area when your vision application requires the true area of objects in the image. Use scale to fit for all other vision applications.

## Correction Region

You can correct an entire image or regions in the image based on user-defined ROIs or the calibration ROI defined by the calibration software. Figure 3-10 illustrates the different image areas you can specify for correction. IMAQ Vision learns calibration information for only the regions you specify.



**Figure 3-10.** ROI Modes

- Full Image—Corrects the entire image regardless of the calibration ROI and the user-defined ROI
- User or Calibration ROI—Corrects pixels in both the user-defined ROI and the calibration ROI

- User ROI—Corrects only the pixels inside the user-defined ROI specified during the learn calibration phase
- User and Calibration ROI—Corrects only the pixels that lie in the intersection of the user-defined ROI and the calibration ROI
- Calibration ROI—Corrects only the pixels inside the calibration ROI, which the calibration algorithm choice outputs. If you set the scaling mode to scale to fit, this ROI is fit in the whole image.

The valid coordinate indicates whether the pixel coordinate you are trying to map to a real-world coordinate lies within the image region you corrected. For example, if you corrected only the pixels within the calibration ROI but you try to map a pixel outside the calibration ROI to real-world coordinates, the Corrected Image Learn ROI parameter indicates an error.

## Simple Calibration

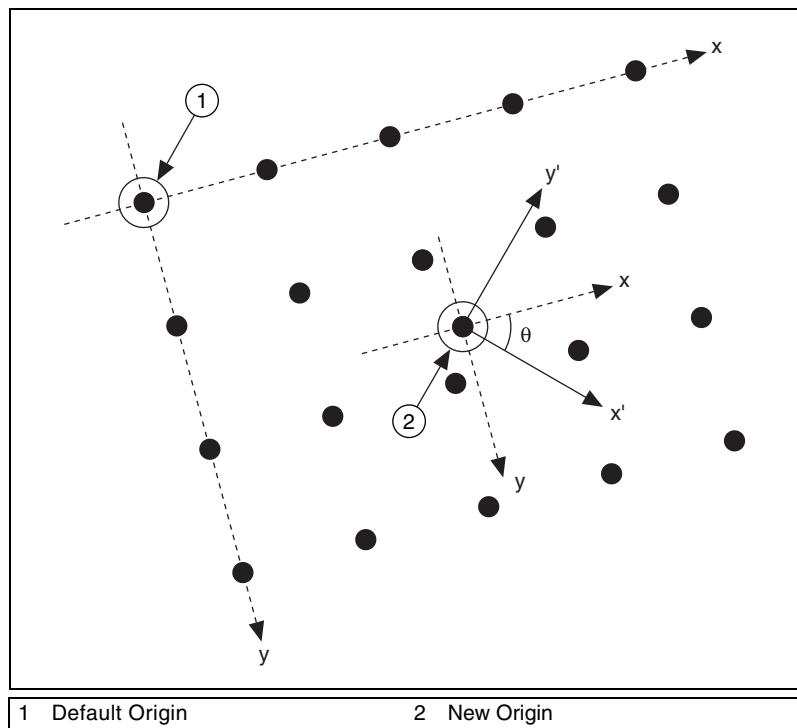
When your camera axis is perpendicular to the image plane and lens distortion is negligible, you can use simple calibration to calibrate your imaging setup. In simple calibration, a pixel coordinate is transformed to a real-world coordinate through scaling in the *x* and *y* (horizontal and vertical) directions.

Simple calibration maps pixel coordinates to real-world coordinates directly without a calibration grid. The software rotates and scales a pixel coordinate according to predefined coordinate reference and scaling factors.

To perform a simple calibration, define a coordinate system and scaling mode. Figure 3-11 illustrates how to define a coordinate system. To set a coordinate reference, define the angle between the *x*-axis and the horizontal axis of the image in degrees. Express the center as the position, in pixels, where you want the coordinate reference origin. Set the axis direction to direct or indirect. Set the scaling mode option to scale to fit or scale to preserve area. Simple Calibration also offers a correction table option.



**Note** If you use simple calibration with the angle set to 0, you do not need to learn for correction because you do not need to correct your image.



**Figure 3-11.** Defining a New Coordinate System

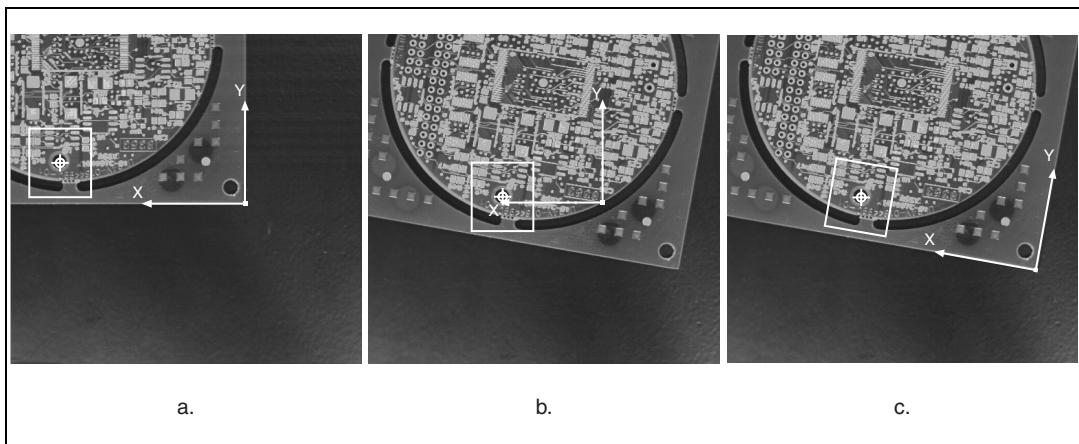
## Redefining a Coordinate System

You can use simple calibration to change the coordinate system assigned to a calibrated image. When you define a new coordinate system, remember the following:

- Express the origin in pixels. Always choose an origin location that lies within the calibration grid so that you can convert the location to real-world units.
- Specify the angle as the angle between the new coordinate system and the horizontal direction in the real world.

In some vision applications, you may need to image several regions of an object to inspect it completely. You can image these different regions by moving the object until the desired region lies under the camera or by moving the camera so that it lies above the desired region. In either case, each image maps to different regions in the real world. You can specify a new position for the origin and orientation of the coordinate system so that the origin lies on a point on the object under inspection.

Figure 3-12 shows an inspection application whose objective is to determine the location of the hole in the board with respect to the corner of the board. The board is on a stage that can translate in the x and y directions and can rotate about its center. The corner of the board is located at the center of the stage. In the initial setup, shown in Figure 3-12a, you define a coordinate system that aligns with the corner of the board using simple calibration. Specify the origin of the coordinate system as the location in pixels of the corner of the board, set the angle of the axis to 180°, and set the axis direction to indirect. Use pattern matching to find the location in pixels of the hole (indicated by the crosshair in Figure 3-12a). Convert the location of the hole in pixels to a real world location. This conversion returns the real world location of the hole with respect to the defined coordinate system. During the inspection process, the stage may translate and rotate by a known amount. This causes the board to occupy a new location in the camera's field of view, which makes the board appear translated and rotated in subsequent images, as shown in Figure 3-12b. Because the board has moved, the original coordinate system no longer aligns with the corner of the board. Therefore, measurements made using this coordinate system will be inaccurate. Use the information about how much the stage has moved to determine the new location of the corner of the board in the image and update the coordinate system using simple calibration to reflect this change. The origin of the updated coordinate reference system becomes the new pixel location of the corner of the board, and the angle of the coordinate system is the angle by which the stage has rotated. The updated coordinate system is shown in Figure 3-12c. Measurements made with the new coordinate system are accurate.



**Figure 3-12.** Moving coordinate system

# Part II

---

## Image Processing and Analysis

This section describes conceptual information about image analysis and processing, operators, and frequency domain analysis.

Part II, *Image Processing and Analysis*, contains the following chapters:

Chapter 4, *Image Analysis*, contains information about histograms, line profiles, and intensity measurements.

Chapter 5, *Image Processing*, contains information about lookup tables, kernels, spatial filtering, and grayscale morphology.

Chapter 6, *Operators*, contains information about arithmetic and logic operators, which mask, combine, and compare images.

Chapter 7, *Frequency Domain Analysis*, contains information about frequency domain analysis, the Fast Fourier transform, and analyzing and processing images in the frequency domain.

# Image Analysis

This chapter contains information about histograms, line profiles, and intensity measurements.

Image analysis combines techniques that compute statistics and measurements based on the gray-level intensities of the image pixels. You can use the image analysis functions to understand the content of the image and to decide which type of inspection tools to use to solve your application. Image analysis functions also provide measurements that you can use to perform basic inspection tasks such as presence or absence verification.

## Histogram

A *histogram* counts and graphs the total number of pixels at each grayscale level. From the graph, you can tell whether the image contains distinct regions of a certain gray-level value.

A histogram provides a general description of the appearance of an image and helps identify various components such as the background, objects, and noise.

## When to Use

The histogram is a fundamental image analysis tool that describes the distribution of the pixel intensities in an image. Use the histogram to determine if the overall intensity in the image is high enough for your inspection task. You can use the histogram to determine whether an image contains distinct regions of certain grayscale values. You can also use a histogram to tune the image acquisition conditions.

You can detect two important criteria by looking at the histogram:

- Saturation—Too little light in the imaging environment leads to underexposure of the imaging sensor, while too much light causes overexposure, or saturation, of the imaging sensor. Images acquired under underexposed or saturated conditions will not contain all the information that you want to inspect from the scene being observed. It is important to detect these imaging conditions and correct for them

during setup of your imaging system. You can detect whether a sensor is underexposed or saturated by looking at the histogram. An underexposed image contains a large number of pixels with low gray-level values. This appears as a peak at the lower end of the histogram. An overexposed or saturated image contains a large number of pixels with very high gray-level values. This condition is represented by a peak at the upper end of the histogram, as shown in Figure 4-1.

- Lack of contrast—A widely-used type of imaging application involves inspecting and measuring (counting) parts of interest in a scene. A strategy to separate the objects from the background relies on a difference in the intensities of both, for example, a bright part and a darker background. In this case, the analysis of the histogram of the image reveals two or more well-separated intensity populations, as shown in Figure 4-2. Tune your imaging setup until the histogram of your acquired images has the contrast required by your application.

## Histogram Concepts

The histogram is the function  $H$  defined on the grayscale range  $[0, \dots, k, \dots, 255]$  such that the number of pixels equal to the gray-level value  $k$  is

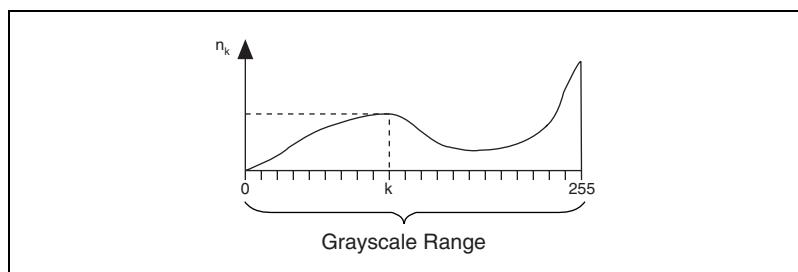
$$H(k) = n_k$$

where  $k$  is the gray-level value,

$n_k$  is the number of pixels in an image with a gray-level value equal to  $k$ , and

$\sum n_k = n$  is the total number of pixels in an image.

The following histogram plot reveals which gray levels occur frequently and which occur rarely.



**Figure 4-1.** Histogram Plot

Two types of histograms can be calculated: the linear and cumulative histograms.

In both cases, the horizontal axis represents the gray-level value that ranges from 0 to 255. For a gray-level value  $k$ , the vertical axis of the linear histogram indicates the number of pixels  $n_k$  set to the value  $k$ , and the vertical axis of the cumulative histogram indicates the percentage of pixels set to a value less than or equal to  $k$ .

## Linear Histogram

The *density function* is

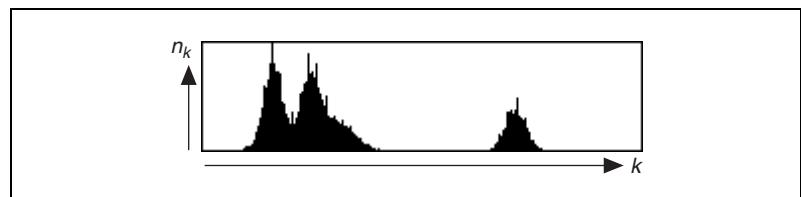
$$H_{Linear}(k) = n_k$$

where  $H_{Linear}(k)$  is the number of pixels equal to  $k$ .

The *probability function* is

$$P_{Linear}(k) = n_k/n$$

where  $P_{Linear}(k)$  is the probability that a pixel is equal to  $k$ .



**Figure 4-2.** Sample of a Linear Histogram

## Cumulative Histogram

The distribution function is

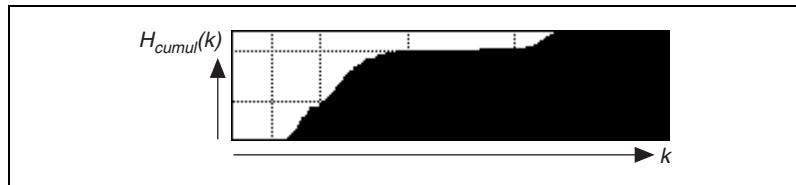
$$H_{Cumul}(k) = \sum_{i=0}^k n_i$$

where  $H_{Cumul}(k)$  is the number of pixels that are less than or equal to  $k$ .

The probability function is

$$P_{Cumul}(k) = \sum_{i=0}^k \frac{n_i}{n}$$

where  $P_{Cumul}(k)$  is the probability that a pixel is less than or equal to  $k$ .



**Figure 4-3.** Sample of a Cumulative Histogram

## Interpretation

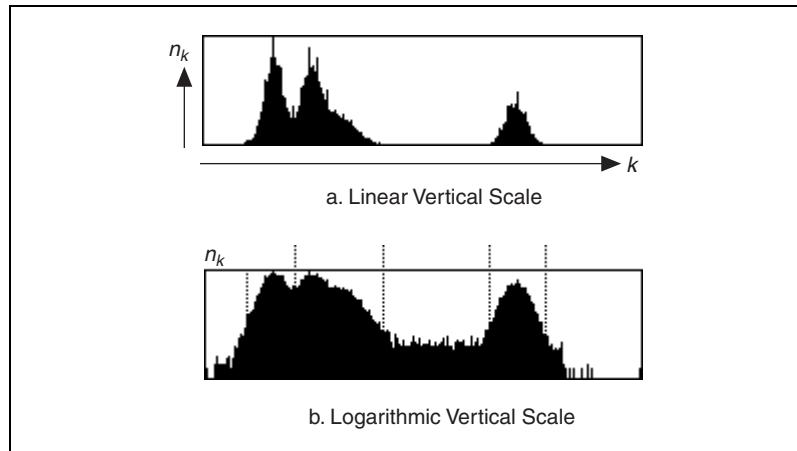
The gray-level intervals featuring a concentrated set of pixels reveal the presence of significant components in the image and their respective intensity ranges.

In Figure 4-2, the linear histogram reveals that the image is composed of three major elements. The cumulative histogram of the same image in Figure 4-3 shows that the two left-most peaks compose approximately 80% of the image, while the remaining 20% corresponds to the third peak.

## Histogram Scale

The vertical axis of a histogram plot can be shown in a linear or logarithmic scale. A logarithmic scale lets you visualize gray-level values used by small numbers of pixels. These values might appear unused when the histogram is displayed in a linear scale.

In a logarithmic scale, the vertical axis of the histogram gives the logarithm of the number of pixels per gray-level value. The use of minor gray-level values becomes more prominent at the expense of the dominant gray-level values. The logarithmic scale emphasizes small histogram values that are not typically noticeable in a linear scale. Figure 4-4 illustrates the difference between the display of the histogram of the same image in a linear and logarithmic scale. In this particular image, three pixels are equal to 0.

**Figure 4-4.** Histogram of the Same Image Using Linear and Logarithmic Vertical Scales

## Histogram of Color Images

The histogram of a color image is expressed as a series of three tables, each corresponding to the histograms of the three primary components in the color model in Table 4-1.

**Table 4-1.** Color Models and Primary Components

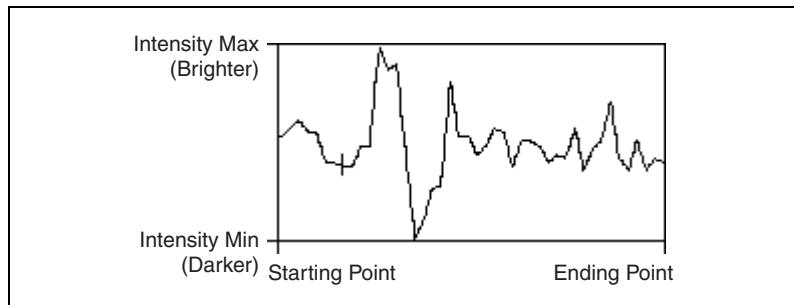
Color Model	Components
RGB	Red, Green, Blue
HSL	Hue, Saturation, Luminance

# Line Profile

A *line profile* plots the variations of intensity along a line. It returns the grayscale values of the pixels along a line and graphs it.

## When to Use

The line profile utility is helpful for examining boundaries between components, quantifying the magnitude of intensity variations, and detecting the presence of repetitive patterns. Figure 4-5 illustrates a typical line profile.



**Figure 4-5.** Line Profile

The peaks and valleys represent increases and decreases of the light intensity along the line selected in the image. Their width and magnitude are proportional to the size and intensity of their related regions.

For example, a bright object with uniform intensity appears in the plot as a plateau. The higher the contrast between an object and its surrounding background, the steeper the slopes of the plateau. Noisy pixels, on the other hand, produce a series of narrow peaks.

# Intensity Measurements

Intensity measurements measure the grayscale image statistics in an image or regions in an image.

## When to Use

You can use intensity measurements to measure the average intensity value in a region of the image to determine, for example, the presence or absence of a part or a defect in a part.

## Concepts

### Densitometry

IMAQ Vision contains the following densitometry parameters:

- Minimum Gray Value—Minimum intensity value in gray-level units
- Maximum Gray Value—Maximum intensity value in gray-level units
- Mean Gray Value—Mean intensity value in the particle expressed in gray-level units
- Standard Deviation—Standard deviation of the intensity values

# Image Processing

This chapter contains information about lookup tables, convolution kernels, spatial filters, and grayscale morphology.

## Lookup Tables

The *lookup table* (LUT) transformations are basic image-processing functions that highlight details in areas containing significant information, at the expense of other areas. These functions include *histogram equalization*, *gamma corrections*, *logarithmic corrections*, and *exponential corrections*.

### When to Use

Use LUT transformations to improve the contrast and brightness of an image by modifying the dynamic intensity of regions with poor contrast.

### LUT Transformation Concepts

A LUT transformation converts input gray-level values (those from the source image) into other gray-level values (in the transformed image).

A LUT transformation applies the transform  $T(x)$  over a specified input range [rangeMin, rangeMax] in the following manner:

$$\begin{aligned} T(x) = \text{dynamicMin} && \text{if } x \leq \text{rangeMin} \\ f(x) && \text{if } \text{rangeMin} < x \leq \text{rangeMax} \\ \text{dynamicMax} && \text{if } x > \text{rangeMax} \end{aligned}$$

$x$  represents the input gray-level value

where dynamicMin = 0 (8-bit images) or the smallest initial pixel value (16-bit and floating point images)

dynamicMax = 255 (8-bit images) or the largest initial pixel value (16-bit and floating point images)

dynamicRange = dynamicMax – dynamicMin

$f(x)$  represents the new value.

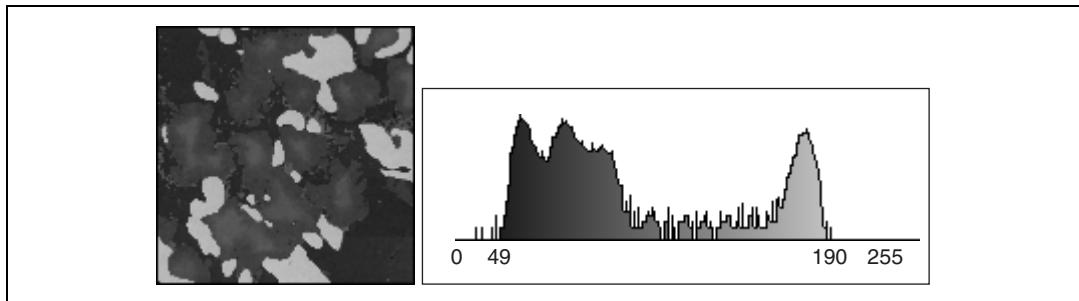
The function scales  $f(x)$  so that  $f(\text{rangeMin}) = \text{dynamicMin}$  and  $f(\text{rangeMax}) = \text{dynamicMax}$ .  $f(x)$  behaves on  $(\text{rangeMin}, \text{rangeMax})$  according to the method you select.

In the case of an 8-bit resolution, a LUT is a table of 256 elements. The index of element of the array represents an input gray-level value. The value of each element indicates the output value.

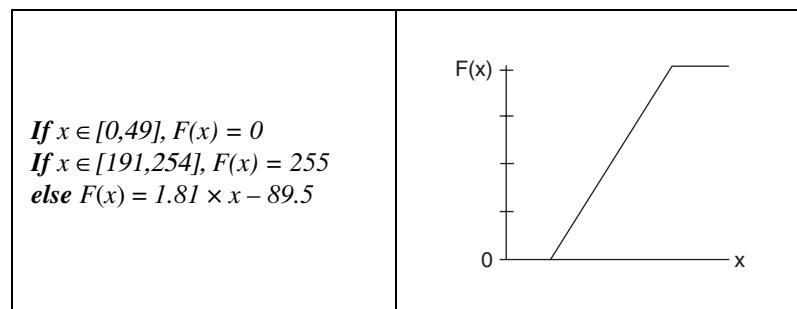
The transfer function associated with a LUT has an intended effect on the brightness and contrast of the image.

## Example

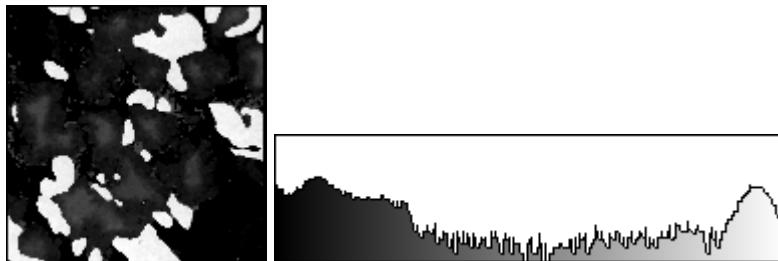
The following example uses the following source image. In the linear histogram of the source image, the gray-level intervals  $[0, 49]$  and  $[191, 254]$  do not contain significant information.



Using the following LUT transformation, any pixel with a value less than 49 is set to 0, and any pixel with a value greater than 191 is set to 255. The interval  $[50, 190]$  expands to  $[1, 254]$ , increasing the intensity dynamic of the regions with a concentration of pixels in the gray-level range  $[50, 190]$ .



The LUT transformation produces the following image. The linear histogram of the new image contains only the two peaks of the interval [50, 190].



## Predefined Lookup Tables

Seven predefined LUTs are available in IMAQ Vision: Linear, Logarithmic, Power 1/Y, Square Root, Exponential, Power Y, and Square. Table 5-1 shows the transfer function for each LUT and describes its effect on an image displayed in a palette that associates dark colors to low-intensity values and bright colors to high-intensity values (such as the Gray palette).

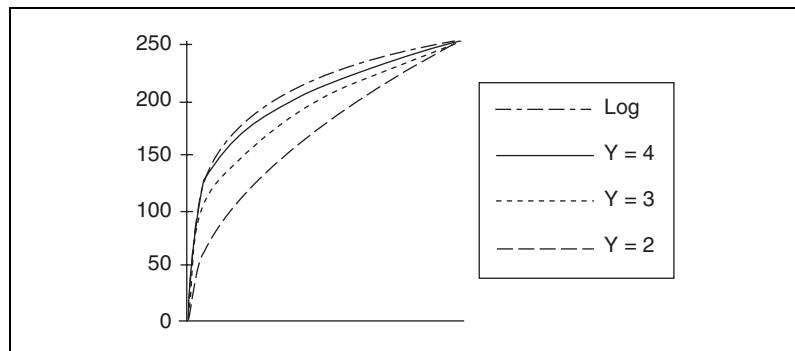
**Table 5-1.** LUT Transfer Functions

LUT	Transfer Function	Shading Correction
Linear		Increases the intensity dynamic by evenly distributing a given gray-level interval [min, max] over the full gray scale [0, 255]. Min and max default values are 0 and 255 for an 8-bit image.
Logarithmic Power 1/Y Square Root		Increases the brightness and contrast in dark regions. Decreases the contrast in bright regions.
Exponential Power Y Square		Decreases the brightness and contrast in dark regions. Increases the contrast in bright regions.

## Logarithmic and Inverse Gamma Correction

The *logarithmic and inverse gamma corrections* expand low gray-level ranges while compressing high gray-level ranges. When using the gray palette, these transformations increase the overall brightness of an image and increase the contrast in dark areas at the expense of the contrast in bright areas.

The following graphs show how the transformations behave. The horizontal axis represents the input gray-level range, and the vertical axis represents the output gray-level range. Each input gray-level value is plotted vertically, and its point of intersection with the look-up curve is plotted horizontally to give an output value.



The *Logarithmic*, *Square Root*, and *Power 1/Y* functions expand intervals containing low gray-level values while compressing intervals containing high gray-level values.

The higher the gamma coefficient  $Y$ , the stronger the intensity correction. The Logarithmic correction has a stronger effect than the Power 1/Y function.

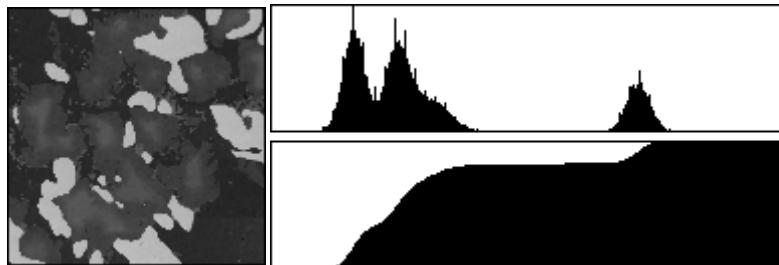
## Logarithmic and Inverse Gamma Correction Examples

The following series of illustrations presents the linear and cumulative histograms of an image after various LUT transformations. The more the histogram is compressed on the right, the brighter the image.

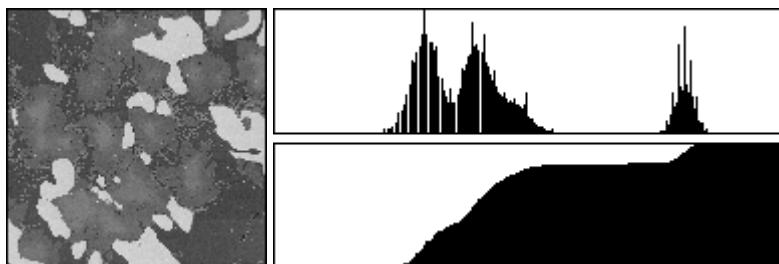


**Note** Graphics on the left represent the original image, graphics on the top right represent the linear histogram, and graphics on the bottom right represent the cumulative histogram.

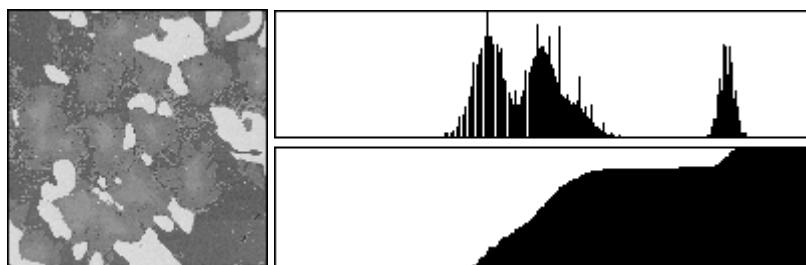
The following graphic shows the original image and histograms.



A Power 1/Y transformation (where  $Y = 1.5$ ) produces the following image and histograms.



A Square Root or Power 1/Y transformation (where  $Y = 2$ ) produces the following image and histograms.



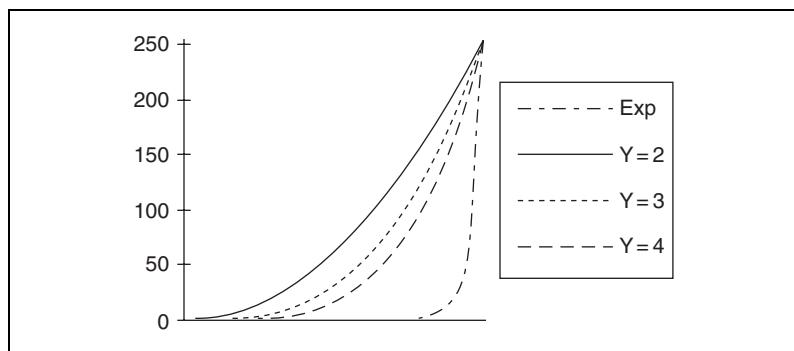
A Logarithm transformation produces the following image and histograms.



## Exponential and Gamma Correction

The *exponential and gamma corrections* expand high gray-level ranges while compressing low gray-level ranges. When using the gray palette, these transformations decrease the overall brightness of an image and increase the contrast in bright areas at the expense of the contrast in dark areas.

The following graphs show how the transformations behave. The horizontal axis represents the input gray-level range, and the vertical axis represents the output gray-level range. Each input gray-level value is plotted vertically, and its point of intersection with the look-up curve then is plotted horizontally to give an output value.



The *Exponential*, *Square*, and *Power Y* functions expand intervals containing high gray-level values while compressing intervals containing low gray-level values.

The higher the gamma coefficient  $Y$ , the stronger the intensity correction. The Exponential correction has a stronger effect than the Power Y function.

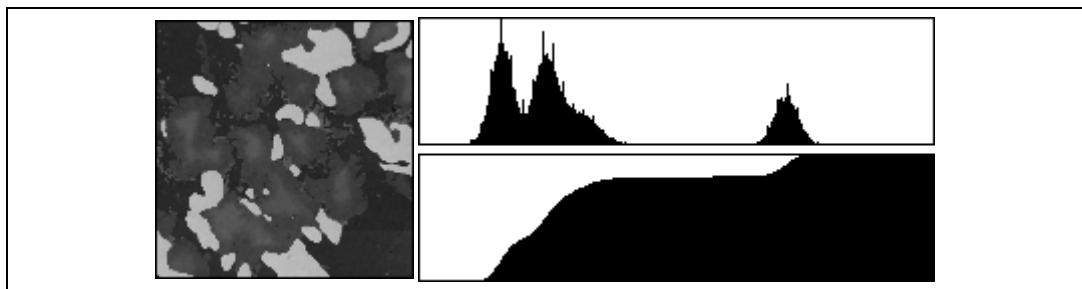
## Exponential and Gamma Correction Examples

The following series of illustrations presents the linear and cumulative histograms of an image after various LUT transformations. The more the histogram is compressed, the darker the image.

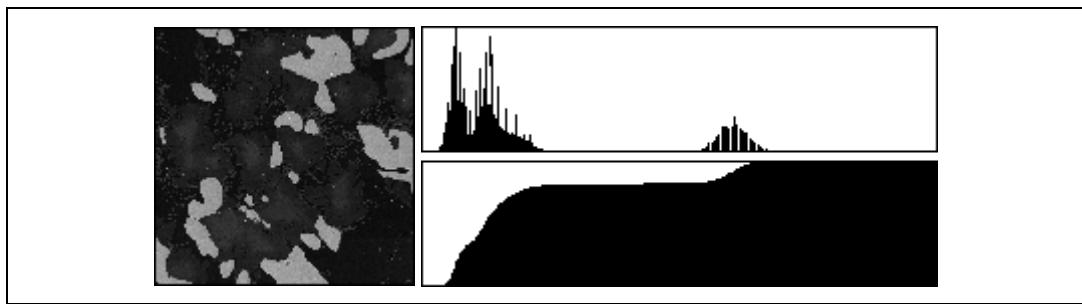


**Note** Graphics on the left represent the original image, graphics on the top right represent the linear histogram, and graphics on the bottom right represent the cumulative histogram.

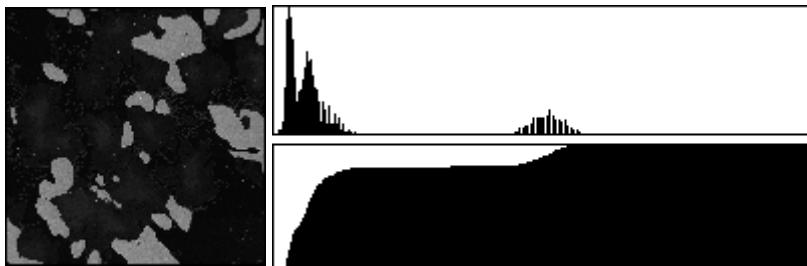
The following graphic shows the original image and histograms.



A Power Y transformation (where  $Y = 1.5$ ) produces the following image and histograms.



A Square or Power Y transformation (where  $Y = 2$ ) produces the following image and histograms.



An Exponential transformation produces the following image and histograms.



## Equalize

The *Equalize* function is a lookup table operation that does not work on a predefined LUT. Instead, the LUT is computed based on the content of the image where the function is applied.

The Equalize function alters the gray-level values of pixels so that they become evenly distributed in the defined grayscale range (0 to 255 for an 8-bit image). The function associates an equal amount of pixels per constant gray-level interval and takes full advantage of the available shades of gray. Use this transformation to increase the contrast in images that do not use all gray levels.

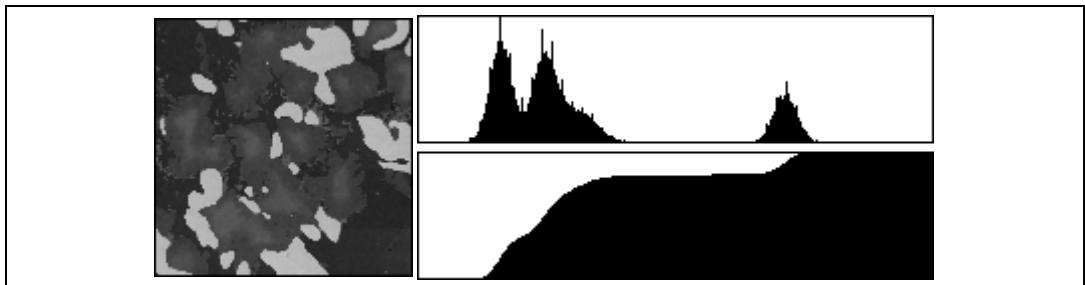
The equalization can be limited to a gray-level interval, also called the equalization range. In this case, the function evenly distributes the pixels belonging to the equalization range over the full interval (0 to 255 for an 8-bit image) and the other pixels are set to 0. The image produced reveals details in the regions that have an intensity in the equalization range; other areas are cleared.

## Equalization Example 1

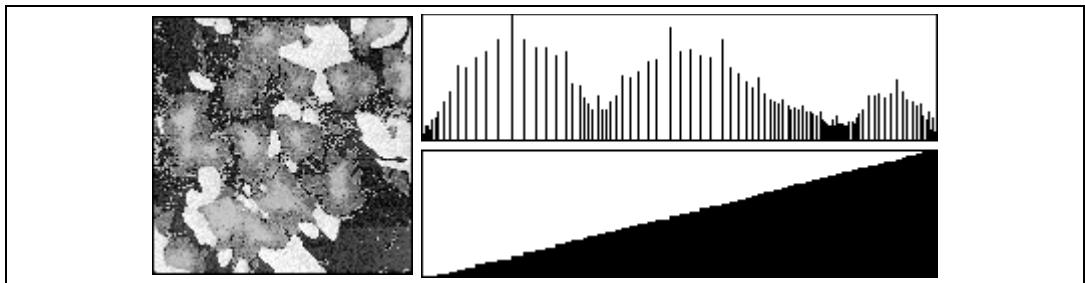
This example shows how an equalization of the interval [0, 255] can spread the information contained in the three original peaks over larger intervals. The transformed image reveals more details about each component in the original image. The following graphics show the original image and histograms.



**Note** In Examples 1 and 2, graphics on the left represent the original image, graphics on the top right represent the linear histogram, and graphics on the bottom right represent the cumulative histogram.



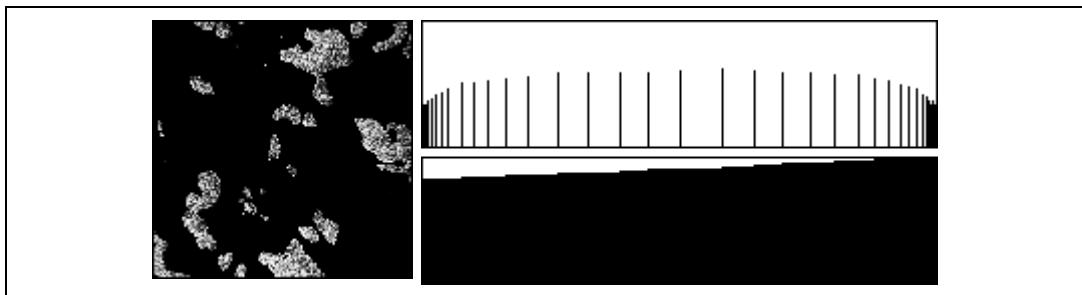
An equalization from [0, 255] to [0, 255] produces the following image and histograms.



**Note** The cumulative histogram of an image after a histogram equalization always has a linear profile, as seen in the preceding example.

## Equalization Example 2

This example shows how an equalization of the interval [166, 200] can spread the information contained in the original third peak (ranging from 166 to 200) to the interval [0, 255]. The transformed image reveals details about the component with the original intensity range [166, 200] while all other components are set to black. An equalization from [166, 200] to [0, 255] produces the following image and histograms.



## Convolution Kernels

A convolution kernel defines a two-dimensional filter that you can apply to a grayscale image. A convolution kernel is a two-dimensional structure whose coefficients define the characteristics of the convolution filter that it represents. In a typical filtering operation, the coefficients of the convolution kernel determine the filtered value of each pixel in the image. IMAQ Vision provides a set of convolution kernels that you can use to perform different types of filtering operations on an image. You also can define your own convolution kernels, thus creating custom filters.

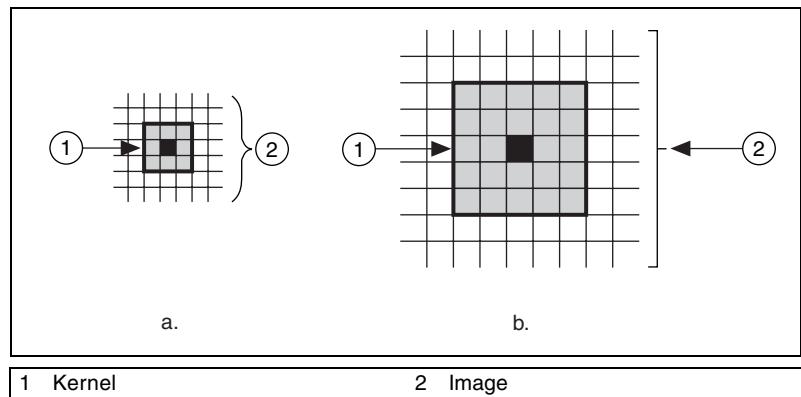
Use a convolution kernel whenever you want to filter a grayscale image. Filtering a grayscale image enhances the quality of the image to meet the requirements of your application. Use filters to smooth an image, remove noise from an image, enhance the edge information in an image, and so on.

## Concepts

A convolution kernel defines how a filter alters the pixel values in a grayscale image. The convolution kernel is a two-dimensional structure whose coefficients define how the filtered value at each pixel is computed. The filtered value of a pixel is a weighted combination of its original value and the values of its neighboring pixels. The convolution kernel coefficients define the contribution of each neighboring pixel to the pixel being updated. The convolution kernel size determines the number of

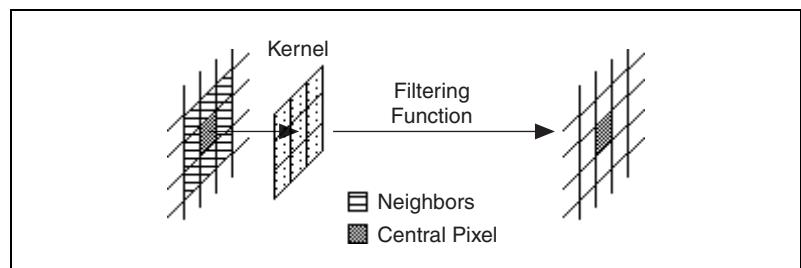
neighboring pixels whose values are considered during the filtering process.

In the case of a  $3 \times 3$  kernel, illustrated in Figure 5-1a, the value of the central pixel (shown in black) is derived from the values of its eight surrounding neighbors (shown in gray). A  $5 \times 5$  kernel, shown in Figure 5-1b, specifies 24 neighbors, a  $7 \times 7$  kernel specifies 48 neighbors, and so forth.



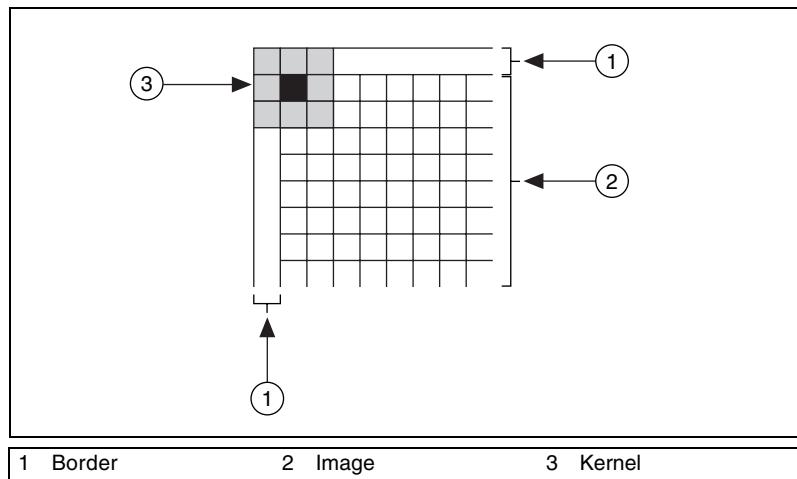
**Figure 5-1.** Examples of Kernels

A filtering operation on an image involves moving the kernel from the leftmost and topmost pixel in the image to the rightmost and bottommost point in the image. At each pixel in the image, the new value is computed using the values that lie under the kernel, as shown in Figure 5-2.



**Figure 5-2.** Mechanics of Filtering

When computing the filtered values of the pixels that lie along the border of the image (the first row, last row, first column, or last column of pixels), part of the kernel falls outside the image. For example, Figure 5-3 shows that one row and one column of a  $3 \times 3$  kernel fall outside the image when computing the value of the topmost leftmost pixel.



**Figure 5-3.** Filtering Border Pixels

IMAQ Vision automatically allocates a border region when you create an image. The default border region is three pixels deep and contains pixel values of 0. You can also define a custom border region and specify the pixel values within the region. The size of the border region should be greater than or equal to half the number of rows or columns in your kernel. The filtering results from along the border of an image are unreliable because the neighbors necessary to compute these values are missing, therefore decreasing the efficiency of the filter, which works on a much smaller number of pixels than specified for the rest of the image. For more information about border regions, see Chapter 1, *Digital Images*, of the *IMAQ Vision Concepts Manual*.

# Spatial Filtering

---

Filters are divided into two types: linear (also called *convolution*) and nonlinear.

A convolution is an algorithm that consists of recalculating the value of a pixel based on its own pixel value and the pixel values of its neighbors weighted by the coefficients of a convolution kernel. The sum of this calculation is divided by the sum of the elements in the kernel to obtain a new pixel value. The size of the *convolution kernel* does not have a theoretical limit and can be either square or rectangular ( $3 \times 3$ ,  $5 \times 5$ ,  $5 \times 7$ ,  $9 \times 3$ ,  $127 \times 127$ , and so on). Convolutions are divided into four families: gradient, Laplacian, smoothing, and Gaussian. This grouping is determined by the convolution kernel contents or the weight assigned to each pixel, which depends on the geographical position of that pixel in relation to the central kernel pixel.

IMAQ Vision features a set of standard convolution kernels for each family and for the usual sizes ( $3 \times 3$ ,  $5 \times 5$ , and  $7 \times 7$ ). You also can create your own kernels and choose what to put into them. The size of the user-defined kernel is virtually unlimited. With this capability, you can create filters with specific characteristics.

## When to Use

Spatial filters serve a variety of purposes, such as detecting edges along a specific direction, contouring patterns, reducing noise, and detail outlining or smoothing. Filters smooth, sharpen, transform, and remove noise from an image so that you can extract the information you need.

The purpose of the nonlinear filters is to either extract the contours (edge detection) or remove the isolated pixels. IMAQ Vision has six different methods you can use for contour extraction (Differentiation, Gradient, Prewitt, Roberts, Sigma, or Sobel). The Canny Edge Detection filter is a specialized edge detection method that locates edges accurately, even under low signal-to-noise conditions in an image.

To harmonize pixel values, choose between two filters, each of which uses a different method: NthOrder and LowPass. These functions require that either a kernel size and order number or percentage is specified on input.

*Spatial filters* alter pixel values with respect to variations in light intensity in their neighborhood. The neighborhood of a pixel is defined by the size of a matrix, or mask, centered on the pixel itself. These filters can be sensitive to the presence or absence of light-intensity variations.

Spatial filters fall into two categories:

- *Highpass filters* emphasize significant variations of the light intensity usually found at the boundary of objects. Highpass frequency filters help isolate abruptly varying patterns that correspond to sharp edges, details, and noise.
- *Lowpass filters* attenuate variations of the light intensity. Lowpass frequency filters help emphasize gradually varying patterns such as objects and the background. They have the tendency to smooth images by eliminating details and blurring edges.

## Spatial Filtering Concepts

### Spatial Filter Classification Summary

Table 5-2 describes the different types of spatial filters.

**Table 5-2.** Spatial Filter Classifications

Filter Type	Filters	
Linear	Highpass	Gradient, Laplacian
	Lowpass	Smoothing, Gaussian
	Nonlinear Filters	
Nonlinear Filters	Highpass	Gradient, Roberts, Sobel, Prewitt, Differentiation, Sigma
	Lowpass	Median, Nth Order, Lowpass

## Linear Filters

A linear filter replaces each pixel by a weighted sum of its neighbors. The matrix defining the neighborhood of the pixel also specifies the weight assigned to each neighbor. This matrix is called the *convolution kernel*.

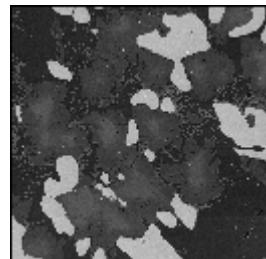
If the filter kernel contains both negative and positive coefficients, the transfer function is equivalent to a weighted differentiation and produces a sharpening or highpass filter. Typical highpass filters include gradient and Laplacian filters.

If all coefficients in the kernel are positive, the transfer function is equivalent to a weighted summation and produces a smoothing or lowpass filter. Typical lowpass filters include smoothing and Gaussian filters.

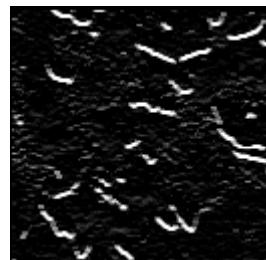
### Gradient Filter

A *gradient filter* highlights the variations of light intensity along a specific direction, which has the effect of outlining edges and revealing texture.

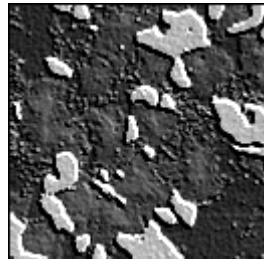
Given the following source image.



A gradient filter extracts horizontal edges to produce the following image.



A gradient filter highlights diagonal edges to produce the following image.



### Kernel Definition

A *gradient convolution filter* is a first-order derivative. Its kernel uses the following model:

$$\begin{matrix} a & -b & c \\ b & x & -d \\ c & d & -a \end{matrix}$$

where  $a$ ,  $b$ ,  $c$  and  $d$  are integers and  $x = 0$  or  $1$ .

### Filter Axis and Direction

This kernel has an axis of symmetry that runs between the positive and negative coefficients of the kernel and through the central element. This axis of symmetry gives the orientation of the edges to outline. For example:

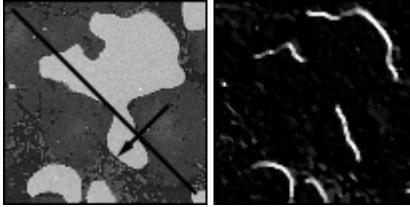
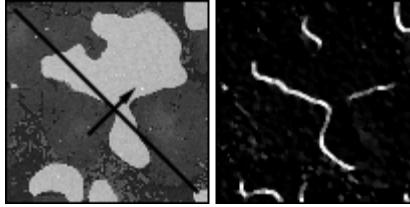
If  $a = 0$ ,  $b = -1$ ,  $c = -1$ ,  $d = -1$ , and  $x = 0$ , the kernel is the following:

$$\begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix}$$

The axis of symmetry is located at  $135^\circ$ .

For a given direction, you can design a gradient filter to highlight or darken the edges along that direction. The filter actually is sensitive to the variations of intensity perpendicular to the axis of symmetry of its kernel. Given the direction  $D$  going from the negative coefficients of the kernel towards the positive coefficients, the filter highlights the pixels where the light intensity increases along the direction  $D$ , and darkens the pixels where the light intensity decreases.

The following two kernels emphasize edges oriented at 135°.

Gradient #1	Gradient #2
$\begin{matrix} 0 & -1 & -1 \\ 1 & \mathbf{0} & -1 \\ 1 & 1 & 0 \end{matrix}$ <p>Gradient #1 highlights pixels where the light intensity increases along the direction going from northeast to southwest. It darkens pixels where the light intensity decreases along that same direction. This processing outlines the northeast front edges of bright regions such as the ones in the illustration.</p> 	$\begin{matrix} 0 & 1 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & -1 & 0 \end{matrix}$ <p>Gradient #2 highlights pixels where the light intensity increases along the direction going from southwest to northeast. It darkens pixels where the light intensity decreases along that same direction. This processing outlines the southwest front edges of bright regions such as the ones in the illustration.</p> 

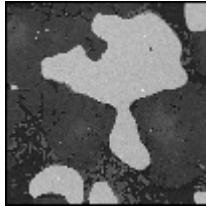


**Note** Applying Gradient #1 to an image returns the same results as applying Gradient #2 to its photometric negative, because reversing the lookup table of an image converts bright regions into dark regions and vice versa.

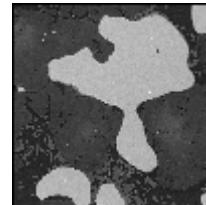
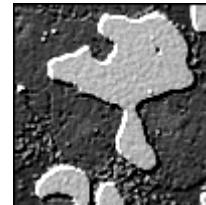
### Edge Extraction and Edge Highlighting

The gradient filter has two effects, depending on whether the central coefficient  $x$  is equal to 1 or 0:

- If the central coefficient is null ( $x = 0$ ), the gradient filter highlights the pixels where variations of light intensity occur along a direction specified by the configuration of the coefficients  $a$ ,  $b$ ,  $c$ , and  $d$ . The transformed image contains black-white borders at the original edges, and the shades of the overall patterns are darkened.

Source Image	Gradient #1	Filtered Image
	<b>Gradient #1</b> $\begin{matrix} -1 & -1 & 0 \\ -1 & \mathbf{0} & 1 \\ 0 & 1 & 1 \end{matrix}$	

- If the central coefficient is equal to 1 ( $x = 1$ ), the gradient filter detects the same variations as mentioned above, but superimposes them over the source image. The transformed image looks like the source image with edges highlighted. Use this type of kernel for grain extraction and perception of texture.

Source Image	Gradient #2	Filtered Image
	<b>Gradient #2</b> $\begin{matrix} -1 & -1 & 0 \\ -1 & \mathbf{1} & 1 \\ 0 & 1 & 1 \end{matrix}$	

Notice that the kernel Gradient #2 can be decomposed as follows:

$$\begin{matrix} -1 & -1 & 0 \\ -1 & \mathbf{1} & 1 \\ 0 & 1 & 1 \end{matrix} = \begin{matrix} -1 & -1 & 0 \\ -1 & \mathbf{0} & 1 \\ 0 & 1 & 1 \end{matrix} + \begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$



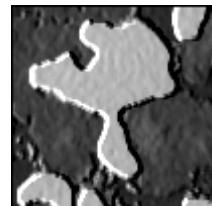
**Note** The convolution filter using the second kernel on the right side of the equation reproduces the source image. All neighboring pixels are multiplied by 0 and the central pixel remains equal to itself:  $(P_{(i,j)} = 1 \times P_{(i,j)})$ .

This equation indicates that Gradient #2 adds the edges extracted by the Gradient #1 to the source image.

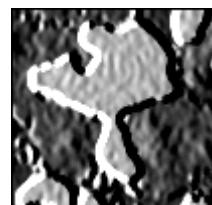
$$\text{Gradient } \#2 = \text{Gradient } \#1 + \text{Source Image}$$

### Edge Thickness

The larger the kernel, the thicker the edges. The following image illustrates gradient west–east  $3 \times 3$ .



The following image illustrates gradient west–east  $5 \times 5$ .



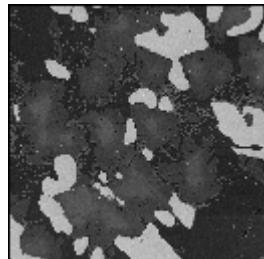
Finally, the following image illustrates gradient west–east  $7 \times 7$ .



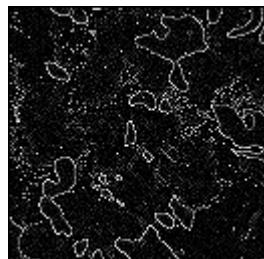
## Laplacian Filters

A [Laplacian filter](#) highlights the variation of the light intensity surrounding a pixel. The filter extracts the contour of objects and outlines details. Unlike the gradient filter, it is omnidirectional.

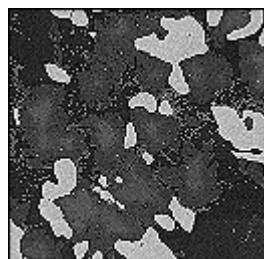
Given the following source image.



A Laplacian filter extracts contours to produce the following image.



A Laplacian filter highlights contours to produce the following image.



## Kernel Definition

The *Laplacian convolution filter* is a second-order derivative, and its kernel uses the following model:

$$\begin{matrix} a & d & c \\ b & x & b \\ c & d & a \end{matrix}$$

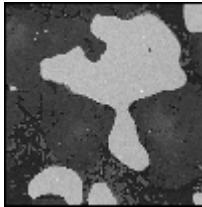
where  $a$ ,  $b$ ,  $c$ , and  $d$  are integers.

The Laplacian filter has two different effects, depending on whether the central coefficient  $x$  is equal to or greater than the sum of the absolute values of the outer coefficients.

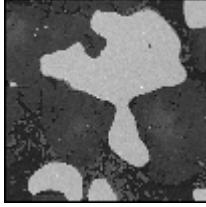
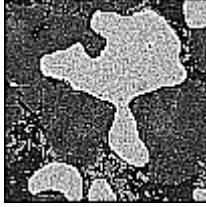
## Contour Extraction and Highlighting

If the central coefficient is equal to this sum  $x = 2(|a| + |b| + |c| + |d|)$ , the Laplacian filter extracts the pixels where significant variations of light intensity are found. The presence of sharp edges, boundaries between objects, modification in the texture of a background, noise, or other effects can cause these variations. The transformed image contains white contours on a black background.

Notice the following source image, Laplacian kernel, and filtered image.

Source Image	Laplacian #1	Filtered Image
	$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$	

If the central coefficient is greater than the sum of the outer coefficients ( $x > 2(a + b + c + d)$ ), the Laplacian filter detects the same variations as mentioned above, but superimposes them over the source image. The transformed image looks like the source image, with all significant variations of the light intensity highlighted.

Source Image	Laplacian #2	Filtered Image
	$\begin{array}{ccc} -1 & -1 & -1 \\ -1 & \mathbf{9} & -1 \\ -1 & -1 & -1 \end{array}$	

Notice that the Laplacian #2 kernel can be decomposed as follows:

$$\begin{array}{ccc} -1 & -1 & -1 \\ -1 & \mathbf{9} & -1 \\ -1 & -1 & -1 \end{array} = \begin{array}{ccc} -1 & -1 & -1 \\ -1 & \mathbf{8} & -1 \\ -1 & -1 & -1 \end{array} + \begin{array}{ccc} 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 \end{array}$$



**Note** The convolution filter, using the second kernel on the right side of the equation, reproduces the source image. All neighboring pixels are multiplied by 0, and the central pixel remains equal to itself: ( $P_{(i,j)} = 1 \times P_{(i,j)}$ ).

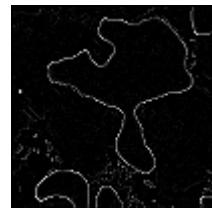
This equation indicates that the Laplacian #2 kernel adds the contours extracted by the Laplacian #1 kernel to the source image.

$$\text{Laplacian #2} = \text{Laplacian #1} + \text{Source Image}$$

For example, if the central coefficient of Laplacian #2 kernel is 10, the Laplacian filter adds the contours extracted by Laplacian #1 kernel to the source image times 2, and so forth. A greater central coefficient corresponds to less-prominent contours and details highlighted by the filter.

### Contour Thickness

Larger kernels correspond to thicker contours. The following image is a Laplacian  $3 \times 3$ .



The following image is a Laplacian  $5 \times 5$ .



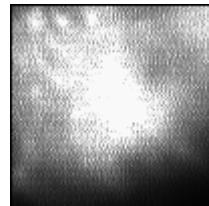
The following image is a Laplacian  $7 \times 7$ .



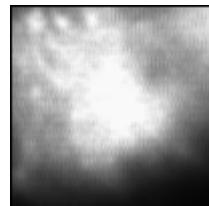
## Smoothing Filter

A *smoothing filter* attenuates the variations of light intensity in the neighborhood of a pixel. It smooths the overall shape of objects, blurs edges, and removes details.

Given the following source image.



A smoothing filter produces the following image.



### Kernel Definition

A *smoothing convolution filter* is an averaging filter whose kernel uses the following model:

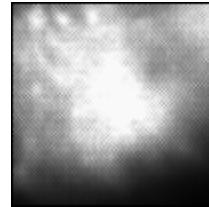
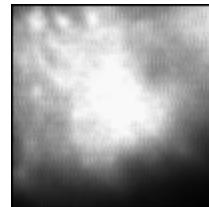
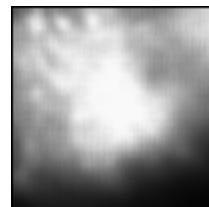
$$\begin{matrix} a & d & c \\ b & x & b \\ c & d & a \end{matrix}$$

where  $a, b, c$ , and  $d$  are positive integers, and  $x = 0$  or  $1$ .

Because all the coefficients in a smoothing kernel are positive, each central pixel becomes a weighted average of its neighbors. The stronger the weight of a neighboring pixel, the more influence it has on the new value of the central pixel.

For a given set of coefficients  $(a, b, c, d)$ , a smoothing kernel with a central coefficient equal to  $0$  ( $x = 0$ ) has a stronger blurring effect than a smoothing kernel with a central coefficient equal to  $1$  ( $x = 1$ ).

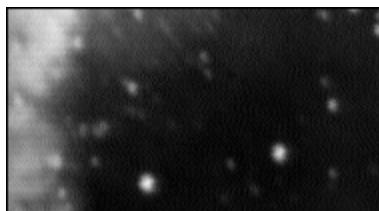
Notice the following smoothing kernels and filtered images. A larger kernel size corresponds to a stronger smoothing effect.

<b>Kernel #1</b> $\begin{matrix} 0 & 1 & 0 \\ 1 & \mathbf{0} & 1 \\ 0 & 1 & 0 \end{matrix}$	<b>Filtered Image</b> 
<b>Kernel #2</b> $\begin{matrix} 2 & 2 & 2 \\ 2 & \mathbf{1} & 2 \\ 2 & 2 & 2 \end{matrix}$	<b>Filtered Image</b> 
<b>Kernel #3</b> $\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$	<b>Filtered Image</b> 
<b>Kernel #4</b> $\begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{matrix}$	<b>Filtered Image</b> 

## Gaussian Filters

A *Gaussian filter* attenuates the variations of light intensity in the neighborhood of a pixel. It smooths the overall shape of objects and attenuates details. It is similar to a smoothing filter, but its blurring effect is more subdued.

Given the following source image.



A Gaussian filter produces the following image.



### Kernel Definition

A *Gaussian convolution filter* is an averaging filter, and its kernel uses the following model:

$$\begin{matrix} a & d & c \\ b & \mathbf{x} & b \\ c & d & a \end{matrix}$$

where  $a$ ,  $b$ ,  $c$ , and  $d$  are integers, and  $x > 1$ .

The coefficients of a Gaussian convolution kernel of a given size are the best possible approximation using integer numbers of a Gaussian curve. For example:

**3 x 3**

1	2	1
2	<b>4</b>	2
1	2	1

**5 x 5**

1	2	4	2	1
2	4	8	4	2
4	8	16	8	4
2	4	8	4	2
1	2	4	2	1

Because all the coefficients in a Gaussian kernel are positive, each pixel becomes a weighted average of its neighbors. The stronger the weight of a neighboring pixel, the more influence it has on the new value of the central pixel.

Unlike a smoothing kernel, the central coefficient of a Gaussian filter is greater than 1. Therefore the original value of a pixel is multiplied by a weight greater than the weight of any of its neighbors. As a result, a greater central coefficient corresponds to a more subtle smoothing effect. A larger kernel size corresponds to a stronger smoothing effect.

## Nonlinear Filters

A *nonlinear filter* replaces each pixel value with a nonlinear function of its surrounding pixels. Like the linear filters, the nonlinear filters operate on a neighborhood.

### Nonlinear Prewitt Filter

The *nonlinear Prewitt filter* is a highpass filter that extracts the outer contours of objects. It highlights significant variations of the light intensity along the vertical and horizontal axes.

Each pixel is assigned the maximum value of its horizontal and vertical gradient obtained with the following Prewitt convolution kernels:

**Kernel #1**

-1	0	1
-1	<b>0</b>	1
-1	0	1

**Kernel #2**

-1	-1	-1
0	<b>0</b>	0
1	1	1

## Nonlinear Sobel Filter

The *nonlinear Sobel filter* is a highpass filter that extracts the outer contours of objects. It highlights significant variations of the light intensity along the vertical and horizontal axes.

Each pixel is assigned the maximum value of its horizontal and vertical gradient obtained with the following Sobel convolution kernels:

**Kernel #1**

$$\begin{matrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{matrix}$$

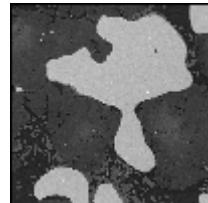
**Kernel #2**

$$\begin{matrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{matrix}$$

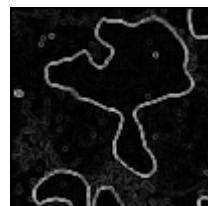
As opposed to the Prewitt filter, the Sobel filter assigns a higher weight to the horizontal and vertical neighbors of the central pixel.

## Nonlinear Prewitt and Nonlinear Sobel Example

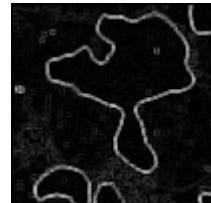
This example uses the following source image.



A nonlinear Prewitt filter produces the following image.



A nonlinear Sobel filter produces the following image.



Both filters outline the contours of the objects. Because of the different convolution kernels they combine, the nonlinear Prewitt has the tendency to outline curved contours while the nonlinear Sobel extracts square contours. This difference is noticeable when observing the outlines of isolated pixels.

### **Nonlinear Gradient Filter**

The *nonlinear gradient filter* outlines contours where an intensity variation occurs along the vertical axis.

### **Roberts Filter**

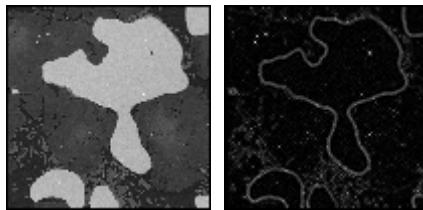
The *Roberts filter* outlines the contours that highlight pixels where an intensity variation occurs along the diagonal axes.

### **Differentiation Filter**

The *differentiation filter* produces continuous contours by highlighting each pixel where an intensity variation occurs between itself and its three upper-left neighbors.

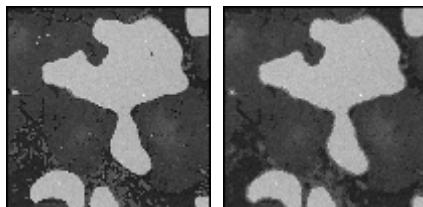
## Sigma Filter

The *Sigma filter* is a highpass filter. It outlines contours and details by setting pixels to the mean value found in their neighborhood, if their deviation from this value is not significant. The example on the left shows an image before filtering. The example on the right shows the image after filtering.



## Lowpass Filter

The *lowpass filter* reduces details and blurs edges by setting pixels to the mean value found in their neighborhood, if their deviation from this value is large. The example on the left shows an image before filtering. The example on the right shows the image after filtering.



## Median Filter

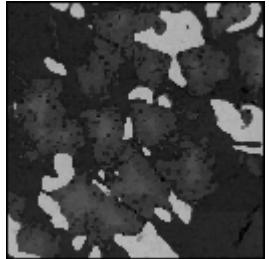
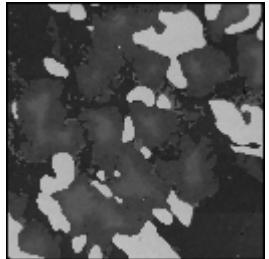
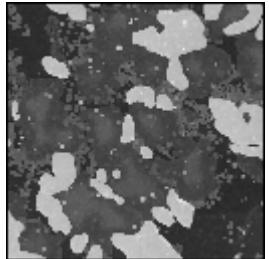
The *median filter* is a lowpass filter. It assigns to each pixel the median value of its neighborhood, effectively removing isolated pixels and reducing detail. However, the median filter does not blur the contour of objects.

You can implement the median filter by performing an Nth order filter and setting the order to  $(f^2 - 1) / 2$  for a given filter size of  $f \times f$ .

## Nth Order Filter

The *Nth order filter* is an extension of the median filter. It assigns to each pixel the  $N$ th value of its neighborhood (when sorted in increasing order). The value  $N$  specifies the order of the filter, which you can use to moderate the effect of the filter on the overall light intensity of the image. A lower order corresponds to a darker transformed image; a higher order corresponds to a brighter transformed image.

To see the effect of the  $N$ th order filter, notice the example of an image with bright objects and a dark background. When viewing this image with the gray palette, the objects have higher gray-level values than the background.

For a Given Filter Size $f \times f$	Example of a Filter Size $3 \times 3$
<ul style="list-style-type: none"> <li>If <math>N &lt; (f^2 - 1)/2</math>, the <math>N</math>th order filter has the tendency to erode bright regions (or dilate dark regions).</li> <li>If <math>N = 0</math>, each pixel is replaced by its local minimum.</li> </ul>	Order 0 (smooths image, erodes bright objects) 
<ul style="list-style-type: none"> <li>If <math>N = (f^2 - 1)/2</math>, each pixel is replaced by its local median value. Dark pixels isolated in objects are removed, as well as bright pixels isolated in the background. The overall area of the background and object regions does not change.</li> </ul>	Order 4 (equivalent to a median filter) 
<ul style="list-style-type: none"> <li>If <math>N &gt; (f^2 - 1)/2</math>, the <math>N</math>th order filter has the tendency to dilate bright regions (or erode dark regions).</li> <li>If <math>N = f^2 - 1</math>, each pixel is replaced by its local maximum.</li> </ul>	Order 8 (smooths image, dilates bright objects) 

## In-Depth Discussion

If  $P_{(i,j)}$  represents the intensity of the pixel  $P$  with the coordinates  $(i,j)$ , the pixels surrounding  $P_{(i,j)}$  can be indexed as follows (in the case of a  $3 \times 3$  matrix):

$P_{(i-1, j-1)}$	$P_{(i, j-1)}$	$P_{(i+1, j-1)}$
$P_{(i-1, j)}$	$P_{(i, j)}$	$P_{(i+1, j)}$
$P_{(i-1, j+1)}$	$P_{(i, j+1)}$	$P_{(i+1, j+1)}$

A *linear filter* assigns to  $P_{(i,j)}$  a value that is a linear combination of its surrounding values. For example:

$$P_{(i,j)} = P_{(i,j-1)} + P_{(i-1,j)} + 2P_{(i,j)} + P_{(i+1,j)} + P_{(i,j+1)}$$

A *nonlinear filter* assigns to  $P_{(i,j)}$  a value that is not a linear combination of the surrounding values. For example:

$$P_{(i,j)} = \max(P_{(i-1,j-1)}, P_{(i+1,j-1)}, P_{(i-1,j+1)}, P_{(i+1,j+1)})$$

In the case of a  $5 \times 5$  neighborhood, the  $i$  and  $j$  indexes vary from -2 to 2. The series of pixels that includes  $P_{(i,j)}$  and its surrounding pixels is annotated as  $P_{(n,m)}$ .

## Linear Filters

For each pixel  $P_{(i,j)}$  in an image (where  $i$  and  $j$  represent the coordinates of the pixel), the convolution kernel is centered on  $P_{(i,j)}$ . Each pixel masked by the kernel is multiplied by the coefficient placed on top of it.  $P_{(i,j)}$  becomes the sum of these products divided by the sum of the coefficient or 1 (whichever is greater).

In the case of a  $3 \times 3$  neighborhood, the pixels surrounding  $P_{(i,j)}$  and the coefficients of the kernel,  $K$ , can be indexed as follows:

$P_{(i-1, j-1)}$	$P_{(i, j-1)}$	$P_{(i+1, j-1)}$
$P_{(i-1, j)}$	$P_{(i, j)}$	$P_{(i+1, j)}$
$P_{(i-1, j+1)}$	$P_{(i, j+1)}$	$P_{(i+1, j+1)}$

$K_{(i-1, j-1)}$	$K_{(i, j-1)}$	$K_{(i+1, j-1)}$
$K_{(i-1, j)}$	$K_{(i, j)}$	$K_{(i+1, j)}$
$K_{(i-1, j+1)}$	$K_{(i, j+1)}$	$K_{(i+1, j+1)}$

The pixel  $P_{(i,j)}$  is given the value  $(1/N)\sum K_{(a,b)}P_{(a,b)}$ , with  $a$  ranging from  $(i-1)$  to  $(i+1)$ , and  $b$  ranging from  $(j-1)$  to  $(j+1)$ .  $N$  is the *normalization factor*, equal to  $\sum K_{(a,b)}$  or 1, whichever is greater.

If the new value  $P_{(i,j)}$  is negative, it is set to 0. If the new value  $P_{(i,j)}$  is greater than 255, it is set to 255 (in the case of 8-bit resolution).

The greater the absolute value of a coefficient  $K_{(a,b)}$ , the more the pixel  $P_{(a,b)}$  contributes to the new value of  $P_{(i,j)}$ . If a coefficient  $K_{(a,b)}$  is 0, the neighbor  $P_{(a,b)}$  does not contribute to the new value of  $P_{(i,j)}$  (notice that  $P_{(a,b)}$  might be  $P_{(i,j)}$  itself).

If the convolution kernel is

$$\begin{matrix} 0 & 0 & 0 \\ -2 & \mathbf{1} & 2 \\ 0 & 0 & 0 \end{matrix}$$

then

$$P_{(i,j)} = (-2P_{(i-1,j)} + P_{(i,j)} + 2P_{(i+1,j)})$$

If the convolution kernel is

$$\begin{matrix} 0 & 1 & 0 \\ 1 & \mathbf{0} & 1 \\ 0 & 1 & 0 \end{matrix}$$

then

$$P_{(i,j)} = (P_{(i,j-1)} + P_{(i-1,j)} + P_{(i+1,j)} + P_{(i,j+1)})$$

## Nonlinear Prewitt Filter

$$P_{(i,j)} = \max[|P_{(i+1,j-1)} - P_{(i-1,j-1)} + P_{(i+1,j)} - P_{(i-1,j)} + P_{(i+1,j+1)} - P_{(i-1,j+1)}|, |P_{(i-1,j+1)} - P_{(i-1,j-1)} + P_{(i,j+1)} - P_{(i,j-1)} + P_{(i+1,j+1)} - P_{(i+1,j-1)}|]$$

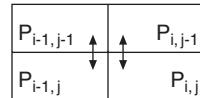
## Nonlinear Sobel Filter

$$P_{(i,j)} = \max[|P_{(i+1,j-1)} - P_{(i-1,j-1)} + 2P_{(i+1,j)} - 2P_{(i-1,j)} + P_{(i+1,j+1)} - P_{(i-1,j+1)}|, |P_{(i-1,j+1)} - P_{(i-1,j-1)} + 2P_{(i,j+1)} - 2P_{(i,j-1)} + P_{(i+1,j+1)} - P_{(i+1,j-1)}|]$$

## Nonlinear Gradient Filter

The new value of a pixel becomes the maximum absolute value between its deviation from the upper neighbor and the deviation of its two left neighbors.

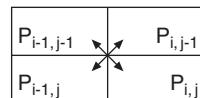
$$P_{(i,j)} = \max[|P_{(i,j-1)} - P_{(i,j)}|, |P_{(i-1,j-1)} - P_{(i-1,j)}|]$$



## Roberts Filter

The new value of a pixel becomes the maximum absolute value between the deviation of its upper-left neighbor and the deviation of its two other neighbors.

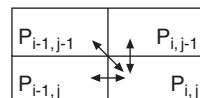
$$P_{(i,j)} = \max[|P_{(i-1,j-1)} - P_{(i,j)}|, |P_{(i,j-1)} - P_{(i-1,j)}|]$$



## Differentiation Filter

The new value of a pixel becomes the absolute value of its maximum deviation from its upper-left neighbors.

$$P_{(i,j)} = \max[|P_{(i-1,j)} - P_{(i,j)}|, |P_{(i-1,j-1)} - P_{(i,j)}|, |P_{(i,j-1)} - P_{(i,j)}|]$$



## Sigma Filter

*If*  $P_{(i,j)} - M > S$ ,  
*then*  $P_{(i,j)} = P_{(i,j)}$ ,  
*else*  $P_{(i,j)} = M$ .

Given  $M$ , the mean value of  $P_{(i,j)}$  and its neighbors and  $S$ , their standard deviation, each pixel  $P_{(i,j)}$  is set to the mean value  $M$  if it falls inside the range  $[M - S, M + S]$ .

## Lowpass Filter

*If*  $P_{(i,j)} - M < S$ ,  
*then*  $P_{(i,j)} = P_{(i,j)}$ ,  
*else*  $P_{(i,j)} = M$ .

Given  $M$ , the mean value of  $P_{(i,j)}$  and its neighbors and  $S$ , their standard deviation, each pixel  $P_{(i,j)}$  is set to the mean value  $M$  if it falls outside the range  $[M - S, M + S]$ .

## Median Filter

$P_{(i,j)}$  = median value of the series  $[P_{(n,m)}]$

## Nth Order Filter

$P_{(i,j)}$  = Nth value in the series  $[P_{(n,m)}]$

where the  $P_{(n,m)}$  are sorted in increasing order.

The following example uses a  $3 \times 3$  neighborhood:

13	10	9
12	4	8
5	5	6

The following table shows the new output value of the central pixel for each  $N$ th order value:

$N$ th Order	0	1	2	3	4	5	6	7	8
New Pixel Value	4	5	5	6	8	9	10	12	13

Notice that for a given filter size  $f$ , the  $N$ th order can rank from 0 to  $f^2 - 1$ . For example, in the case of a filter size 3, the  $N$ th order ranges from 0 to 8 ( $3^2 - 1$ ).

## Grayscale Morphology

---

*Morphological transformations* extract and alter the structure of particles in an image. They fall into two categories:

- *Binary Morphology* functions, which apply to binary images
- *Grayscale morphology* functions, which apply to gray-level images

In grayscale morphology, a pixel is compared to those pixels surrounding it in order to keep those pixel values that are the smallest (erosion) or the largest (dilation).

### When to Use

Use grayscale morphology functions to filter or smooth the pixel intensities of an image. Applications include noise filtering, uneven background correction, and gray-level feature extraction.

### Grayscale Morphology Concepts

The gray-level morphology functions apply to gray-level images. You can use these functions to alter the shape of regions by expanding bright areas at the expense of dark areas and vice versa. These functions smooth gradually varying patterns and increase the contrast in boundary areas. This section describes the following gray-level morphology functions:

- Erosion
- Dilation
- Opening
- Closing

- Proper-opening
- Proper-closing
- Auto-median

These functions are derived from the combination of gray-level erosions and dilations that use a structuring element.

## Erosion Function

A gray-level *erosion* reduces the brightness of pixels that are surrounded by neighbors with a lower intensity. The neighborhood is defined by a structuring element.

## Dilation Function

This function increases the brightness of each pixel that is surrounded by neighbors with a higher intensity. The neighborhood is defined by a structuring element. The *gray-level dilation* has the opposite effect as the gray-level erosion because dilating bright regions also erodes dark regions.

## Erosion and Dilation Examples

This example uses the following source image.

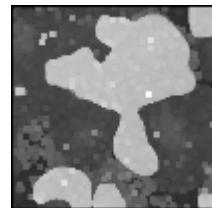
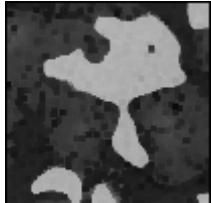
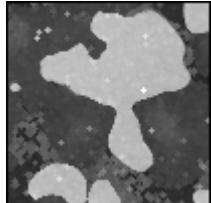
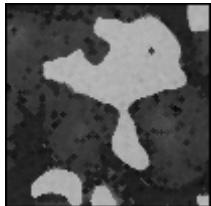
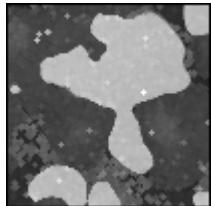


Table 5-3 provides example structuring elements and the corresponding eroded and dilated images.

**Table 5-3.** Erosion and Dilation Examples

Structuring Element	Erosion	Dilation
<pre> 1 1 1 1 <b>1</b> 1 1 1 1 </pre>		
<pre> 0 1 0 1 <b>1</b> 1 0 1 0 </pre>		

## Opening Function

The gray-level *opening function* consists of a gray-level erosion followed by a gray-level dilation. It removes bright spots isolated in dark regions and smooths boundaries. The effects of the function are moderated by the configuration of the structuring element.

$$\text{opening}(I) = \text{dilation}(\text{erosion}(I))$$

This operation does not significantly alter the area and shape of particles because erosion and dilation are morphological opposites. Bright borders reduced by the erosion are restored by the dilation. However, small bright particles that vanish during the erosion do not reappear after the dilation.

## Closing Function

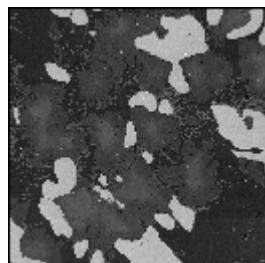
The gray-level *closing function* consists of a gray-level dilation followed by a gray-level erosion. It removes dark spots isolated in bright regions and smooths boundaries. The effects of the function are moderated by the configuration of the structuring element.

$$\text{closing}(I) = \text{erosion}(\text{dilation}(I))$$

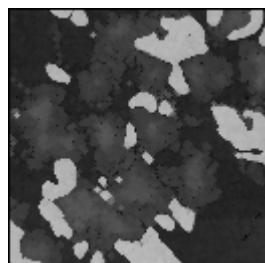
This operation does not significantly alter the area and shape of particles because dilation and erosion are morphological opposites. Bright borders expanded by the dilation are reduced by the erosion. However, small dark particles that vanish during the dilation do not reappear after the erosion.

## Opening and Closing Examples

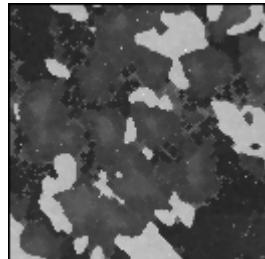
This example uses the following source image.



The opening function produces the following image.



A closing function produces the following image.



**Note** Consecutive applications of an opening or closing function always give the same results.

## Proper-Opening Function

The gray-level *proper-opening function* is a finite and dual combination of openings and closings. It removes bright pixels isolated in dark regions and smooths the boundaries of bright regions. The effects of the function are moderated by the configuration of the structuring element.

## Proper-Closing Function

The *proper-closing function* is a finite and dual combination of closings and openings. It removes dark pixels isolated in bright regions and smooths the boundaries of dark regions. The effects of the function are moderated by the configuration of the structuring element.

## Auto-Median Function

The *auto-median function* uses dual combinations of openings and closings. It generates simpler particles that have fewer details.

## In-Depth Discussion

### Erosion Concept and Mathematics

Each pixel in an image becomes equal to the minimum value of its neighbors.

For a given pixel  $P_0$ , the structuring element is centered on  $P_0$ . The pixels masked by a coefficient of the structuring element equal to 1 are then referred as  $P_i$ .

$$P_0 = \min(P_i)$$



**Note** A gray-level erosion using a structuring element  $f \times f$  with all its coefficients set to 1 is equivalent to an  $N$ th order filter with a filter size  $f \times f$  and the value  $N$  equal to 0. See the [Nonlinear Filters](#) section of this chapter for more information.

### Dilation Concept and Mathematics

Each pixel in an image becomes equal to the maximum value of its neighbors.

For a given pixel  $P_0$ , the structuring element is centered on  $P_0$ . The pixels masked by a coefficient of the structuring element equal to 1 are then referred as  $P_i$ .

$$P_0 = \max(P_i)$$



**Note** A gray-level dilation using a structuring element  $f \times f$  with all its coefficients set to 1 is equivalent to an  $N$ th order filter with a filter size  $f \times f$  and the value  $N$  equal to  $f^2 - 1$  (refer to the nonlinear spatial filters). See the [Nonlinear Filters](#) section of this chapter for more information.

## Proper-Opening Concept and Mathematics

If  $I$  is the source image, the proper-opening function extracts the minimum value of each pixel between the source image  $I$  and its transformed image obtained after an opening, followed by a closing, and followed by another opening.

$$\begin{aligned} \text{proper-opening}(I) &= \min(I, OCO(I)) \text{ or} \\ \text{proper-opening}(I) &= \min(I, DEEDDE(I)) \end{aligned}$$

where  $I$  is the source image,

$E$  is an erosion,

$D$  is a dilation,

$O$  is an opening,

$C$  is a closing,

$F(I)$  is the image obtained after applying the function  $F$  to the image  $I$ , and

$GF(I)$  is the image obtained after applying the function  $F$  to the image  $I$  followed by the function  $G$  to the image  $I$ .

## Proper-Closing Concept and Mathematics

If  $I$  is the source image, the proper-closing function extracts the maximum value of each pixel between the source image  $I$  and its transformed image obtained after a closing, followed by an opening, and followed by another closing.

$$\begin{aligned} \text{proper-closing}(I) &= \max(I, COC(I)) \text{ or} \\ \text{proper-closing}(I) &= \max(I, EDDEED(I)) \end{aligned}$$

where  $I$  is the source image,

$E$  is an erosion,

$D$  is a dilation,

$O$  is an opening,

$C$  is a closing,

$F(I)$  is the image obtained after applying the function  $F$  to the image  $I$ , and

$GF(I)$  is the image obtained after applying the function  $F$  to the image  $I$  followed by the function  $G$  to the image  $I$ .

## Auto-Median Concept and Mathematics

If  $I$  is the source image, the auto-median function extracts the minimum value of each pixel between the two images obtained by applying a proper-opening and a proper-closing of the source image  $I$ .

$$\begin{aligned} \text{auto-median}(I) &= \min(OCO(I), COC(I)) \text{ or} \\ \text{auto-median}(I) &= \min(DEEDDE(I), EDDEED(I)) \end{aligned}$$

where  $I$  is the source image,

$E$  is an erosion,

$D$  is a dilation,

$O$  is an opening,

$C$  is a closing,

$F(I)$  is the image obtained after applying the function  $F$  to the image  $I$ , and

$GF(I)$  is the image obtained after applying the function  $F$  to the image  $I$  followed by the function  $G$  to the image  $I$ .

# Operators

This chapter contains information about arithmetic and logic *operators*, which mask, combine, and compare images.

## Introduction

Operators perform basic arithmetic and logical operations on images. Use operators to add, subtract, multiply, and divide an image with other images or constants. You can also perform logical operations, such as AND/NAND, OR/NOR, and XOR/XNOR, and make pixel comparisons between an image and other images or a constant.

## When to Use

Common applications of these operators include time-delayed comparisons, identification of the union or intersection between images, correction of image backgrounds to eliminate light drifts, and comparisons between several images and a model. You can also use operators to *threshold* or *mask* images and to alter contrast and brightness.

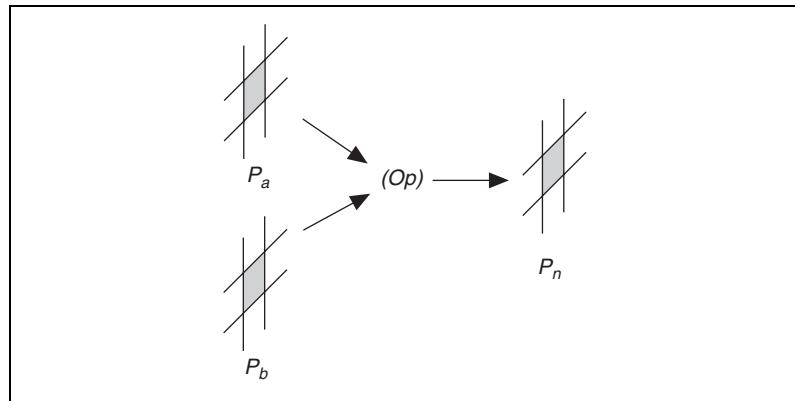
## Operator Concepts

An arithmetic or logic operation between images is a pixel-by-pixel transformation. It produces an image in which each pixel derives its value from the values of pixels with the same coordinates in other images.

If  $A$  is an image with a resolution  $XY$ ,  $B$  is an image with a resolution  $XY$ , and  $Op$  is the operator, then the image  $N$  resulting from the combination of  $A$  and  $B$  through the operator  $Op$  is such that each pixel  $P$  of the resulting image  $N$  is assigned the value

$$p_n = (p_a)(Op)(p_b)$$

where  $p_a$  is the value of pixel  $P$  in image  $A$ , and  $p_b$  is the value of pixel  $P$  in image  $B$ .



## Arithmetic Operators

The equations in Table 6-1 describe the usage of *arithmetic operators* with 8-bit resolution images a and b.

**Table 6-1.** Arithmetic Operators

Operator	Equation
Multiply	$p_n = \min(p_a \times p_b, 255)$
Divide	$p_n = \max(p_a/p_b, 0)$
Add	$p_n = \min(p_a + p_b, 255)$
Subtract	$p_n = \max(p_a - p_b, 0)$
Modulo	$p_n = p_a \bmod p_b$
Absolute Difference	$p_n =  p_a - p_b $

If the resulting pixel value  $p_n$  is negative, it is set to 0. If it is greater than 255, it is set to 255.

## Logic and Comparison Operators

*Logic operators* are bitwise operators. They manipulate gray-level values coded on one byte at the bit level. The equations in Table 6-2 describe the usage of logical operators. The truth tables for logic operators are presented in the *Truth Tables* section in this chapter.

**Table 6-2.** Logical and Comparison Operators

Operator	Equation
Logical Operators	
AND	$p_n = p_a \text{ AND } p_b$
NAND	$p_n = p_a \text{ NAND } p_b$
OR	$p_n = p_a \text{ OR } p_b$
NOR	$p_n = p_a \text{ NOR } p_b$
XOR	$p_n = p_a \text{ XOR } p_b$
Logic Difference	$p_n = p_a \text{ AND } (\text{NOT } p_b)$
Comparison Operators	
Mask	<i>if</i> $p_b = 0$ , <i>then</i> $p_n = 0$ , <i>else</i> $p_n = p_a$
Mean	$p_n = \text{mean}[p_a, p_b]$
Max	$p_n = \max[p_a, p_b]$
Min	$p_n = \min[p_a, p_b]$

In the case of images with 8-bit resolution, logic operators are mainly designed to combine gray-level images with mask images composed of pixels equal to 0 or 255 (in binary format 0 is represented as 00000000, and 255 is represented as 11111111), or to combine or compare images with a binary or labeled content (after thresholding the image).

Table 6-3 illustrates how logic operators can be used to extract or remove information in an image.

**Table 6-3.** Using Logical Operators with Binary Image Masks

<i>For a given <math>p_a</math></i>	<i>If <math>p_b = 255</math>, then</i>	<i>If <math>p_b = 0</math>, then</i>
AND	$p_a \text{ AND } 255 = p_a$	$p_a \text{ AND } 0 = 0$
NAND	$p_a \text{ NAND } 255 = \text{NOT } p_a$	$p_a \text{ NAND } 0 = 255$
OR	$p_a \text{ OR } 255 = 255$	$p_a \text{ OR } 0 = p_a$
NOR	$p_a \text{ NOR } 255 = 0$	$p_a \text{ NOR } 0 = \text{NOT } p_a$
XOR	$p_a \text{ XOR } 255 = \text{NOT } p_a$	$p_a \text{ XOR } 0 = p_a$
Logic Difference	$p_a - \text{NOT } 255 = p_a$	$p_a - \text{NOT } 0 = 0$

## Truth Tables

The following truth tables describe the rules used by the logic operators. The top row and left column give the values of input bits. The cells in the table give the output value for a given set of two input bits.

AND			
		$b = 0$	$b = 1$
$a = 0$	0	0	
$a = 1$	0	1	

NAND			
		$b = 0$	$b = 1$
$a = 0$	1	1	
$a = 1$	1	0	

OR			
		$b = 0$	$b = 1$
$a = 0$	0	1	
$a = 1$	1	1	

NOR			
		$b = 0$	$b = 1$
$a = 0$	1	0	
$a = 1$	0	0	

**XOR**

		$b = 0$	$b = 1$
		$a = 0$	0    1
		$a = 1$	1    0

**XNOR**

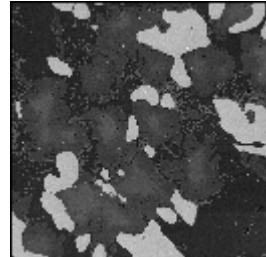
		$b = 0$	$b = 1$
		$a = 0$	1    0
		$a = 1$	0    1

**NOT**

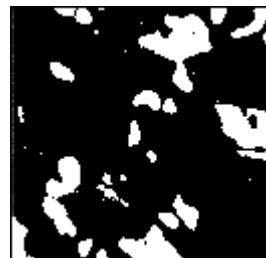
		$\text{NOT } a$
		$a = 0$
		$a = 1$

## Example 1

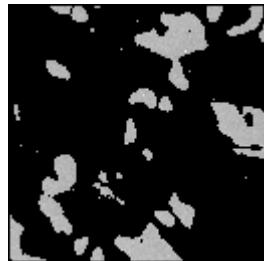
The following figure shows the source grayscale image used in this example.



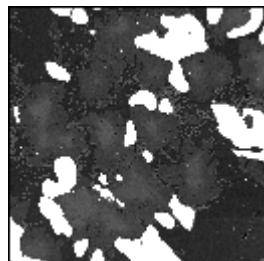
Regions of interest have been isolated in a binary format, retouched with morphological manipulations, and finally multiplied by 255 to obtain the following *mask image*:



The operation (*source image AND mask image*) restores the original intensity of the object regions in the mask.



The operation (*source image OR mask image*) restores the original intensity of the background region in the mask.

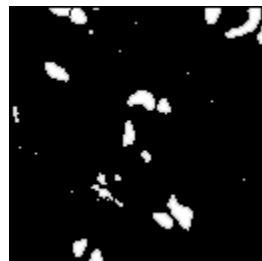


## Example 2

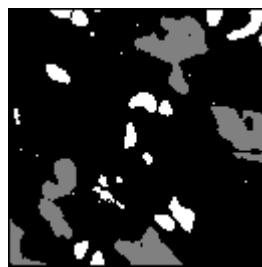
This example demonstrates the use of the OR operation to produce an image containing the union of two binary images. The following image represents the first image, with a background of 0 and objects with a gray-level value of 128.



The following figure shows the second image, featuring a background of 0 and objects with gray-level values of 255.



*Image #1 OR Image #2* produces a union, as shown in the following image.



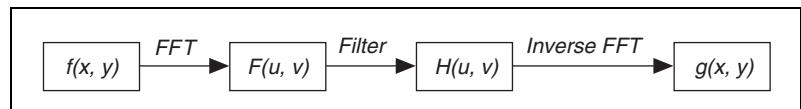
# Frequency Domain Analysis

This chapter contains information about converting images into the frequency domain using the Fast Fourier transform, and information about analyzing and processing images in the frequency domain.

## Introduction

*Frequency filters* alter pixel values with respect to the periodicity and spatial distribution of the variations in light intensity in the image. Unlike spatial filters, frequency filters do not apply directly to a spatial image, but to its frequency representation. The frequency representation of an image is obtained through a function called the *Fast Fourier transform* (FFT). It reveals information about the periodicity and dispersion of the patterns found in the source image.

You can filter the spatial frequencies seen in an FFT image. The inverse FFT function then restores a spatial representation of the filtered FFT image.



*Frequency processing* is another technique for extracting information from an image. Instead of using the location and direction of light-intensity variations, you can use frequency processing to manipulate the frequency of the occurrence of these variations in the spatial domain. This new component is called the *spatial frequency*, which is the frequency with which the light intensity in an image varies as a function of spatial coordinates.

Spatial frequencies of an image are computed with the Fast Fourier transform (FFT). The FFT is calculated in two steps—a 1D Fast Fourier transform of the rows, followed by a Fast Fourier 1D transform of the columns of the previous results. The complex numbers that compose the FFT plane are encoded in a 64-bit floating-point image called a complex image. The complex image is formed by a 32-bit floating point number

representing the real part and a 32-bit floating point number representing the imaginary part.

In an image, details and sharp edges are associated with mid- to high spatial frequencies because they introduce significant gray-level variations over short distances. Gradually varying patterns are associated with low spatial frequencies. By filtering spatial frequencies, you can remove, attenuate, or highlight the spatial components to which they relate.

Use a *lowpass frequency filter* to attenuate or remove (truncate) high frequencies present in the image. This filter suppresses information related to rapid variations of light intensities in the spatial image. An inverse FFT, used after a lowpass frequency filter, produces an image in which noise, details, texture, and sharp edges are smoothed.

A *highpass frequency filter* attenuates or removes (truncates) low frequencies present in the complex image. This filter suppresses information related to slow variations of light intensities in the spatial image. In this case, an inverse FFT used after a highpass frequency filter produces an image in which overall patterns are sharpened and details are emphasized.

A *mask frequency filter* removes frequencies contained in a mask specified by the user. Using a mask to alter the Fourier transform of an image offers more possibilities than applying a lowpass or highpass filter. The image mask is composed by the user and can describe very specific frequencies and directions in the image. You can apply this technique, for example, to filter dominant frequencies as well as their harmonics in the frequency domain.

## When to Use

In an image, details and sharp edges are associated with mid- to high spatial frequencies. They are associated because details and sharp edges introduce significant gray-level variations over short distances. Gradually varying patterns are associated with low spatial frequencies.

An image can have extraneous noise introduced during the digitization process, such as periodic stripes. In the frequency domain, the periodic pattern is reduced to a limited set of high spatial frequencies. Truncating these particular frequencies and converting the filtered FFT image back to the spatial domain produces a new image in which the grid pattern has disappeared, while the overall features remain.

## Fast Fourier Transform Concepts

The FFT of an image is a two-dimensional array of complex numbers, also represented as a complex image. It represents the frequencies of occurrence of light-intensity variations in the spatial domain. The low frequencies correspond to smooth and gradual intensity variations found in the overall patterns of the source image. The high frequencies correspond to abrupt and short-intensity variations found at the edges of objects, around noisy pixels, and around details.

## FFT Representation

There are two possible representations of the Fast Fourier transform of an image: the *standard representation* and the *optical representation*.

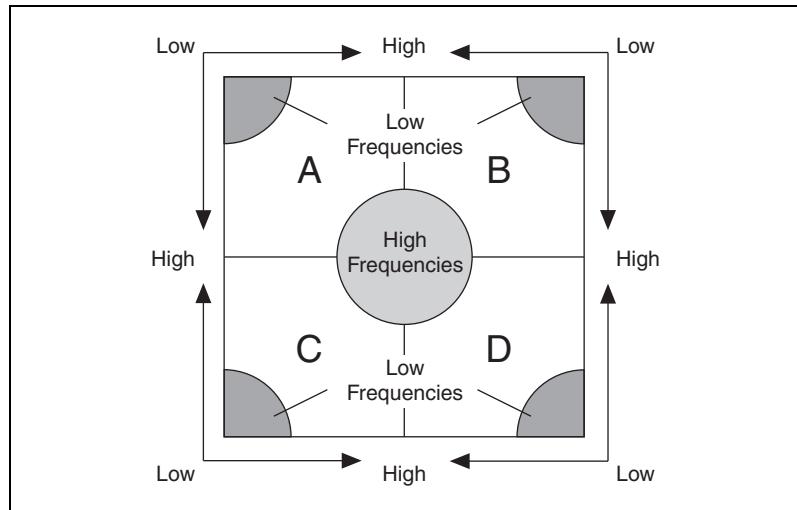
### Standard Representation

In the standard representation, high frequencies are grouped at the center of the image while low frequencies are located at the edges. The constant term, or null frequency, is in the upper-left corner of the image. The frequency range is

$$[0, N] \times [0, M]$$

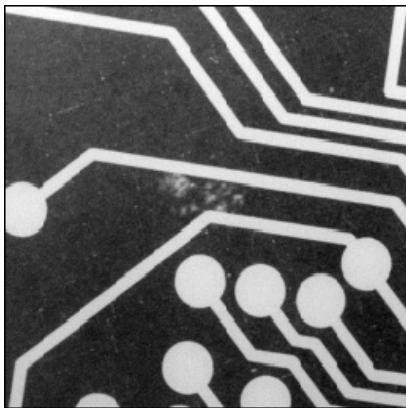
where  $M$  is the horizontal resolution of the image

$N$  is the vertical resolution of the image

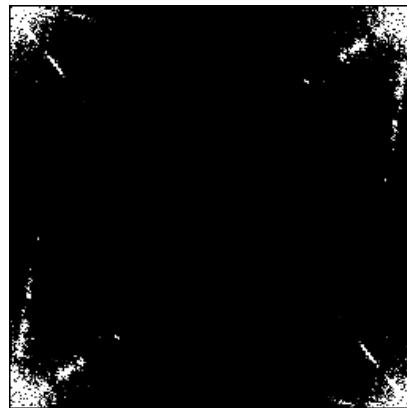


**Note** IMAQ Vision uses this representation to represent complex images in memory. Use this representation when building an image mask.

Figure 7-1a shows an image. Figure 7-1b shows the FFT of the same image using standard representation.



a. Original Image



b. FFT in Standard Representation

**Figure 7-1. FFT of an Image in Standard Representation**

## Optical Representation

In the optical representation, low frequencies are grouped at the center of the image while high frequencies are located at the edges. The constant term, or null frequency, is at the center of the image. The frequency range is

$$\left[-\frac{N}{2}, \frac{N}{2}\right] \times \left[-\frac{M}{2}, \frac{M}{2}\right]$$

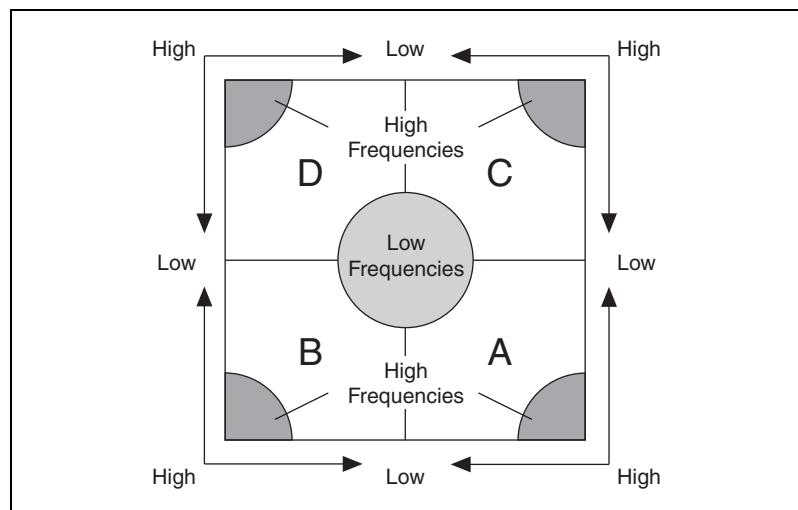
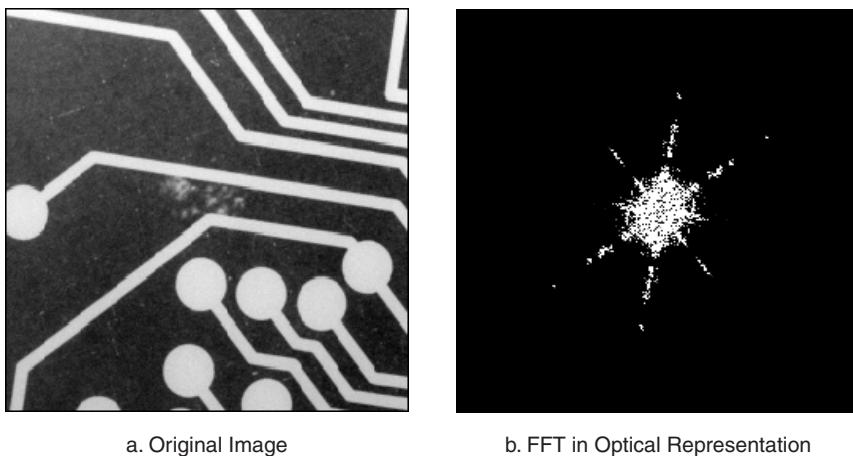


Figure 7-2a shows the same original image as shown in Figure 7-1a. Figure 7-2b shows the FFT of the image in optical representation.



**Figure 7-2.** FFT of an Image in Optical Representation



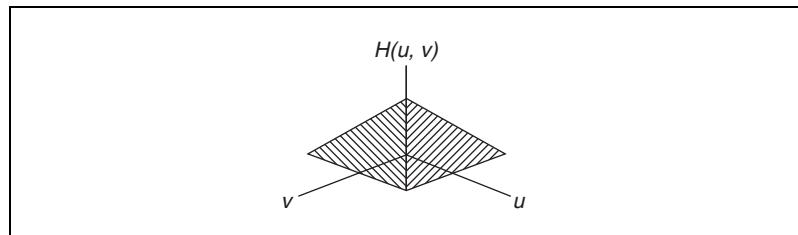
**Note** This is the representation IMAQ Vision uses when displaying a complex image.

You can switch from standard representation to optical representation by permuting the  $A$ ,  $B$ ,  $C$ , and  $D$  quarters.

Intensities in the FFT image are proportional to the amplitude of the displayed component.

## Lowpass FFT Filters

A *lowpass frequency filter* attenuates or removes high frequencies present in the FFT plane. This filter suppresses information related to rapid variations of light intensities in the spatial image. In this case, an inverse FFT produces an image in which noise, details, texture, and sharp edges are smoothed.



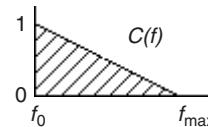
A lowpass frequency filter removes or attenuates spatial frequencies located outside a frequency range centered on the fundamental (or null) frequency.

## Lowpass Attenuation

*Lowpass attenuation* applies a linear attenuation to the full frequency range, increasing from the null frequency  $f_0$  to the maximum frequency  $f_{\max}$ . This is done by multiplying each frequency by a coefficient  $C$ , which is a function of its deviation from the fundamental and maximum frequencies.

$$C(f) = \frac{f_{\max} - f}{f_{\max} - f_0}$$

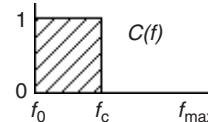
where  $C(f_0) = 1$  and  $C(f_{\max}) = 0$



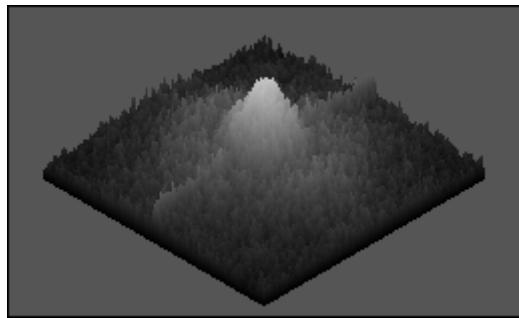
## Lowpass Truncation

*Lowpass truncation* removes a frequency  $f$  if it is higher than the cutoff or truncation frequency,  $f_c$ . This is done by multiplying each frequency  $f$  by a coefficient  $C$  equal to 0 or 1, depending on whether the frequency  $f$  is greater than the truncation frequency  $f_c$ .

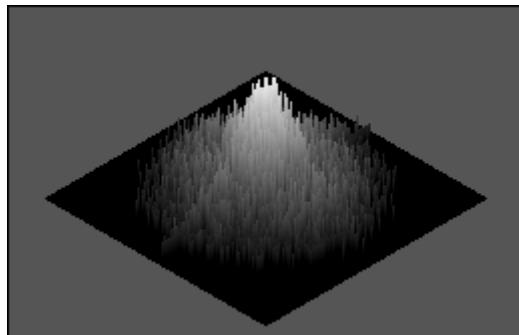
```
If      f > f_c
then   C(f) = 0
else   C(f) = 1
```



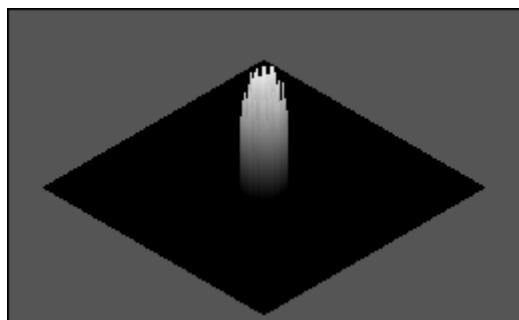
The following series of graphics illustrates the behavior of both types of lowpass filters. They give the 3D-view profile of the magnitude of the FFT. This example uses the following original FFT.



After lowpass attenuation, the magnitude of the central peak is the same, and variations at the edges almost have disappeared.

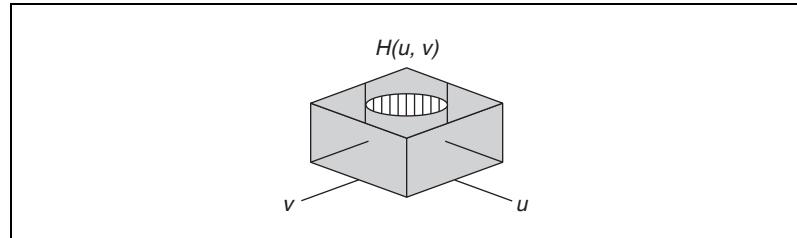


After lowpass truncation with  $f_c = f_0 + 20\%(f_{\max} - f_0)$ , spatial frequencies outside the truncation range  $[f_0, f_c]$  are removed. The part of the central peak that remains is identical to the one in the original FFT plane.



## Highpass FFT Filters

A *highpass FFT filter* attenuates or removes low frequencies present in the FFT plane. It has the effect of suppressing information related to slow variations of light intensities in the spatial image. In this case, the Inverse FFT command produces an image in which overall patterns are attenuated and details are emphasized.

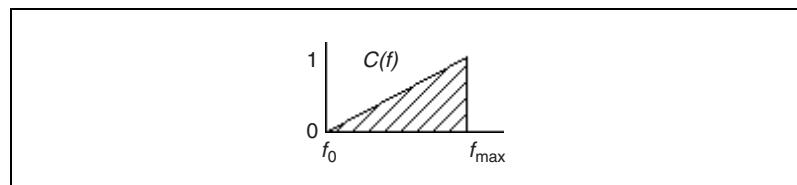


### Highpass Attenuation

*Highpass attenuation* applies a linear attenuation to the full frequency range, increasing from the maximum frequency  $f_{\max}$  to the null frequency  $f_0$ . This is done by multiplying each frequency by a coefficient  $C$ , which is a function of its deviation from the fundamental and maximum frequencies.

$$C(f) = \frac{f - f_0}{f_{\max} - f_0}$$

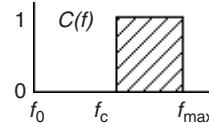
where  $C(f_0) = 1$  and  $C(f_{\max}) = 0$ .



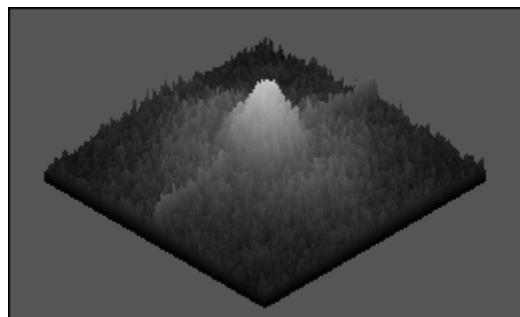
## Highpass Truncation

*Highpass truncation* removes a frequency  $f$  if it is lower than the cutoff or truncation frequency,  $f_c$ . This is done by multiplying each frequency  $f$  by a coefficient  $C$  equal to 1 or 0, depending on whether the frequency  $f$  is greater than the truncation frequency  $f_c$ .

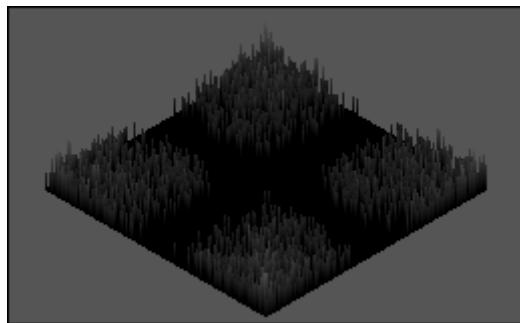
If       $f < f_c$   
then     $C(f) = 0$   
else     $C(f) = 1$



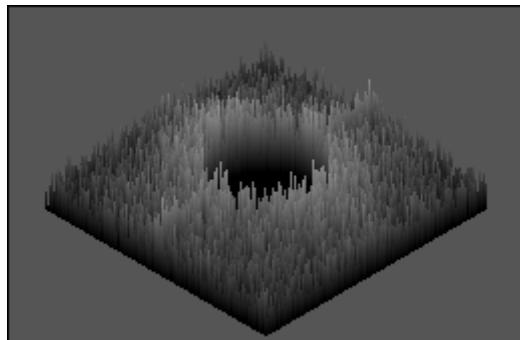
The following series of graphics illustrates the behavior of both types of highpass filters. They give the 3D-view profile of the magnitude of the FFT. This example uses the following original FFT image.



After highpass attenuation, the central peak has been removed, and variations present at the edges remain.

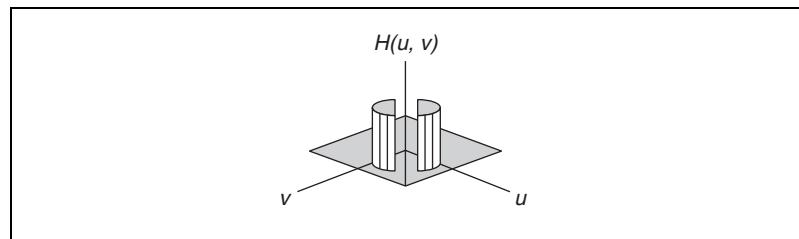


After highpass truncation with  $f_c = f_0 + 20\%(f_{\max} - f_0)$ , spatial frequencies inside the truncation range  $[f_0, f_c]$  are set to 0. The remaining frequencies are identical to the ones in the original FFT plane.



## Mask FFT Filters

A *mask FFT filter* removes frequencies contained in a mask specified by the user. Depending on the mask definition, this filter can behave as a lowpass, bandpass, highpass, or any type of selective filter.



## In-Depth Discussion

### Fourier Transform

The spatial frequencies of an image are calculated by a function called the *Fourier Transform*. It is defined in the continuous domain as

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j2\pi(xu + yv)} dx dy$$

where  $f(x, y)$  is the light intensity of the point  $(x, y)$ , and  $(u, v)$  are the horizontal and vertical spatial frequencies. The Fourier Transform assigns a complex number to each set  $(u, v)$ .

Inversely, a Fast Fourier Transform  $F(u, v)$  can be transformed into a spatial image  $f(x, y)$  of resolution  $NM$  using the following formula:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} F(u, v) e^{j2\pi\left(\frac{ux}{N} + \frac{vy}{M}\right)}$$

In the discrete domain, the Fourier Transform is calculated with an efficient algorithm called the Fast Fourier Transform (FFT).

$$F(u, v) = \frac{1}{NM} \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) e^{-j2\pi\left(\frac{ux}{N} + \frac{vy}{M}\right)}$$

where  $N \times M$  is the resolution of the spatial image  $f(x, y)$ .

Because  $e^{-j2\pi ux} = \cos 2\pi ux - j \sin 2\pi ux$ ,  $F(u, v)$  is composed of an infinite sum of sine and cosine terms. Each pair  $(u, v)$  determines the frequency of its corresponding sine and cosine pair. For a given set  $(u, v)$ , notice that all values  $f(x, y)$  contribute to  $F(u, v)$ . Because of this complexity, the FFT calculation is time consuming.

Given an image with a resolution  $N \times M$  and given  $\Delta x$  and  $\Delta y$  the spatial step increments, the FFT of the source image has the same resolution  $NM$  and its frequency step increments  $\Delta u$  and  $\Delta v$ , which are defined in the following equations:

$$\Delta u = \frac{1}{N \times \Delta x} \quad \Delta v = \frac{1}{M \times \Delta y}$$

## FFT Display

An FFT image can be visualized using any of its four complex components: real part, imaginary part, magnitude, and phase. The relation between these components is expressed by

$$F(u, v) = R(u, v) + jI(u, v)$$

where  $R(u, v)$  is the real part and  $I(u, v)$  is the imaginary part, and

$$F(u, v) = |F(u, v)| \times e^{j\varphi(u, v)}$$

where  $|F(u, v)|$  is the magnitude and  $\varphi(u, v)$  is the phase.

The magnitude of  $F(u, v)$  is also called the *Fourier spectrum* and is equal to

$$|F(u, v)| = \sqrt{R(u, v)^2 + I(u, v)^2}$$

The Fourier spectrum to the power of two is known as the power spectrum or spectral density.

The phase  $\varphi(u, v)$  is also called the phase angle and is equal to

$$\varphi(u, v) = \text{atan} \left[ \frac{I(u, v)}{R(u, v)} \right]$$

By default, when you display a complex image, the magnitude plane of the complex image is displayed using the optical representation. To visualize the magnitude values properly, the magnitude values are scaled by the factor  $m$  before they are displayed. The factor  $m$  is calculated as

$$\frac{128}{w \times h}$$

where  $w$  is the width of the image and  
 $h$  is the height of the image.

# Part III

---

## Blob Analysis

This section describes conceptual information about blob analysis, including thresholding, morphology, and particle measurements.

Part III, *Blob Analysis*, contains the following chapters:

Chapter 8, *Thresholding*, contains information about thresholding and color thresholding.

Chapter 9, *Binary Morphology*, contains information about structuring elements, connectivity, and primary and advanced morphological transformations.

Chapter 10, *Particle Measurements*, contains information about areas, lengths, coordinates, chords and axes, shape equivalence, shape features, and diverse measurements of particles.

## Introduction

---

*Blobs* are areas of touching pixels within an image, in which all pixels have the same logical state. All pixels in an image that belong to a blob are in a foreground state. All other pixels are in a background state. In a binary image, pixels in the background have values equal to zero while every non-zero pixel is part of a binary object.

You can use blob analysis to detect connected regions or groupings of pixels in an image and then make selected measurements of those regions. These regions are commonly referred to as *blobs* (*binary large objects*).

Blob analysis consists of a series of processing operations and analysis functions that produce information about blobs in an image. Using blob

analysis you can detect and analyze any two-dimensional shape in an image.

## When to Use

Use blob analysis when you are interested in finding blobs whose spatial characteristics satisfy certain criteria. In many applications where computation is time-consuming, you can use blob analysis to eliminate blobs that are of no interest based on their spatial characteristics, and keep only the relevant blobs for further analysis.

You can use blob analysis to find statistical information—such as the size of the blobs or the number, location, and the presence of blob regions. This information allows you to perform many machine vision inspection tasks—such as detecting flaws on silicon wafers, detecting soldering defects on electronic boards, or web inspection applications such as finding structural defects on wood planks or detecting cracks on plastics sheets. You can also locate objects in motion control applications when there is significant variance in part shape or orientation.

In applications where there is a significant variance in the shape or orientation of an object, blob analysis is a powerful and flexible way to search for the object. You can use a combination of the measurements obtained through blob analysis to define a feature set that uniquely defines the shape of the object.

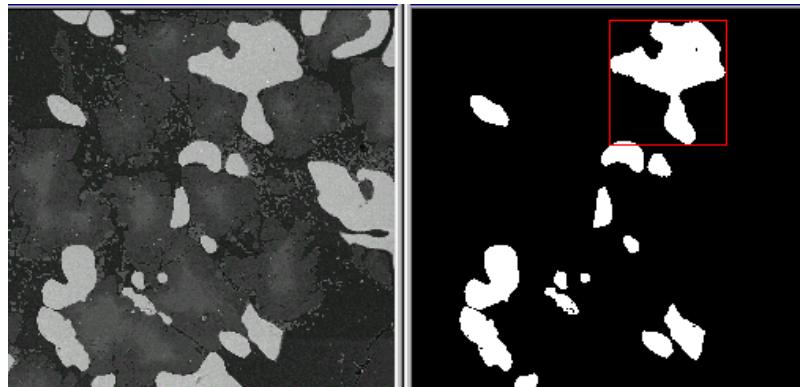
## Blob Analysis Concepts

A typical blob analysis process scans through an entire image and detects all the particles, or blobs, in the image and builds a detailed report on each particle. This report usually contains approximately 50 pieces of information about the blob, including the blob's location in the image, size, shape, orientation to other blobs, longest segment, and moment of inertia.

You can use multiple parameters such as perimeter, angle, area, and center of mass to identify and classify these blobs. Using multiple parameters can be faster and more effective than pattern matching in many applications.

Also, by using different sets of parameters, you can uniquely identify a feature in an image. For example, you could use the size of the template blob as a criterion for removing all blobs that do not match it within some tolerance. You can then perform a more refined search on the remaining particles using another list of parameter tolerances. These include the longest segment in each blob and compactness factor (the ratio of the area of the blob to the area of the smallest rectangle that encloses the blob).

The following figure shows a sample list of parameters that you can obtain in a blob analysis application. The binary image in this example was obtained by thresholding the source image and removing particles that touch the border of the image. You can use these parameters to identify and classify particles. The following table shows the values obtained for the blob enclosed in a rectangle, shown in the figure below:



Global rectangle:

x1Left	125
y2Top	7
x2Right	198
y2Bottom	85
Area (pixels)	2456
Number of holes	1
Area (calibrated)	2456.00
Hole's area (pixels)	2
Sum X	406482.00
Sum XX	67885136.00
Sum Y	89909.00
Sum YY	4158045.00
Sum XY	14856285.00
Projection x	99
Projection y	94

Perimeter	289.02
Hole's perimeter	5.01
Longest segment coordinates ( <i>x</i> and <i>y</i> )	<i>x</i> = 125 <i>y</i> = 36
Longest segment length	72

To use blob analysis, first create a binary image using a thresholding process. You can then improve the binary image using morphological transformations and make measurements on the particles in the image.

# Thresholding

This chapter contains information about thresholding and color thresholding.

## Introduction

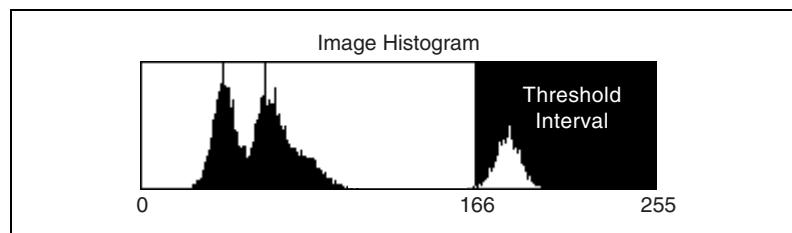
Thresholding consists of segmenting an image into two regions: a particle region and a background region. This process works by setting to 1 all pixels that belong to a gray-level interval, called the threshold interval, and setting all other pixels in the image to 0.

Use thresholding to isolate objects of interest in an image. Thresholding converts the image from a grayscale image, with pixel values ranging from 0 to 255, to a binary image, with pixel values of 0 or 1.

Thresholding enables you to select ranges of pixel values in grayscale and color images that separate the objects under consideration from the background.

## When to Use

Use thresholding to extract areas that correspond to significant structures in an image and to focus the analysis on these areas.



Pixels outside the threshold interval are set to 0 and are considered as part of the background area. Pixels inside the threshold interval are set to 1 and are considered as part of a particle area.

## Thresholding Concepts

### Intensity Threshold

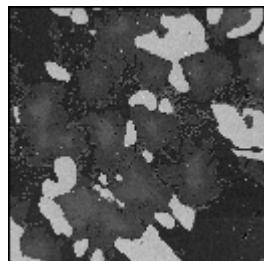
Particles are characterized by an *intensity range*. They are composed of pixels with gray-level values belonging to a given threshold interval (overall luminosity or gray shade). All other pixels are considered to be part of the background.

The *threshold interval* is defined by the two parameters

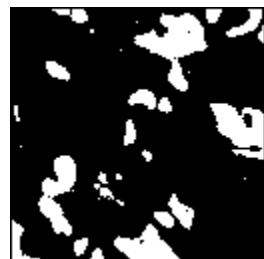
**[Lower Threshold, Upper Threshold]**. All pixels that have gray-level values equal to or greater than the Lower Threshold and equal to or smaller than the Upper Threshold are selected as pixels belonging to particles in the image.

### Thresholding Example

This example uses the following source image.



Highlighting the pixels that belong to the threshold interval [166, 255] (the brightest areas) produces the following image.



A critical and frequent problem in segmenting an image into particle and background regions occurs when the boundaries are not sharply demarcated. In such a case, the choice of a correct threshold becomes subjective. Therefore, you may want to enhance your images before thresholding to outline where the correct borders lie. You can use lookup

tables, filters, FFTs, or equalize functions to enhance your images.

Observing the intensity profile of a line crossing a boundary area is also helpful in selecting a correct threshold value. Finally, keep in mind that morphological transformations can help you retouch the shape of binary particles and therefore correct unsatisfactory selections that occurred during the thresholding.

## Automatic Threshold

Various automatic thresholding techniques are available:

- Clustering
- Entropy
- Metric
- Moments
- Interclass Variance

In contrast to manual thresholding, these methods do not require that you set the minimum and maximum light intensities. These techniques are well suited for conditions in which the light intensity varies.

Depending on your source image, it is sometimes useful to invert (reverse) the original grayscale image before applying an automatic threshold function, such as moments and entropy. This is especially true for cases in which the background is brighter than the foreground.

Clustering is the only multi-class thresholding method available. Clustering operates on multiple classes so you can create tertiary or higher-level images. The other four methods (entropy, metric, moments, and interclass variance) are reserved for strictly binary thresholding techniques.

The choice of which algorithm to apply depends on the type of image to threshold.

### Clustering

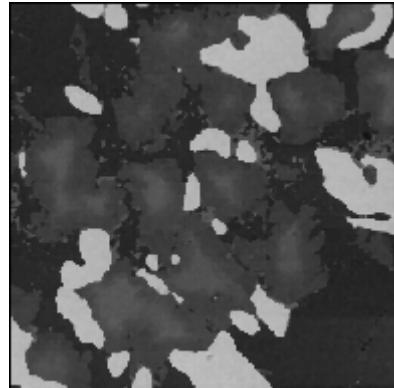
This technique sorts the histogram of the image within a discrete number of classes corresponding to the number of phases perceived in an image. The gray values are determined, and a *barycenter* is determined for each class. This process repeats until it obtains a value that represents the center of mass for each phase or class.

The automatic thresholding method most frequently used is clustering, also known as multi-class thresholding.

### Example of Clustering

This example uses a clustering technique in two and three phases on an image. Notice that the results from this function are generally independent of the lighting conditions as well as the histogram values from the image.

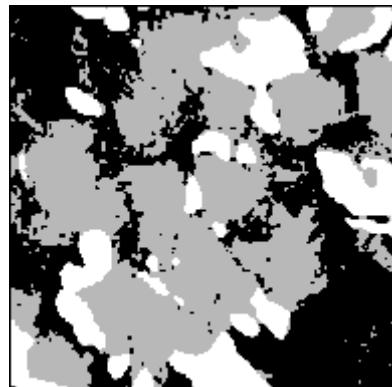
This example uses the following original image.



Clustering in two phases produces the following image.



Clustering in three phases produces the following image.



## Entropy

Based on a classical image analysis technique, entropy is best for detecting particles that are present in minuscule proportions on the image. For example, this function would be suitable for default detection.

## Metric

Use this technique in situations similar to interclass variance. For each threshold, a value is calculated that is determined by the surfaces representing the initial gray scale. The optimal threshold corresponds to the smallest value.

## Moments

This technique is suited for images that have poor contrast. The moments method is based on the hypothesis that the observed image is a blurred version of the theoretically binary original. The blurring that is produced from the acquisition process, caused by electronic noise or slight defocalization, is treated as if the statistical moments (average and variance) were the same for both the blurred image and the original image. This function recalculates a theoretical binary image.

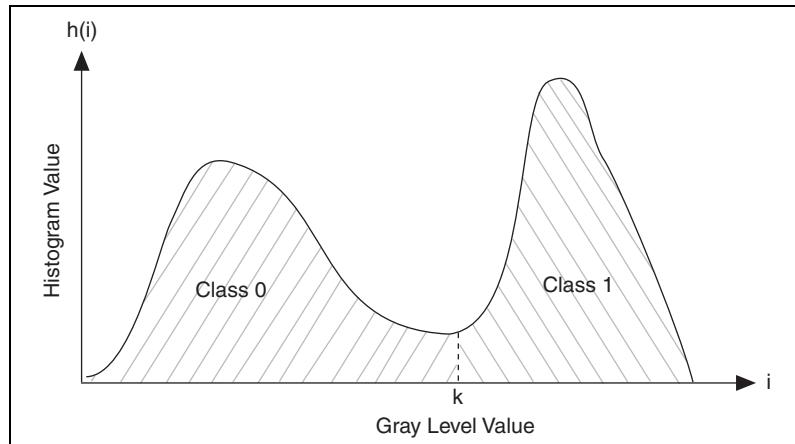
## Interclass Variance

Interclass variance is a classical statistical technique used in discriminating factorial analysis. This method is well suited for images in which classes are not too disproportionate. For satisfactory results, the smallest class must be at least 5% of the largest one. Notice that this method tends to underestimate the class of the smallest standard deviation if the two classes have a significant variation.

## In-Depth Discussion

### Auto-Thresholding Techniques

All auto-thresholding methods use the histogram of an image to determine the threshold. The following notations are used to describe the parameters of the histogram:



$i$ —Represents the gray level value

$k$ —Represents the gray level value chosen as the threshold

$h(i)$ —Represents the number of pixels in the image at each gray level value

$N$ —Represents the total number of gray levels in the image (256 for an eight bit image)

$n$ —Total number of pixels in the image

IMAQ Vision currently has five auto-thresholding techniques

- Clustering
- Entropy

- InterVariance
- Metric
- Moments

All of the five methods can be used to threshold an image into two classes. The auto thresholding techniques are used to determine the threshold pixel value ( $k$ ) such that all gray level values less than or equal to  $k$  belong to one class 0 and the other gray level values belong to another class 1 as shown in the figure above.

The *clustering* method is used when the image has to be thresholded into more than two classes.

## Clustering

The threshold value is the pixel value  $k$  for which the following condition is true:

$$\frac{\mu_1 + \mu_2}{2} = k,$$

where  $\mu_1$  is the mean of all pixel values in the image that lie between 0 and  $k$ , and  $\mu_2$  is the mean of all the pixel values in the image that lie between  $k+1$  and 255.

## Entropy

In this method the threshold value is obtained by applying information theory to the histogram data. In information theory, the entropy of the histogram signifies the amount of information associated with the histogram. Let

$$p(i) = \frac{h(i)}{\sum_{i=0}^{N-1} h(i)}$$

represent the probability of occurrence of the gray level  $i$ . The entropy of a histogram of an image with gray levels in the range [0,N-1] is given by:

$$H = - \sum_{i=0}^{i=N-1} p(i) \log_e p(i).$$

If k is the value of the threshold, then the two entropies

$$H_b = - \sum_{i=0}^{i=k} p(i) \log_e p(i),$$

$$H_w = - \sum_{i=k+1}^{i=N-1} p(i) \log_e p(i)$$

represent the measures of the entropy (information) associated with the black and white pixels in the image after thresholding. The optimal threshold value is gray level value that maximizes the entropy in the thresholded image given by:

$$H = H_b + H_w$$

Simplified, the threshold value is the pixel value  $k$  at which the following expression is maximized:

$$-\sum_{i=0}^{i=k} \log(h(i) + 1)(h(i)) - \sum_{i=k+1}^{i=N-1} \log(h(i) + 1)(h(i)) + \log\left(\sum_{i=0}^{i=k} h(i) \sum_{i=k+1}^{i=N-1} h(i)\right)$$

## InterVariance

Works similar to the clustering technique.

## Metric

The threshold value is the pixel value  $k$  at which the following expression is minimized:

$$\sum_{i=0}^{i=k} h(i)|(i - \mu_1)| = \sum_{i=k+1}^{i=N-1} h(i)|(i - \mu_2)|,$$

where,  $\mu_1$  is the mean of all pixel values in the image that lie between 0 and  $k$ , and  $\mu_2$  is the mean of all the pixel values in the image that lie between  $k+1$  and 255.

## Moments

In this method the threshold value is computed in such a way that the moments of the image to be thresholded are preserved in the output (binary) image.

*The kth moment m* of an image is calculated as

$$m_k = \frac{1}{n} \sum_{i=0}^{i=N-1} i^k h(i),$$

where *n* is the total number of pixels in the image.

## Color Thresholding

Color thresholding converts a color image to a binary image.

To threshold a color image, three threshold intervals need to be specified, one for each color component. A pixel in the output image is set if and only if its color components fall within the specified range. Otherwise, the pixel is set to 0.

## When to Use

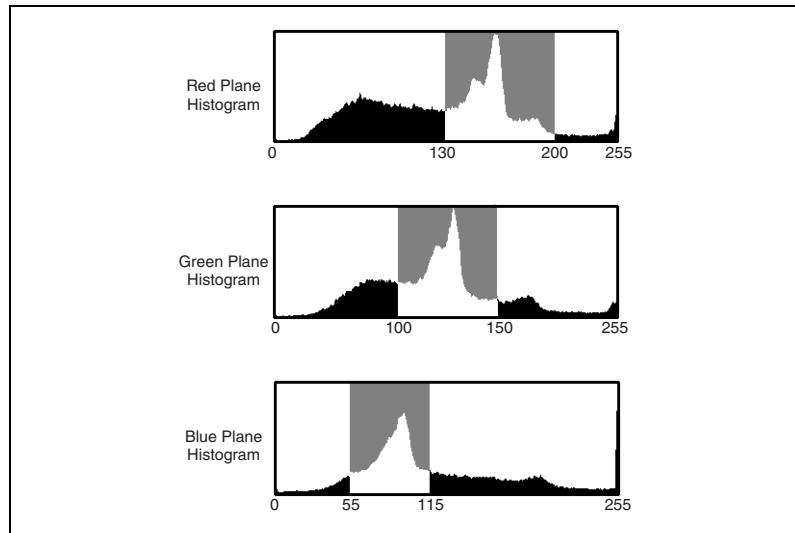
Threshold a color image when you need to isolate features for analysis and processing and remove features that do not interest you.



**Note** Before performing a color threshold, you may need to enhance your image with lookup tables or the equalize function.

To threshold a color image, specify a threshold interval for each of the three color components. The value of a pixel in the original image is set to 1 if and only if its color components fall within the specified range. Otherwise, the pixel value is set to 0.

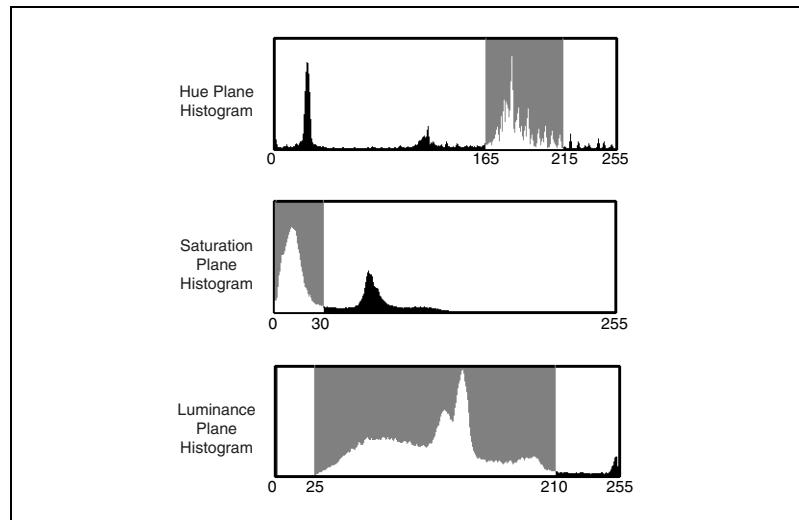
Figure 8-1 shows the histograms of each plane of a color image stored in RGB format. The gray shaded region indicates the threshold range for each of the color planes. For a pixel in the color image to be set to 1 in the binary image, its red value should lie between 130 and 200, its green value should lie between 100 and 150, and its blue value should lie between 55 and 115.



**Figure 8-1.** Threshold Ranges for an RGB Image

To threshold an RGB image, first determine the red, green, and blue values of the pixels that constitute the objects you want to analyze after thresholding. Then, specify a threshold range for each color plane that encompasses the color values of interest. You must choose correct ranges for all three color planes to isolate a color of interest.

Figure 8-2 shows the histograms of each plane of a color image stored in HSL format. The gray shaded region indicates the threshold range for each of the color planes. For a pixel in the color image to be set to 1 in the binary image, its hue value should lie between 165 and 215, its saturation value should lie between 0 and 30, and its luminance value should lie between 25 and 210.



**Figure 8-2.** Threshold Ranges for an HSL Image

The hue plane contains the main color information in an image. To threshold an HSL image, first determine the hue values of the pixels that you want to analyze after thresholding. In some applications, you may need to select colors with the same hue value but various saturation values. Because the luminance plane only contains information about the intensity levels in the image, you can set the luminance threshold range to include all the luminance values, making the thresholding process independent from intensity information.

# Binary Morphology

This chapter contains information about structuring elements, connectivity, and primary and advanced binary morphology operations.

## Introduction

*Binary morphological operations* extract and alter the structure of particles in a binary image. You can use these operations during your inspection application to improve the information in a binary image before making particle measurements, such as the area, perimeter, and orientation.

A *binary image* is an image containing particle regions with pixel values of 1 and a background region with pixel values of 0. Binary images are the result of the *thresholding* process. Because thresholding is a subjective process, the resulting binary image may contain unwanted information, such as noise particles, particles touching the border of images, particles touching each other, and particles with uneven borders. By affecting the shape of particles, morphological functions can remove this unwanted information, thus improving the information in the binary image.

## Structuring Elements

Morphological operators that change the shape of particles process a pixel based on its number of neighbors and the values of those neighbors. A *neighbor* is a pixel whose value affects the values of nearby pixels during certain image processing functions. Morphological transformations use a 2D binary mask called a *structuring element* to define the size and effect of the neighborhood on each pixel, controlling the effect of the binary morphological functions on the shape and the boundary of a particle.

## When to Use

Use a structuring element when you perform any primary binary morphology operation or the advanced binary morphology operation Separation. You can modify the size and the values of a structuring element to alter the shape of the particles in a specific way. However, you should

study the basic morphology operations before defining your own structuring element.

## Structuring Elements Concepts

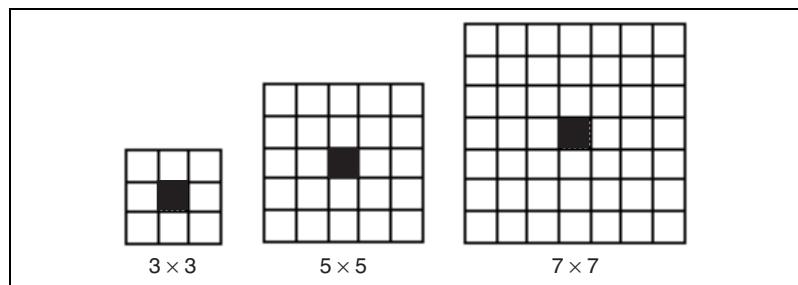
The size and contents of a structuring element specify which pixels a morphological operation takes into account when determining the new value of the pixel being processed. A structuring element must have an odd-sized axis to accommodate a center pixel, which is the pixel being processed. The contents of the structuring element are always binary, composed of 1 and 0 values. The most common structuring element is a  $3 \times 3$  matrix containing values of 1. This matrix, shown below, is the default structuring element for most binary and grayscale morphological transformations.

$$\begin{matrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{matrix}$$

Three factors influence how a structuring element defines which pixels to process during a morphological transformation: the size of the structuring element, the values of the structuring element sectors, and the shape of the pixel frame.

## Structuring Element Size

The size of a structuring element determines the size of the neighborhood surrounding the pixel being processed. The coordinates of the pixel being processed are determined as a function of the structuring element. In Figure 9-1, the coordinates of the pixels being processed are (1,1), (2,2), and (3,3), respectively. The origin (0,0) is always the top, left corner pixel.



**Figure 9-1.** Structuring Element Sizes

Using structuring elements requires an image border. A  $3 \times 3$  structuring element requires a minimum border size of 1. In the same way, structuring elements of  $5 \times 5$  and  $7 \times 7$  require a minimum border size of 2 and 3, respectively. Bigger structuring elements require corresponding increases in the image border size. For more information about image borders, see the *Image Borders* section of Chapter 1, *Digital Images*.



**Note** IMAQ Vision images have a default border size of 3. This border size enables you to use structuring elements as large as  $7 \times 7$  without any modification. If you plan to use structuring elements larger than  $7 \times 7$ , specify a correspondingly larger border when creating your image.

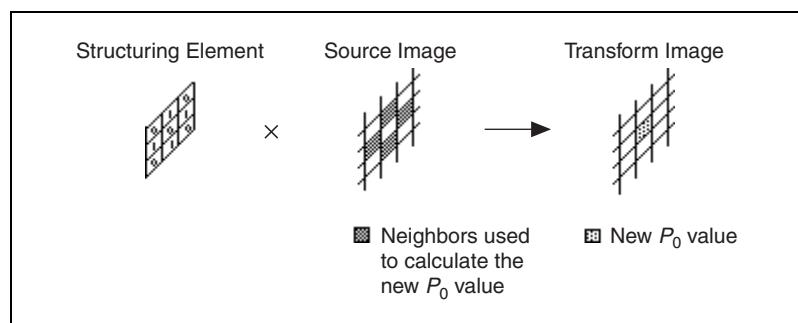
The size of the structuring element determines the speed of the morphological transformation. The smaller the structuring element, the faster the transformation.

## Structuring Element Values

The binary values of a structuring element determine which neighborhood pixels to consider during a transformation in the following manner:

- If the value of a structuring element sector is 1, the value of the corresponding source image pixel affects the central pixel's value during a transformation.
- If the value of a structuring element sector is 0, the morphological function disregards the value of the corresponding source image pixel.

Figure 9-2 illustrates the effect of structuring element values during a morphological function. A morphological transformation using a structuring element alters a pixel  $P_0$  so that it becomes a function of its neighboring pixel values.



**Figure 9-2.** Effect of Structuring Element Values on a Morphological Function

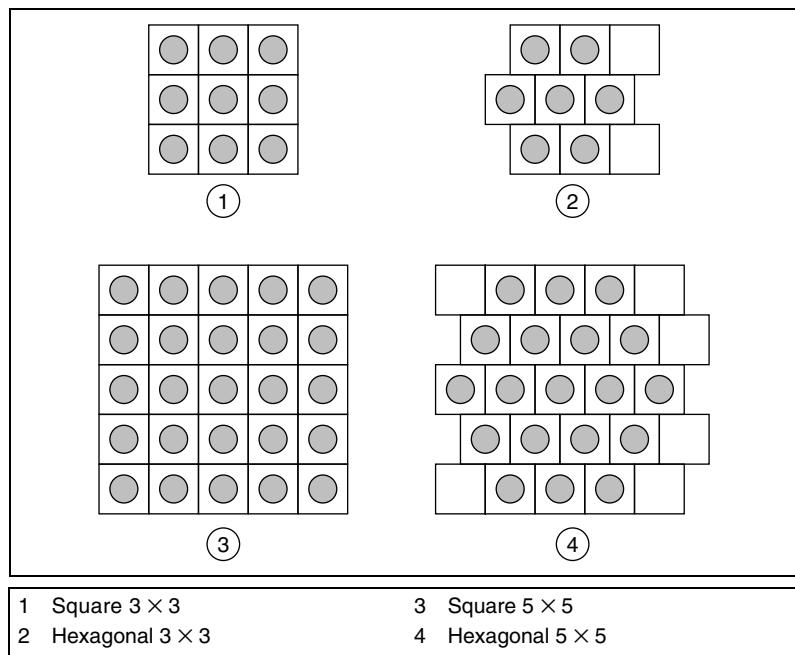
## Pixel Frame Shape

A digital image is a 2D array of pixels arranged in a rectangular grid. Morphological transformations that extract and alter the structure of particles allow you to process pixels in either a square or hexagonal configuration. These pixel configurations introduce the concept of a pixel frame. Pixel frames can either be aligned (square) or shifted (hexagonal). The pixel frame parameter is important for functions that alter the value of pixels according to the intensity values of their neighbors. Your decision to use a square or hexagonal frame affects how IMAQ Vision analyzes the image when you process it with functions that use this frame concept. IMAQ Vision uses the square frame by default.



**Note** Pixels in the image do not physically shift in a horizontal pixel frame. Functions that allow you to set the pixel frame shape merely process the pixel values differently when you specify a hexagonal frame.

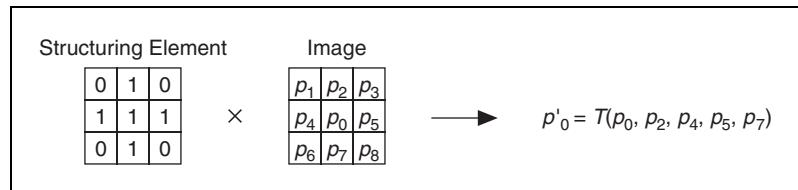
Figure 9-3 illustrates the difference between a square and hexagonal pixel frame when a  $3 \times 3$  and a  $5 \times 5$  structuring element are applied.



**Figure 9-3.** Square and Hexagonal Pixel Frames

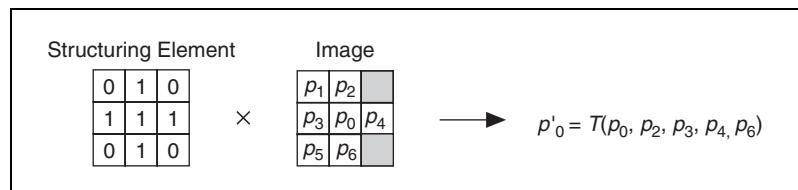
If a morphological function uses a  $3 \times 3$  structuring element and a hexagonal frame mode, the transformation does not consider the elements [2, 0] and [2, 2] when calculating the effect of the neighbors on the pixel being processed. If a morphological function uses a  $5 \times 5$  structuring element and a hexagonal frame mode, the transformation does not consider the elements [0, 0], [4, 0], [4, 1], [4, 3], [0, 4] and [4, 4].

Figure 9-4 illustrates a morphological transformation using a  $3 \times 3$  structuring element and a rectangular frame mode.



**Figure 9-4.** Transformation Using a  $3 \times 3$  Structuring Element and Rectangular Frame

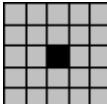
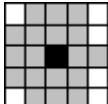
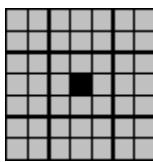
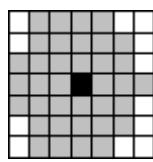
Figure 9-5 illustrates a morphological transformation using a  $3 \times 3$  structuring element and a hexagonal frame mode.



**Figure 9-5.** Transformation Using a  $3 \times 3$  Structuring Element and Hexagonal Frame

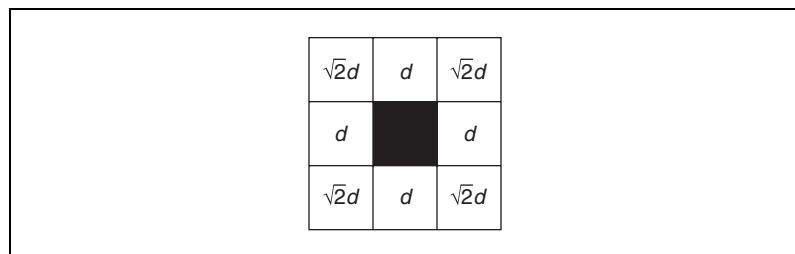
Table 9-1 illustrates the effect of the pixel frame shape on a neighborhood given three structuring element sizes. The gray boxes indicate the neighbors of each black center pixel.

**Table 9-1.** Pixel Neighborhoods Based on Pixel Frame Shapes

Structuring Element Size	Square Pixel Frame	Hexagonal Pixel Frame
$3 \times 3$		
$5 \times 5$		
$7 \times 7$		

## Square Frame

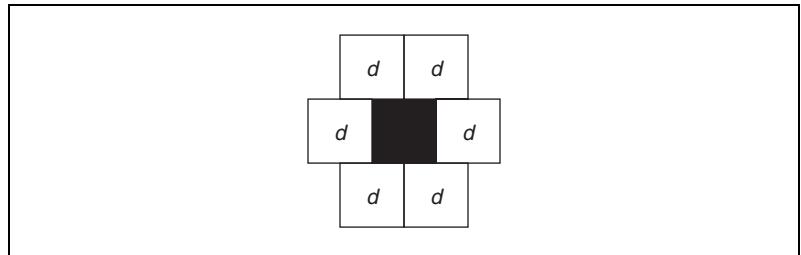
In a square frame, pixels line up as they normally do. Figure 9-6 shows a pixel in a square frame surrounded by its eight neighbors. If  $D$  is the distance from the vertical and horizontal neighbors to the central pixel, then the diagonal neighbors are at a distance of  $\sqrt{2}D$  from the central pixel.



**Figure 9-6.** Square Frame

## Hexagonal Frame

In a hexagonal frame, the even lines of an image shift half a pixel to the right. Therefore, the hexagonal frame places the pixels in a configuration similar to a true circle. Figure 9-7 shows a pixel in a hexagonal frame surrounded by its six neighbors. Each neighbor is an equal distance  $d$  from the central pixel, which results in highly precise morphological measurements.



**Figure 9-7.** Hexagonal Frame

## Connectivity

---

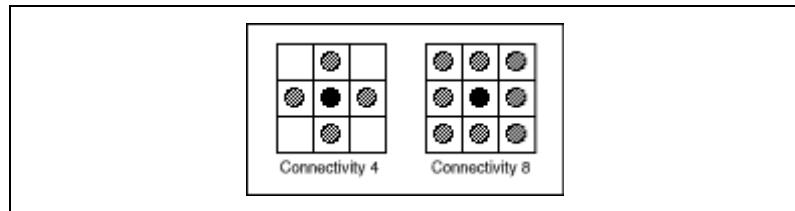
After you identify the pixels belonging to a specified intensity threshold, IMAQ Vision groups them into particles. This process introduces adjacent pixels, or *connectivity*. In some functions, you can set the pixel connectivity to specify how IMAQ Vision determines whether two adjacent pixels are included in the same particle.

### When to Use

If your binary image contains particles that touch at one point, use connectivity-4 to ensure that IMAQ Vision recognizes each particle. If you have particles that contain narrow areas, use connectivity-8. Once you select a connectivity setting, you should use the same connectivity setting throughout your application.

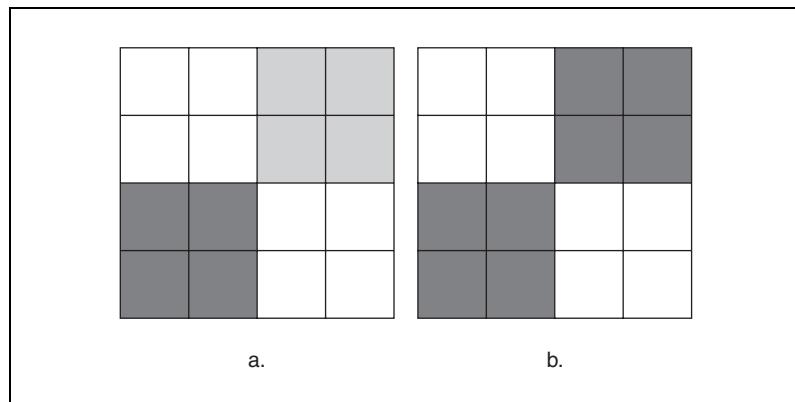
### Connectivity Concepts

With connectivity-4, two pixels are considered part of the same particle if they are horizontally or vertically adjacent. With connectivity-8, two pixels are considered part of the same particle if they are horizontally, vertically, or diagonally adjacent. Figure 9-8 illustrates the two types of connectivity.



**Figure 9-8.** Connectivity Types

Figure 9-9 illustrates how connectivity-4 and connectivity-8 affect the way the number of particles in an image are determined. In Figure 9-9a, the image has two particles with connectivity-4. In Figure 9-9b, the same image has one particle with connectivity-8.



**Figure 9-9.** Example of Connectivity Processing

The advanced morphology VIs IMAQ RemoveParticle, IMAQ RejectBorder, IMAQ FillHole, and IMAQ ParticleFilter use the input **Connectivity 4/8** (default is 8) to determine whether a neighboring pixel is considered to be part of same particle.

## In-Depth Discussion

In a rectangular pixel frame, each pixel  $P_0$  has eight neighbors, as shown in the following graphic. From a mathematical point of view, the pixels  $P_1, P_3, P_5, P_7$  are closer to  $P_0$  than the pixels  $P_2, P_4, P_6$ , and  $P_8$ .

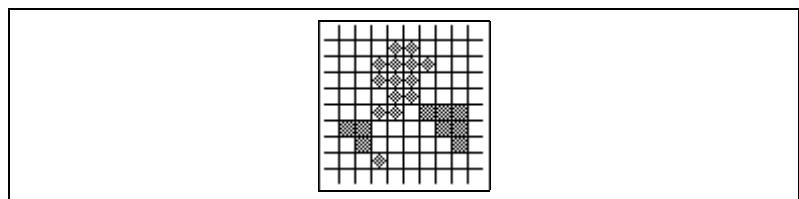
$$\begin{bmatrix} P_8 & P_1 & P_2 \\ P_7 & P_0 & P_3 \\ P_6 & P_5 & P_4 \end{bmatrix}$$

If  $D$  is the distance from  $P_0$  to  $P_1$ , then the distances between  $P_0$  and its eight neighbors can range from  $D$  to  $\sqrt{2}D$ , as shown in the following graphic.

$$\begin{bmatrix} \sqrt{2}D & D & \sqrt{2}D \\ D & 0 & D \\ \sqrt{2}D & D & \sqrt{2}D \end{bmatrix}$$

## Connectivity-4

A pixel belongs to a particle if it is located a distance of  $D$  from another pixel in the particle. In other words, two pixels are considered to be part of the same particle if they are horizontally or vertically adjacent. They are considered as part of two different particles if they are diagonally adjacent. In Figure 9-10, the particle count equals 4.



**Figure 9-10.** Connectivity-4

## Connectivity-8

A pixel belongs to a particle if it is located a distance of  $D$  or  $\sqrt{2}D$  from another pixel in the particle. In other words, two pixels are considered to be part of the same particle if they are horizontally, vertically, or diagonally adjacent. In Figure 9-11, the particle count equals 1.

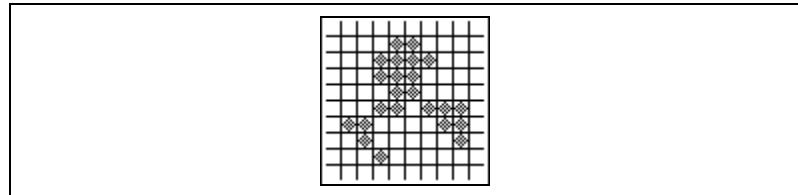


Figure 9-11. Connectivity-8

## Primary Morphology Operations

Primary morphological operations work on binary images to process each pixel based on its neighborhood. Each pixel is set either to 1 or 0, depending on its neighborhood information and the operation used. These operations always change the overall size and shape of the particles in the image.

### When to Use

Use the primary morphological operations for expanding or reducing particles, smoothing the borders of objects, finding the external and internal boundaries of particles, and locating particular configurations of pixels.

You can also use these transformations to prepare particles for quantitative analysis, to observe the geometry of regions, and to extract the simplest forms for modeling and identification purposes.

### Primary Morphology Concepts

The *primary morphology* functions apply to binary images in which particles have been set to 1 and the background is equal to 0. They include three fundamental binary processing functions—erosion, dilation, and hit-miss. The other transformations are combinations of these three functions.

This section describes the following primary morphology transformations:

- Erosion
- Dilation
- Opening
- Closing
- Inner gradient
- Outer gradient

- Hit-miss
- Thinning
- Thickening
- Proper-opening
- Proper-closing
- Auto-median



**Note** In the following descriptions, the term *pixel* denotes a pixel equal to 1, and the term *particle* denotes a group of pixels equal to 1.

## Erosion and Dilation Functions

An *erosion* eliminates pixels isolated in the background and erodes the contour of particles according to the template defined by the structuring element.

For a given pixel  $P_0$ , the structuring element is centered on  $P_0$ . The pixels masked by a coefficient of the structuring element equal to 1 are then referred as  $P_i$ .

- *If* the value of one pixel  $P_i$  is equal to 0, *then*  $P_0$  is set to 0, *else*  $P_0$  is set to 1.
- *If*  $\text{AND}(P_i) = 1$ , *then*  $P_0 = 1$ , *else*  $P_0 = 0$ .

A *dilation* eliminates tiny holes isolated in particles and expands the contour of the particles according to the template defined by the structuring element. This function has the reverse effect of an erosion, because the dilation is equivalent to eroding the background.

For a given pixel  $P_0$ , the structuring element is centered on  $P_0$ . The pixels masked by a coefficient of the structuring element equal to 1 then are referred to as  $P_i$

- *If* the value of one pixel  $P_i$  is equal to 1, *then*  $P_0$  is set to 1, *else*  $P_0$  is set to 0.
- *If*  $\text{OR}(P_i) = 1$ , *then*  $P_0 = 1$ , *else*  $P_0 = 0$ .

Figure 9-12 illustrates the effects of erosion and dilation. Figure 9-12a is the binary source image. Figure 9-12b represents the source image after erosion, and Figure 9-12c shows the source image after dilation.

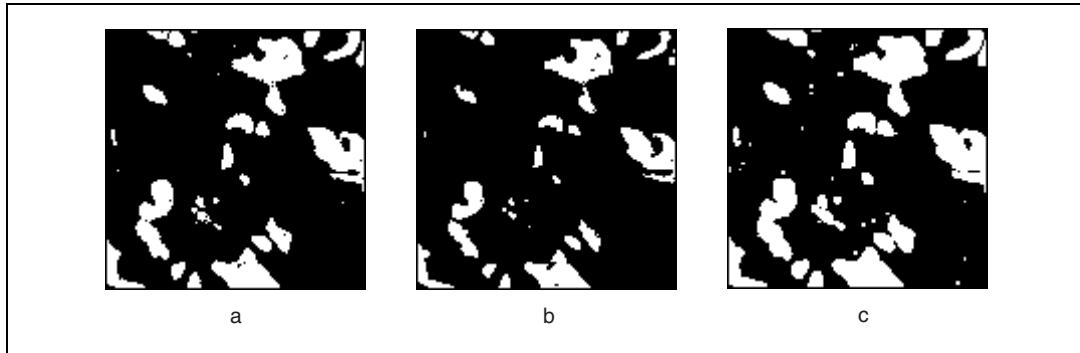
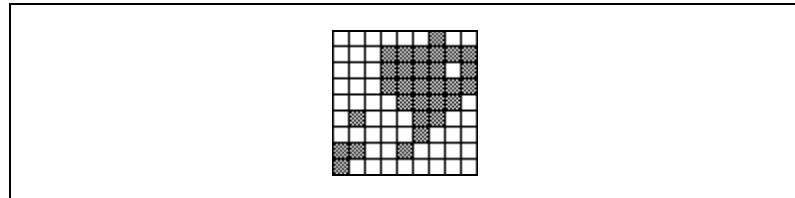
**Figure 9-12.** Erosion and Dilation Functions

Figure 9-13 is the source image for the examples in Tables 9-2 and 9-3, in which gray cells indicate pixels equal to 1.

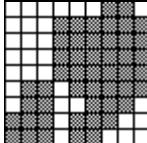
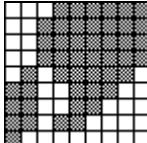
**Figure 9-13.** Source Image Before Erosion and Dilation

Tables 9-2 and 9-3 show how the structuring element can control the effects of erosion or dilation, respectively. The larger the structuring element, the more templates can be edited and the more selective the effect.

**Table 9-2.** How the Structure Element Affects Erosion

Structuring Element	After Erosion	Description
		A pixel is cleared if it is equal to 1 and if its three upper-left neighbors do not equal 1. The erosion truncates the upper-left borders of the particles.
		A pixel is cleared if it is equal to 1 and if its lower and right neighbors do not equal 1. The erosion truncates the bottom and right borders of the particles but retains the corners.

**Table 9-3.** How the Structure Element Affects Dilation

Structuring Element	After Dilation	Description
		A pixel is set to 1 if it is equal to 1 or if one of its three upper-left neighbors equals 1. The dilation expands the lower-right borders of the particles.
		A pixel is set to 1 if it is equal to 1 or if its lower or right neighbor equals 1. The dilation expands the upper and left borders of the particles.

## Opening and Closing Functions

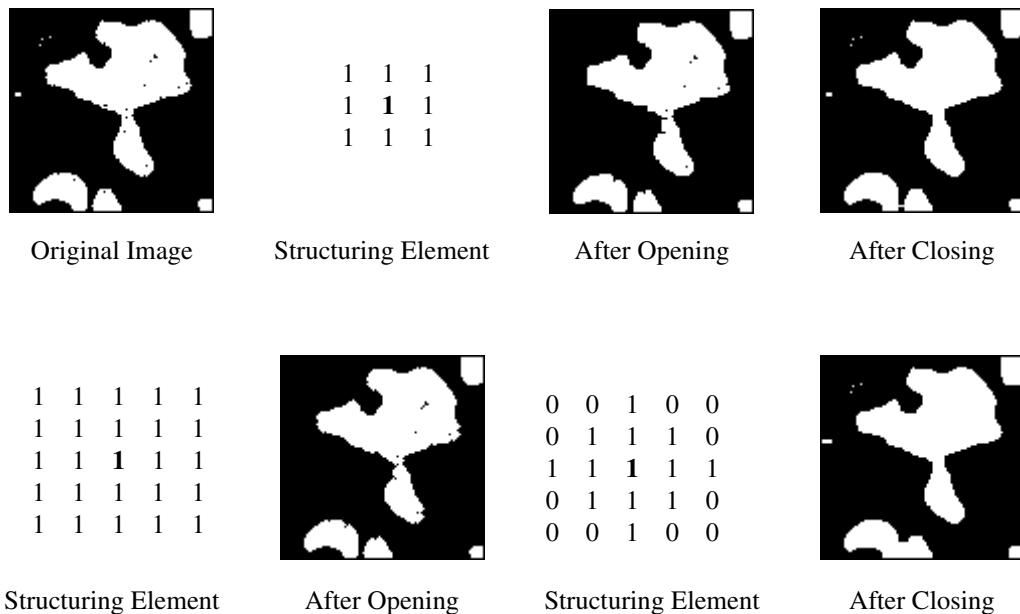
The *opening function* is an erosion followed by a dilation. This function removes small particles and smooths boundaries. This operation does not significantly alter the area and shape of particles because erosion and dilation are *dual transformations*, in which borders removed by the erosion process are restored during dilation. However, small particles eliminated during the erosion are not restored by the dilation. If  $I$  is an image,

$$\text{opening}(I) = \text{dilation}(\text{erosion}(I))$$

The *closing function* is a dilation followed by an erosion. This function fills tiny holes and smooths boundaries. This operation does not significantly alter the area and shape of particles because dilation and erosion are *morphological complements*, where borders expanded by the dilation function are then reduced by the erosion function. However, erosion does not restore any tiny holes filled during dilation. If  $I$  is an image,

$$\text{closing}(I) = \text{erosion}(\text{dilation}(I))$$

The following figures illustrate examples of the opening and closing function.

**Figure 9-14.** Opening and Closing Functions

## Inner Gradient Function

The *internal edge* subtracts the eroded image from its source image. The remaining pixels correspond to the pixels eliminated by the erosion process. If  $I$  is an image,

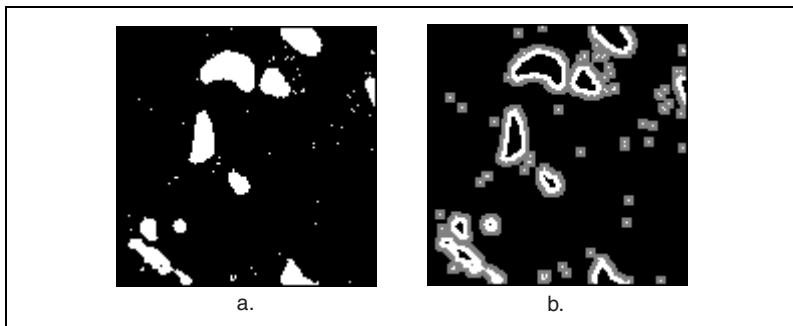
$$\text{internal edge}(I) = I - \text{erosion}(I) = \text{XOR}(I, \text{erosion}(I))$$

## Outer Gradient Function

The *external edge* subtracts the source image from the dilated image of the source image. The remaining pixels correspond to the pixels added by the dilation process. If  $I$  is an image,

$$\text{external edge}(I) = \text{dilation}(I) - I = \text{XOR}(I, \text{dilation}(I))$$

Figure 9-15a shows the binary source image. Figure 9-15b shows the image produced from an extraction using a  $5 \times 5$  structuring element. The superimposition of the internal edge is shown in white, and the external edge is shown in gray. The thickness of the extended contours depends on the size of the structuring element.

**Figure 9-15.** External Edges

## Hit-Miss Function

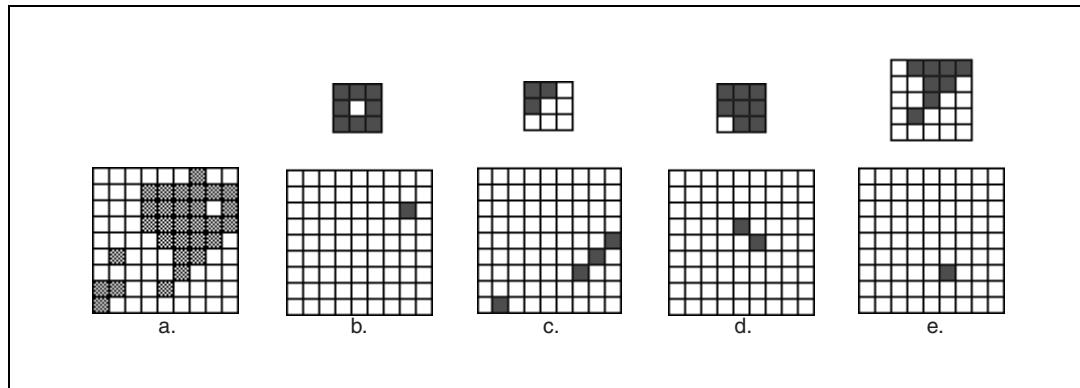
Use the [hit-miss function](#) to locate particular configurations of pixels. This function extracts each pixel located in a neighborhood exactly matching the template defined by the structuring element. Depending on the configuration of the structuring element, the hit-miss function can locate single isolated pixels, cross-shape or longitudinal patterns, right angles along the edges of particles, and other user-specified shapes. The larger the size of the structuring element, the more specific the researched template can be. See Table 9-4 for strategies on using the hit-miss function.

In a structuring element with a central coefficient equal to 0, a hit-miss function changes all pixels set to 1 in the source image to the value 0.

For a given pixel  $P_0$ , the structuring element is centered on  $P_0$ . The pixels masked by the structuring element are then referred to as  $P_i$ .

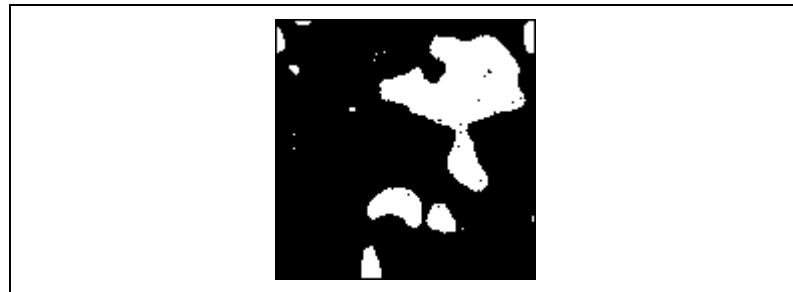
- If the value of each pixel  $P_i$  is equal to the coefficient of the structuring element placed on top of it, then the pixel  $P_0$  is set to 1, else the pixel  $P_0$  is set to 0.
- In other words, if the pixels  $P_i$  define the exact same template as the structuring element, then  $P_0 = 1$ , else  $P_0 = 0$ .

Figures 9-16b, 9-16c, and 9-16d show the result of three hit-miss functions applied to the same source image, represented in Figure 9-16a. Each hit-miss function uses a different structuring element, which is specified above each transformed image. Gray cells indicate pixels equal to 1.



**Figure 9-16.** Hit-Miss Function

A second example of the hit-miss function shows how, when given the binary image shown in Figure 9-17, the function can locate various patterns specified in the structuring element. The results are displayed in Table 9-4.



**Figure 9-17.** Binary Image Before Application of Hit and Miss Function

**Table 9-4.** Using the Hit-Miss Function

Strategy	Structuring Element	Resulting Image
Use the hit-miss function to locate pixels isolated in a background.  The structuring element on the right extracts all pixels equal to 1 that are surrounded by at least two layers of pixels that are equal to 0.	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 </pre>	
Use the hit-miss function to locate single pixel holes in particles.  The structuring element on the right extracts all pixels equal to 0 that are surrounded by at least one layer of pixels that are equal to 1.	<pre> 1 1 1 1 0 1 1 1 1 </pre>	
Use the hit-miss function to locate pixels along a vertical left edge.  The structuring element on the right extracts pixels surrounded by at least one layer of pixels equal to 1 to the left and pixels that are equal to 0 to the right.	<pre> 1 1 0 1 1 0 1 1 0 </pre>	

## Thinning Function

The *thinning function* eliminates pixels that are located in a neighborhood matching a template specified by the structuring element. Depending on the configuration of the structuring element, you can also use thinning to remove single pixels isolated in the background and right angles along the edges of particles. The larger the size of the structuring element, the more specific the template can be.

The thinning function extracts the intersection between a source image and its transformed image after a hit-miss function. In binary terms, the operation subtracts its hit-miss transformation from a source image.

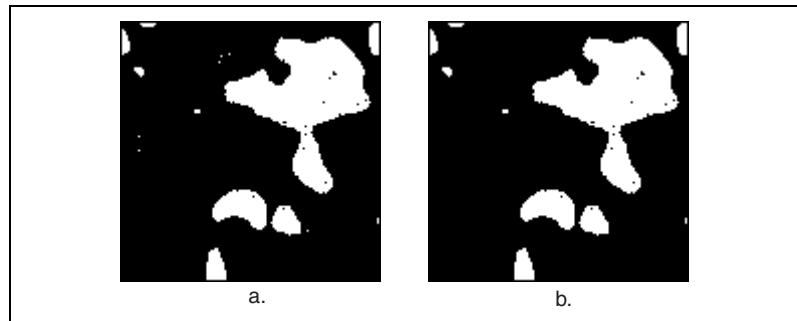
Do not use this function when the central coefficient of the structuring element is equal to 0. In such cases, the hit-miss function can only change the value of certain pixels in the background from 0 to 1. However, the subtraction of the thinning function then resets these pixels back to 0 anyway.

If  $I$  is an image,

$$\text{thinning}(I) = I - \text{hit-miss}(I) = \text{XOR}(I, \text{hit-miss}(I))$$

Figure 9-18a shows the binary source image used in the following example of thinning. Figure 9-18b illustrates the resulting image, in which single pixels in the background are removed from the image. This example uses the following structuring element:

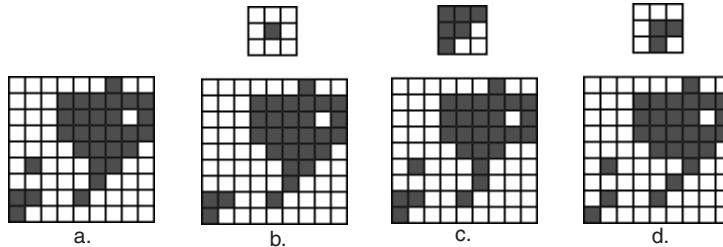
$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$



**Figure 9-18.** Thinning Function

Another thinning example uses the source image shown in Figure 9-19a. Figures 9-19b through 9-19d show the results of three thinnings applied to the source image. Each thinning uses a different structuring element, which

is specified above each transformed image. Gray cells indicate pixels equal to 1.



**Figure 9-19.** Thinning Function with Structuring Elements

## Thickening Function

The *thickening function* adds to an image those pixels located in a neighborhood that matches a template specified by the structuring element. Depending on the configuration of the structuring element, you can use thickening to fill holes and smooth right angles along the edges of particles. A larger structuring element allows for a more specific template.

The thickening function extracts the union between a source image and its transformed image, which was created by a hit-miss function using a structuring element specified for thickening. In binary terms, the operation adds a hit-miss transformation to a source image.

Do not use this function when the central coefficient of the structuring element is equal to 1. In such cases, the hit-miss function can only turn certain pixels of the particles from 1 to 0. However, the addition of the thickening function resets these pixels to 1 anyway.

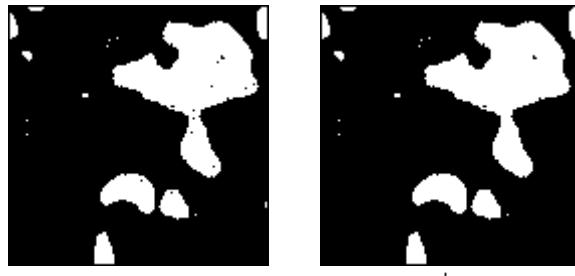
If  $I$  is an image,

$$\text{thickening}(I) = I + \text{hit-miss}(I) = \text{OR}(I, \text{hit-miss}(I))$$

Figure 9-20a represents the binary source file used in the following thickening example. Figure 9-20b shows the result of the thickening

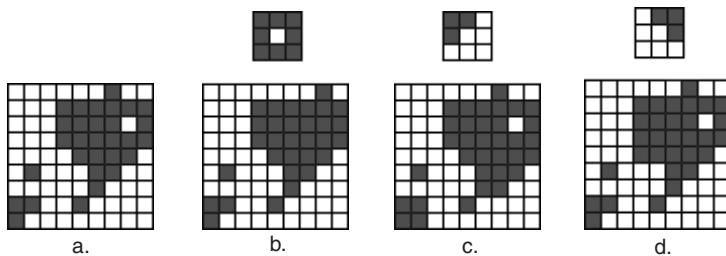
function applied to the source image, which filled single pixel holes using the following structuring element:

$$\begin{matrix} 1 & 1 & 1 \\ 1 & \mathbf{0} & 1 \\ 1 & 1 & 1 \end{matrix}$$



**Figure 9-20.** Thickening Function

Figure 9-21a shows the source image for another thickening example. Figures 9-21b through 9-21d illustrate the results of three thickenings applied to the source image. Each thickening uses a different structuring element which is specified on top of each transformed image. Gray cells indicate pixels equal to 1.



**Figure 9-21.** Thickening Function with Different Structuring Elements

## Proper-Opening Function

The *proper-opening function* is a finite and dual combination of openings and closings. It removes small particles and smooths the contour of particles according to the template defined by the structuring element.

If  $I$  is the source image, the proper-opening function extracts the intersection between the source image  $I$  and its transformed image obtained after an opening, followed by a closing, and then followed by another opening.

$$\begin{aligned} \text{proper-opening}(I) &= \text{AND}(I, OCO(I)) \text{ or} \\ \text{proper-opening}(I) &= \text{AND}(I, DEEDDE(I)) \end{aligned}$$

where  $I$  is the source image,

$E$  is an erosion,

$D$  is a dilation,

$O$  is an opening,

$C$  is a closing,

$F(I)$  is the image obtained after applying the function  $F$  to the image  $I$ , and

$GF(I)$  is the image obtained after applying the function  $F$  to the image  $I$  followed by the function  $G$  to the image  $I$ .

## Proper-Closing Function

The *proper-closing function* is a finite and dual combination of closings and openings. It fills tiny holes and smooths the inner contour of particles according to the template defined by the structuring element.

If  $I$  is the source image, the proper-closing function extracts the union of the source image  $I$  and its transformed image obtained after a closing, followed by an opening, and then followed by another closing.

$$\begin{aligned} \text{proper-closing}(I) &= \text{OR}(I, COC(I)) \text{ or} \\ \text{proper-closing}(I) &= \text{OR}(I, EDDEED(I)) \end{aligned}$$

where  $I$  is the source image,

$E$  is an erosion,

$D$  is a dilation,

$O$  is an opening,

$C$  is a closing,

$F(I)$  is the image obtained after applying the function  $F$  to the image  $I$ , and

$GF(I)$  is the image obtained after applying the function  $F$  to the image  $I$  followed by the function  $G$  to the image  $I$ .

## Auto-Median Function

The *auto-median function* uses dual combinations of openings and closings. It generates simpler particles that contain fewer details.

If  $I$  is the source image, the auto-median function extracts the intersection between the proper-opening and proper-closing of the source image  $I$ .

$$\begin{aligned} \text{auto-median}(I) &= \text{AND}(\text{OCO}(I), \text{COC}(I)) \text{ or} \\ \text{auto-median}(I) &= \text{AND}(\text{DEEDDE}(I), \text{EDDEED}(I)) \end{aligned}$$

where  $I$  is the source image,

$E$  is an erosion,

$D$  is a dilation,

$O$  is an opening,

$C$  is a closing,

$F(I)$  is the image obtained after applying the function  $F$  to the image  $I$ , and

$GF(I)$  is the image obtained after applying the function  $F$  to the image  $I$  followed by the function  $G$  to the image  $I$ .

## Advanced Morphology Operations

---

The advanced morphology operations are built upon the primary morphological operators and work on particles as opposed to pixels in the image. Each of the operations have been developed to perform specific operations on the particles in a binary image.

### When to Use

Use the advanced morphological operations for filling holes in particles, removing particles that touch the border of the image, remove unwanted small and large particles, separate touching particles, finding the convex hull of particles, and more.

You can use these transformations to prepare particles for quantitative analysis, observe the geometry of regions, extract the simplest forms for modeling and identification purposes, and so forth.

## Advanced Morphology Transforms Concepts

The advanced morphology functions are conditional combinations of fundamental transformations, such as binary erosion and dilation. The functions apply to binary images in which a threshold of 1 has been applied to particles and where the background is equal to 0. This section describes the following advanced binary morphology functions:

- Border
- Hole filling
- Labeling
- Lowpass filters
- Highpass filters
- Separation
- Skeleton
- Segmentation
- Distance
- Danielsson
- Circle
- Convex



**Note** In this section of the manual, the term *pixel* denotes a pixel equal to 1, and the term *particle* denotes a group of pixels equal to 1.

### Border Function

The [border function](#) removes particles that touch the border of the image. These particles may have been truncated during the digitization of the image, and eliminating them helps to avoid erroneous particle measurements and statistics.

### Hole Filling Function

The [hole filling function](#) fills the holes within particles.

### Labeling Function

The [labeling function](#) assigns a different gray-level value to each particle. The image produced is not a binary image, but a labeled image using a number of gray-level values equal to the number of particles in the image plus the gray level 0 used in the background area.

The labeling function identifies particles using either connectivity-4 or connectivity-8 criteria. For more information on connectivity, see the [Connectivity](#) section of this chapter.

## Lowpass and Highpass Filters

The [lowpass filter](#) removes small particles according to their widths as specified by a parameter called [filter size](#). For a given filter size  $N$ , the lowpass filter eliminates particles whose widths are less than or equal to  $(N - 1)$  pixels. These particles would disappear after  $(N - 1)/2$  erosions.

The [highpass filter](#) removes large particles according to their widths as specified by a parameter called filter size. For a given filter size  $N$ , the highpass filter eliminates particles with widths greater than or equal to  $N$  pixels. These particles would not disappear after  $(N/2 + 1)$  erosions.

Both the highpass and lowpass morphological filters use erosions to select particles for removal. Since erosions or filters cannot discriminate particles with widths of  $2k$  pixels from particles with widths of  $2k - 1$  pixels, a single erosion eliminates both particles that are 2 pixels wide and 1 pixel wide.

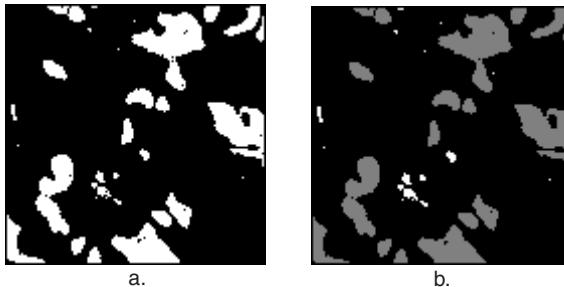
Table 9-5 shows the effect of lowpass and highpass filtering.

**Table 9-5.** Effect of Lowpass and Highpass Filtering

Filter Size ( $N$ )	Highpass Filter	Lowpass Filter
<b><math>N</math> is an even number (<math>N = 2k</math>)</b>	<ul style="list-style-type: none"> <li>Removes particles with a width greater than or equal to <math>2k</math></li> <li>Uses <math>k - 1</math> erosions</li> </ul>	<ul style="list-style-type: none"> <li>Removes particles with a width less than or equal to <math>2k - 2</math></li> <li>Uses <math>k - 1</math> erosions</li> </ul>
<b><math>N</math> is an odd number (<math>N = 2k + 1</math>)</b>	<ul style="list-style-type: none"> <li>Removes particles with a width greater than or equal to <math>2k + 1</math></li> <li>Uses <math>k</math> erosions</li> </ul>	<ul style="list-style-type: none"> <li>Removes particles with a width less than or equal to <math>2k</math></li> <li>Uses <math>k</math> erosions</li> </ul>

Figure 9-22a shows the binary source image used in this example. Figure 9-22b shows how, for a given filter size, a highpass filter produces

the following image. Gray particles and white particles are filtered out by a lowpass and highpass filter, respectively.



**Figure 9-22.** Highpass and Lowpass Filter Functions

## Separation Function

The *separation function* breaks narrow isthmuses and separates touching particles with respect to a user-specified filter size. This operation uses erosions, labeling, and conditional dilations.

For example, after thresholding an image, two gray-level particles overlapping one another might appear as a single binary particle. You can observe narrowing where the original particles intersected each other. If the narrowing has a width of  $M$  pixels, a separation using a filter size of  $(M + 1)$  breaks it and restores the two original particles. This applies to all particles that contain a narrowing shorter than  $N$  pixels.

For a given filter size  $N$ , the separation function segments particles with a narrowing shorter than or equal to  $(N - 1)$  pixels. These particles are divided into two parts after  $(N - 1)/2$  erosions.

The above definition is true when  $N$  is an odd number, but should be modified slightly when  $N$  is an even number, due to the use of erosions in determining whether a narrowing should be broken or kept. The function cannot discriminate a narrowing with a width of  $2k$  pixels from a narrowing with a width of  $(2k - 1)$  pixels, therefore, one erosion breaks both a narrowing that is 2 pixels wide as well as a narrowing that is 1 pixel wide.

The precision of the separation is limited to the elimination of constrictions that have a width smaller than an even number of pixels:

- If  $N$  is an even number ( $2k$ ), the separation breaks a narrowing with a width smaller than or equal to  $(2k - 2)$  pixels. It uses  $(k - 1)$  erosions.
- If  $N$  is an odd number ( $2k + 1$ ), the separation breaks a narrowing with a width smaller than or equal to  $2k$ . It uses  $k$  erosions.

## Skeleton Functions

A *skeleton function* applies a succession of thinnings until the width of each particle becomes equal to 1 pixel. The skeleton functions are both time- and memory-consuming. They are based on conditional applications of thinnings and openings that various configurations of structuring elements.

Skeleton L uses this type of structuring element:

$$\begin{matrix} 0 & ? & 1 \\ 0 & 1 & 1 \\ 0 & ? & 1 \end{matrix}$$

Skeleton M uses this type of structuring element:

$$\begin{matrix} ? & ? & 1 \\ 0 & 1 & 1 \\ ? & ? & 1 \end{matrix}$$

Skiz is an inverse skeleton (Skeleton L on an inverse image).

## L-Skeleton Function

The *L-skeleton function* indicates the L-shaped structuring element skeleton function. Notice the original image in Figure 9-23a, and how the L-skeleton function produces the rectangle pixel frame image in Figure 9-23b.

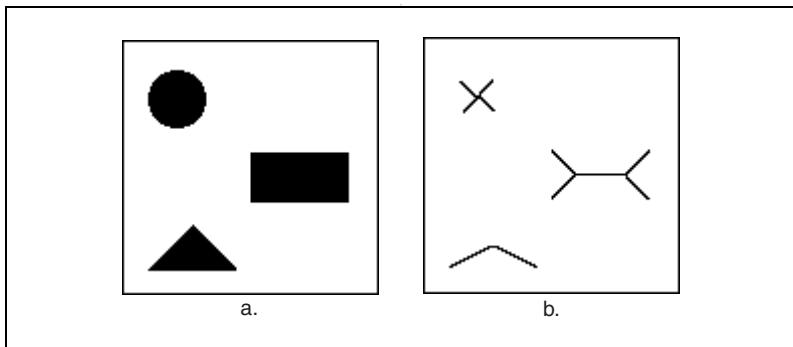


Figure 9-23. L-Skeleton Function

### M-Skeleton Function

The *M-skeleton* (M-shaped structuring element) function extracts a skeleton with more dendrites or branches. Using the same original image from Figure 9-23a, the M-skeleton function produces the image shown in Figure 9-24.

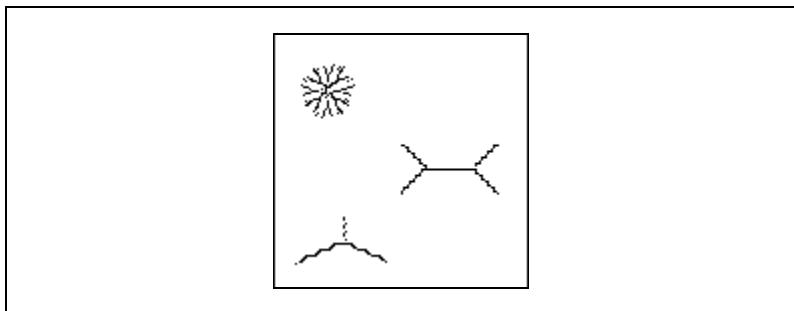


Figure 9-24. M-Skeleton Function

### Skiz Function

The *skiz* (skeleton of influence zones) function behaves like an L-skeleton applied to the background regions instead of the particle regions. It produces median lines that are at an equal distance from the particles.

Using the source image from Figure 9-23a, the skiz function produces the following image, which is shown superimposed on the source image.

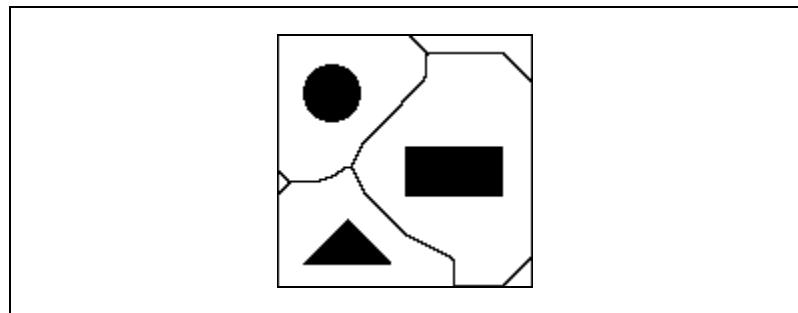


Figure 9-25. Skiz Function

## Segmentation Function

The *segmentation function* is applied only to labeled images. It partitions an image into segments that are centered on a particle such that they do not overlap each other or leave empty zones. Empty zones are caused by dilating particles until they touch one another.



**Note** The segmentation function is time-consuming. Reduce the image to its minimum significant size before selecting this function.

In Figure 9-26, binary particles (shown in black) are superimposed on top of the segments (shown in gray shades).

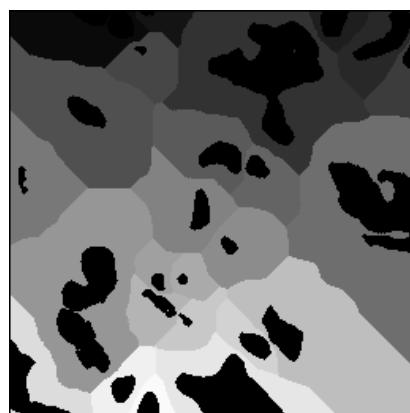


Figure 9-26. Segmentation Function

When applied to an image with binary particles, the transformed image turns completely red because it is entirely composed of pixels set to 1.

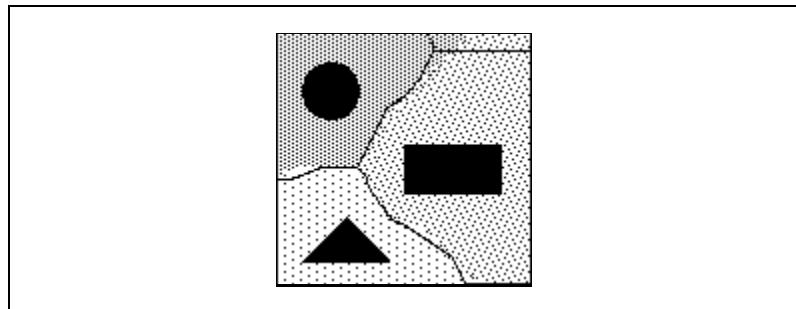
## Comparisons Between Segmentation and Skiz Functions

The segmentation function extracts segments that each contain one particle and represent the area in which this particle can be moved without intercepting another particle (assuming that all particles move at the same speed).

The edges of these segments give a representation of the external skeletons of the particles. Unlike the skiz function, segmentation does not involve median distances.

You can obtain segments using successive dilations of particles until they touch each other and cover the entire image. The final image contains as many segments as there were particles in the original image. However, if you consider the inside of closed skiz lines as segments, you may produce more segments than particles originally present in the image. Consider the upper-right region in Figure 9-27. This image shows:

- Original particles in black
- Segments in dotted patterns
- Skiz lines



**Figure 9-27.** Segmentation with Skiz Lines

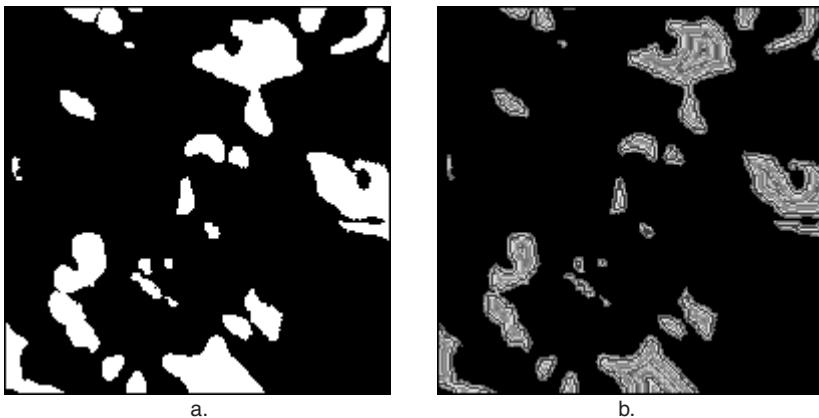
## Distance Function

The *distance function* assigns a gray-level value to each pixel equal to the shortest distance to the border of the particle. That distance may be equal to the distance to the outer border of the particle or to a hole within the particle.

## Danielsson Function

The *Danielsson function* also creates a distance map but is a more accurate algorithm than the classical distance function. Use the Danielsson function instead of the distance function when possible.

Figure 9-28a shows the source threshold image used in the following example. The image is sequentially processed with a lowpass filter, hole filling, and the Danielsson function. The Danielsson function produces the following distance map image, shown in Figure 9-28b.



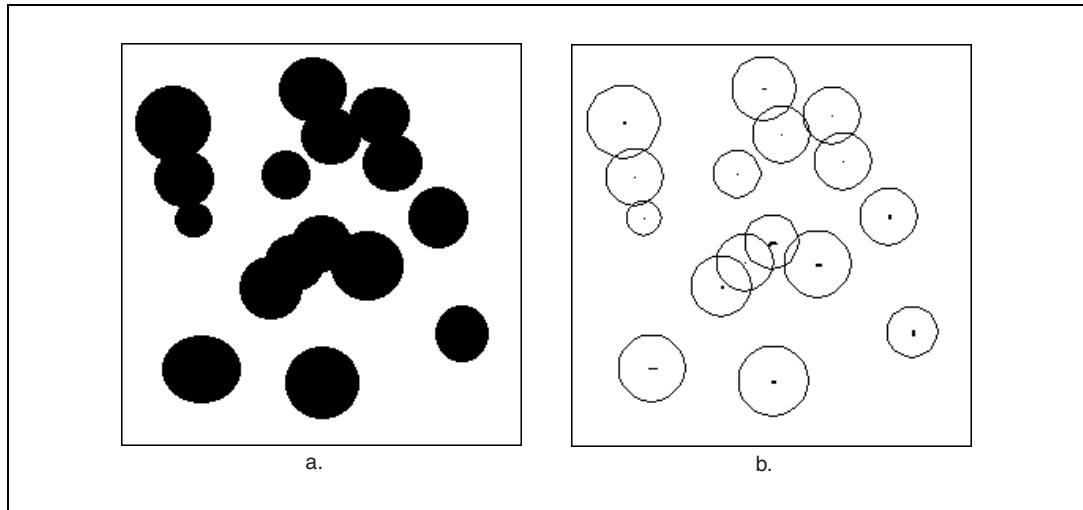
**Figure 9-28.** Danielsson Function

View this final image with a binary palette. In this case, each level corresponds to a different color. You can easily determine the relation of a set of pixels to the border of a particle. The first layer, which forms the border, is red. The second layer, closest to the border, is green, the third layer is blue, and so forth.

## Circle Function

The *circle* function allows you to separate overlapping circular particles. The circle function uses the Danielsson coefficient to reconstitute the form of an particle, provided that the particles are essentially circular. The particles are treated as a set of overlapping discs that are then separated into separate discs. This allows you to trace circles corresponding to each particle.

Figure 9-29a illustrates the source image for the following example of the circle function. Figure 9-29b illustrates the processed image.



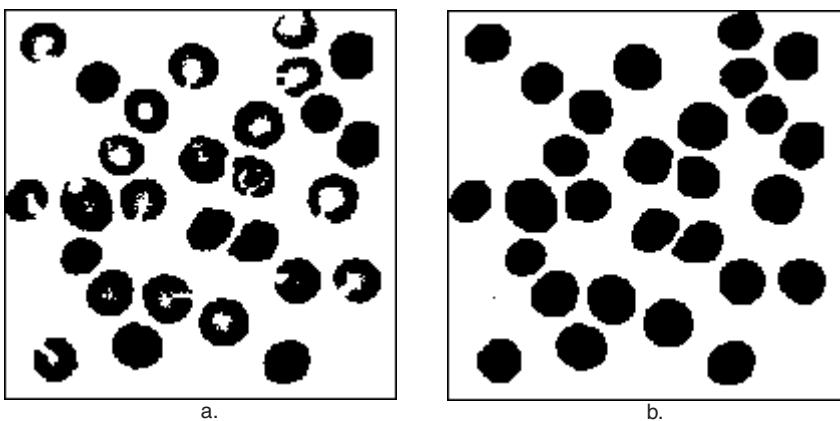
**Figure 9-29.** Circle Function

## Convex Function

The *convex function* is useful for closing particles so that measurements can be made on the particle, even though the contour of the particle is discontinuous. This command is usually necessary when the sample particle is cut because of the acquisition process.

The convex function calculates a convex envelope around the perimeter of each particle, effectively closing the particle. The image to be treated must be both binary and labeled.

Figure 9-30a represents the original binary labeled image used in this example. Figure 9-30b shows the results after the convex function is applied to the image.



**Figure 9-30.** Convex Function

# Particle Measurements

This chapter contains information about the areas, lengths, coordinates, chords and axes, shape equivalence, shape features, and diverse measurements of particles.

## Digital Particles

You can characterize a digital particle by a set of morphological and intensity parameters described in the *Areas*, *Lengths*, *Coordinates*, *Chords and Axes*, *Shape Equivalence*, *Shape Features*, and *Diverse Measurements* sections.

## When to Use

Use particle measurements when you want to make shape measurements on particles in a binary image.

## Digital Particle Concepts

### Areas

This section describes the following area parameters:

- **Number of pixels**—Area of a particle, without holes, in pixel units
- **Particle area**—Area of a particle expressed in real units (based on image spatial calibration). This value is equal to **Number of pixels** when the spatial calibration is such that 1 pixel represents 1 square unit.
- **Scanned area**—Area of the entire image expressed in real units. This value is equal to the product (**Resolution X × X Step**) (**Resolution Y × Y Step**).
- **Ratio**—Ratio of the particle area to the entire image area. The percentage of the image occupied by all particles.

$$\text{Ratio} = \frac{\text{particle area}}{\text{scanned area}}$$

- **Number of holes**—Number of holes inside a particle. The software detects holes inside a particle as small as 1 pixel.
- **Holes' area**—Total area of the holes within a particle
- **Total area**—Area of a particle including the area of its holes. This value is equal to **Particle area + Holes' area**.



**Note** A particle located inside a hole of a bigger particle is identified as a separate particle. The area of a hole that contains a particle includes the area covered by the particle.

Particle #	Particle Area	Holes' Area	Total Area
Particle 1	A	B + C	A + B + C
Particle 2	D	0	D
Particle 3	E	F + G	E + F + G
Particle 4	G	0	G

## Lengths

This section describes the following length parameters:

- **Particle perimeter**—Length of the outer contour of a particle.
- **Holes' perimeter**—Sum of the perimeters of the holes within a particle. Holes' measurements becomes valuable data when studying constituents A and B such that B is occluded in A. If the image can be processed so that the B regions appear as holes in A regions after a threshold, the ratio (**Holes' area / Total area**) gives the percentage of B in A. **Holes' perimeter** gives the length of the boundary between A and B.

- **Breadth**—Distance between the left-most and right-most pixels in a particle, or  $\max(X_i) - \min(X_i)$ . Breadth is also equal to the horizontal side of the smallest horizontal rectangle containing the particle, or the difference  $\max X - \min X$ .
- **Height**—Distance between the upper-most and lower-most pixels in a particle, or  $\max(Y_i) - \min(Y_i)$ . It is also equal to the vertical side of the smallest horizontal rectangle containing the particle, or the difference  $\max Y - \min Y$ .

## Coordinates

Coordinates are expressed with respect to an origin  $(0, 0)$ , located at the upper-left corner of the image. This section describes the following coordinate parameters:

- **Center of Mass (X, Y)**—Coordinates of the center of gravity of a particle. The center of gravity of a particle composed of  $N$  pixels,  $P_i$ , is defined as the point  $G$  such that

$$\overline{OG} = \frac{1}{N} \sum_{i=1}^{i=N} \overline{OP}_i \text{ and}$$

$$\text{the center of mass } X_G = \frac{1}{N} \sum_{i=1}^{i=N} X_i$$

$X_G$  gives the average location of the central points of horizontal segments in a particle.

$$\text{The center of mass } Y_G = \frac{1}{N} \sum_{i=1}^{i=N} Y_i$$

$Y_G$  gives the average location of the central points of vertical segments in a particle.



**Note** G can be located outside a particle if the particle has a non-convex shape.

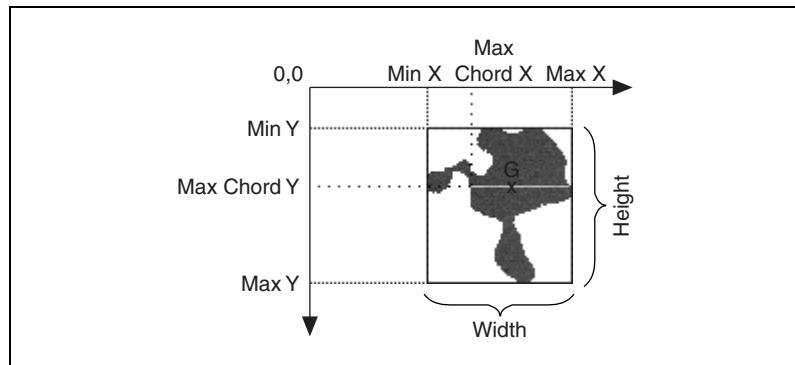
- **Min X, Min Y**—Coordinates of the upper-left and lower-right corners of the smallest horizontal rectangle containing a particle.

The origin (0, 0) has two pixels that have the coordinates ( $\text{min}X$ ,  $\text{min}Y$ ) and ( $\text{max}X$ ,  $\text{max}Y$ ) such that

$$\begin{aligned}\text{min}X &= \min(X_i) \\ \text{min}Y &= \min(Y_i) \\ \text{max}X &= \max(X_i) \\ \text{max}Y &= \max(Y_i)\end{aligned}$$

where  $X_i$  and  $Y_i$  are the coordinates of the pixels  $P_i$  in a particle.

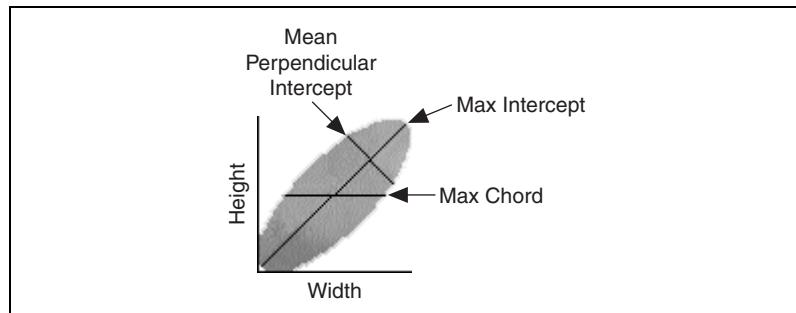
- **Max X, Max Y**—Lower-right corner of the smallest horizontal rectangle containing the particle
- **Max chord X and Y**—Coordinates of the left-most pixel along the longest horizontal chord in a particle



## Chords and Axes

This section describes the following chord and axis parameters:

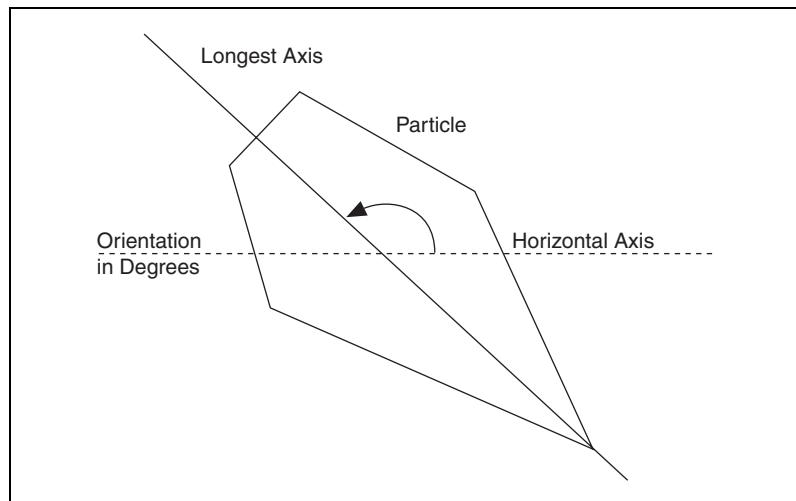
- **Max chord length**—Length of the longest horizontal chord in a particle.
- **Mean chord X**—Mean length of horizontal segments in a particle.
- **Mean chord Y**—Mean length of vertical segments in a particle.



- **Max intercept**—Length of the longest segment in the convex hull of a particle (in all possible directions of projection).
- **Mean intercept perpendicular**—Mean length of the segments in a particle perpendicular to the max intercept.

$$\text{Mean intercept perpendicular} = \frac{\text{area of the convex hull of the particle}}{\text{max intercept}}$$

- **Particle orientation**—The angle of the longest axis with respect to the horizontal axis. The value can range from  $0^\circ$  to  $180^\circ$ . Notice that this value does not provide information regarding the symmetry of the particle. Therefore, an angle of  $190^\circ$  is considered the same as an angle of  $10^\circ$ .



## Shape Equivalence

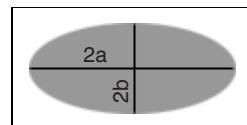
This section describes the following shape-equivalence parameters:

- **Equivalent ellipse minor axis**—Minor axis of the ellipse that has the same area as the particle and a major axis equal to half the max intercept of the particle

This definition gives the following set of equations:

$$\text{particle area} = \pi ab \text{ and}$$

$$\text{max intercept} = 2a$$



The equivalent ellipse minor axis is defined as

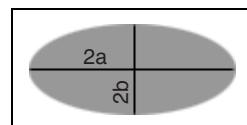
$$2b = \frac{4 \times \text{particle area}}{\pi \times \text{max intercept}}$$

- **Ellipse major axis**—Total length of the major axis of the ellipse that has the same area and same perimeter as a particle. This length is equal to  $2a$ .

This definition gives the following set of equations:

$$\text{Area} = \pi ab$$

$$\text{Perimeter} = \pi\sqrt{2(a^2 + b^2)}$$



This set of equations can be expressed so that the sum  $a + b$  and the product  $ab$  become functions of the parameters **Particle area** and **Particle perimeter**.  $a$  and  $b$  then become the two solutions of the polynomial equation  $X^2 - (a + b)X + ab = 0$ .

Notice that for a given area and perimeter, only one solution ( $a, b$ ) exists.

- **Ellipse minor axis**—Total length of the minor axis of the ellipse that has the same area and same perimeter as a particle. This length is equal to  $2b$

- **Ellipse Ratio**—Ratio of the major axis of the equivalent ellipse to its minor axis

It is defined as  $\frac{\text{ellipse major axis}}{\text{ellipse minor axis}} = \frac{a}{b}$

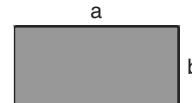
The more elongated the equivalent ellipse, the higher the ellipse ratio. The closer the equivalent ellipse is to a circle, the closer to 1 the ellipse ratio.

- **Rectangle big side**—Length of the big side ( $a$ ) of the rectangle that has the same area and same perimeter as a particle

This definition gives the following set of equations:

$$\text{Area} = ab$$

$$\text{Perimeter} = 2(a + b)$$



This set of equations can be expressed so that the sum  $a + b$  and the product  $ab$  become functions of the parameters **Particle Area** and **Particle Perimeter**.  $a$  and  $b$  then become the two solutions of the following polynomial equation:

$$X^2 - (a + b)X + ab = 0$$

Notice that for a given area and perimeter, only one solution ( $a, b$ ) exists.

- **Rectangle small side**—Length of the small side of the rectangle that has the same area and same perimeter as a particle. This length is equal to  $b$ .

- **Rectangle ratio**—Ratio of the big side of the equivalent rectangle to its small side

It is defined as  $\frac{\text{rectangle big side}}{\text{rectangle small side}} = \frac{a}{b}$

The more elongated the equivalent rectangle, the higher the rectangle ratio.

The closer the equivalent rectangle is to a square, the closer to 1 the rectangle ratio.

## Shape Features

This section describes the following shape-feature parameters:

- **Moments of inertia  $I_{xx}$ ,  $I_{yy}$ ,  $I_{xy}$** —Gives a representation of the distribution of the pixels in a particle with respect to its center of gravity
- **Elongation factor**—Ratio of the longest segment within a particle to the mean length of the perpendicular segments. It is defined as

$$\frac{\text{max intercept}}{\text{mean perpendicular intercept}}$$

The more elongated the shape of a particle, the higher its elongation factor.

- **Compactness factor**— Ratio of a particle area to the area of the smallest rectangle containing the particle. It is defined as

$$\frac{\text{particle area}}{\text{breadth} \times \text{width}}$$

The compactness factor belongs to the interval [0, 1]. The closer the shape of a particle is to a rectangle, the closer to 1 the compactness factor.

- **Heywood circularity factor**—Ratio of a particle perimeter to the perimeter of the circle with the same area. It is defined as

$$\frac{\text{particle perimeter}}{\text{perimeter of circle with same area as particle}} = \frac{\text{particle perimeter}}{2\sqrt{\pi} \times \text{particle area}}$$

The closer the shape of a particle is to a disk, the closer the Heywood circularity factor to 1.

- **Hydraulic radius**—Ratio of a particle area to its perimeter. It is defined as

$$\frac{\text{particle area}}{\text{particle perimeter}}$$

If a particle is a disk with a radius  $R$ , its hydraulic radius is equal to

$$\frac{\pi R^2}{2\pi R} = \frac{R}{2}$$

The hydraulic radius is equal to half the radius  $R$  of the circle such that

$$\frac{\text{circle area}}{\text{circle perimeter}} = \frac{\text{particle area}}{\text{particle perimeter}}$$

- **Waddel disk diameter**—Diameter of the disk with the same area as the particle. It is defined as

$$\frac{2\sqrt{\text{particle area}}}{\sqrt{\pi}}$$

The following tables list the definition of the primary measurements and the measurements that are derived from them.

## Diverse Measurements

These primary coefficients are used in the computation of measurements such as moments of inertia and center of gravity. IMAQ Vision contains the following diverse-measurement parameters:

- **SumX**—Sum of the  $x$  coordinates of each pixel in a particle
- **SumY**—Sum of the  $y$  coordinates of each pixel in a particle
- **SumXX, SumYY, SumXY**—Sum of  $x$  coordinates squared, sum of  $y$  coordinates squared, and sum of  $xy$  coordinates for each pixel in a particle
- **Corrected Projection X**—Sum of the horizontal segments that do not superimpose any other horizontal segment
- **Corrected Projection Y**—Sum of the vertical segments that do not superimpose any other horizontal segment

## In-Depth Discussion

### Definitions of Primary Measurements

$A$	Area
$p$	Perimeter
Left	Leftmost point
Top	Topmost point
Right	Rightmost point
Bottom	Bottommost point

$P_x$  Projection onto the x-axis

$P_y$  Projection onto the y-axis

## Derived Measurements

Table 10-1 describes derived measurements in IMAQ Vision.

**Table 10-1.** Derived Measurements

Symbol	Derived Measurement	Primary Measurement
$l$	<b>Width</b>	Right – Left
$h$	<b>Height</b>	Bottom – Top
$d$	<b>Diagonal</b>	$\sqrt{l^2 + h^2}$
$M_x$	<b>Center of Mass X</b>	$(\Sigma x)/A$
$M_y$	<b>Center of Mass Y</b>	$(\Sigma y)/A$
$I_{xx}$	<b>Inertia XX</b>	$(\Sigma x^2) - A \times M_x^2$
$I_{yy}$	<b>Inertia YY</b>	$(\Sigma y^2) - A \times M_y^2$
$I_{xy}$	<b>Inertia XY</b>	$(\Sigma xy) - A \times M_x \times M_y$
$C_x$	<b>Mean Chord X</b>	$A/P_y$
$C_y$	<b>Mean Chord X</b>	$A/P_x$
$C$	<b>Mean Perpendicular Intercept</b>	<i>Area of convex hull / Max intercept</i>
$A_{2b}$	<b>Equivalent Ellipse Minor Axis</b>	$4 \times A / (\pi S_{\max})$
$d^\circ$	<b>Orientation</b>	<p>If <math>I_{xx} = I_{yy}</math> then <math>d^\circ = 45</math>,  <i>else</i> <math>d^\circ =</math></p> $\frac{90}{\text{atan}(2 \times I_{xy}/(I_{xx} - I_{yy}))}$ <p>If <math>I_{xx} \geq I_{yy}</math> and <math>I_{xy} \geq 0</math>, then <math>d^\circ = 180 - d^\circ</math>  If <math>I_{xx} \geq I_{yy}</math> and <math>I_{xy} &lt; 0</math>, then <math>d^\circ = -d^\circ</math>  If <math>I_{xx} &lt; I_{yy}</math> then <math>d^\circ = 90 - d^\circ</math>  If <math>d^\circ &lt; 0</math>, then <math>d^\circ = 0^\circ</math></p>

**Table 10-1.** Derived Measurements (Continued)

Symbol	Derived Measurement	Primary Measurement
$E_{2a}$	<b>Ellipse Major Axis (2a)</b>	$E_{2a} = \sqrt{\frac{p^2}{2\pi^2} + \frac{2\pi}{A}} + \sqrt{\frac{p^2}{2\pi^2} - \frac{2\pi}{A}}$
$E_{2b}$	<b>Ellipse Minor Axis (2b)</b>	$E_{2b} = \sqrt{\frac{p^2}{2\pi^2} + \frac{2\pi}{A}} - \sqrt{\frac{p^2}{2\pi^2} - \frac{2\pi}{A}}$
$E_{ab}$	<b>Ellipse Ratio</b>	$E_{2a} / E_{2b}$
$R_c$	<b>Rectangle Big Side</b>	$\frac{1}{4}(p + t')$ where $t' = \sqrt{p^2 - 16A}$
$r_c$	<b>Rectangle Small Side</b>	$\frac{1}{4}(p - t')$ where $t' = \sqrt{p^2 - 16A}$
$R_{Rr}$	<b>Rectangle Ratio</b>	$R_c / r_c$
$F_e$	<b>Elongation Factor</b>	<i>Max intercept / C</i>
$F_c$	<b>Compactness Factor</b>	$A / (h \times l)$
$F_H$	<b>Heywood Circularity Factor</b>	$\frac{p}{2\sqrt{\pi A}}$
$F_t$	<b>Type Factor</b>	$\frac{A^2}{4\pi\sqrt{I_{xx} \times I_{yy}}}$
$R_h$	<b>Hydraulic Radius</b>	$A/p$
$R_d$	<b>Waddel Disk Diameter</b>	$2\sqrt{\frac{A}{\pi}}$

# Part IV

---

## Machine Vision

This section describes conceptual information about high-level operations commonly used in machine vision applications such as edge detection, pattern matching, dimensional measurements, and color inspection.

Part IV, *Machine Vision*, contains the following chapters:

Chapter 11, *Edge Detection*, describes edge detection techniques and tools that locate edges, such as the rake, concentric rake, spoke, and caliper.

Chapter 12, *Pattern Matching*, contains information about pattern matching and shape matching.

Chapter 13, *Dimensional Measurements*, contains information about analytic tools, clamps, line fitting, and coordinate systems.

Chapter 14, *Color Inspection*, contains information about the color spectrum, color matching, color location, and color pattern matching.

Chapter 15, *Instrument Readers*, contains information about meters, LCDs, and barcodes.

# Edge Detection

This chapter describes edge detection techniques and tools that locate edges, such as the rake, concentric rake, spoke, and caliper.

## Introduction

---

*Edge detection* finds edges along a line of pixels in the image.

Use the edge detection tools to identify and locate discontinuities in the pixel intensities of an image. The discontinuities are typically associated with abrupt changes in pixel intensity values that characterize the boundaries of objects in a scene. To use the edge detection tools in IMAQ Vision, first specify a search region in the image.

You can specify the search path interactively or programmatically. When specified interactively, you can use one of the line ROI tools to select the search path you want to analyze. You can also programmatically fix the search regions based either on constant values or the result of a previous processing step. For example, you may want to locate edges along a specific portion of a part that has been previously located using blob analysis or pattern matching algorithms. The edge detection software analyzes the pixels along this region to detect edges. You can configure the edge detection tool to find all edges, find the first edge, or find the first and last edges in the region.

## When to Use

Edge detection is an effective tool for many machine vision applications. Edge detection provides your application with information about the location of the boundaries of objects and the presence of discontinuities.

Use edge detection in the following three applications areas: gauging, detection, and alignment.

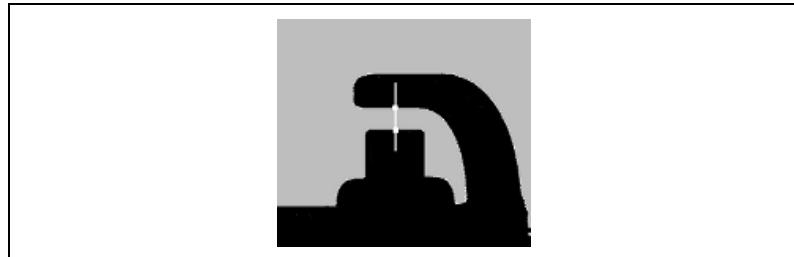
## Gauging

Use *gauging* to make critical dimensional measurements such as lengths, distances, diameters, angles, and counts to determine if the product under inspection is manufactured correctly. The component or part is either classified or rejected, depending on whether the gauged parameters fall inside or outside of the user-defined tolerance limits.

Gauging is often used both inline and offline in production. During inline processes, each component is inspected as it is manufactured. Visual inline gauging inspection is a widely used inspection technique in applications such as mechanical assembly verification, electronic packaging inspection, container inspection, glass vial inspection, and electronic connector inspection.

Gauging applications also often measure the quality of products offline. A sample of products is extracted from the production line. Next, measured distances between features on the object are studied to determine if the sample falls within a tolerance range. You can measure the distances separating the different edges located in an image, as well as positions measured using blob analysis or pattern matching techniques. Edges can also be combined in order to derive best fit lines, projections, intersections, and angles. You can also use edge locations to compute estimations of shape measurements such as circles, ellipses, polygons, and so on.

Figure 11-1 shows how a gauging application uses edge detection to measure the length of the gap in a spark plug.

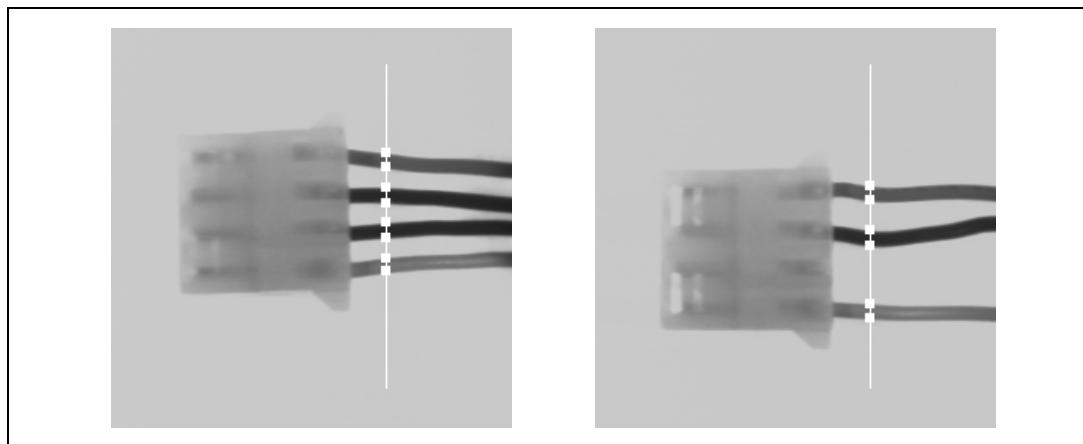


**Figure 11-1.** Gauging Application Using Edge Detection

## Detection

Part present/not present applications are typical in electronic connector assembly and mechanical assembly applications. The objective of the application is to determine if a part is present or not present using line profiles and edge detection. An edge along the line profile is defined by the level of contrast between background and foreground and the slope of the transition. Using this technique, you can count the number of edges along the line profile and compare the result to an expected number of edges. This method offers a less numerically intensive alternative to other image processing methods such as image correlation and pattern matching.

Figure 11-2 shows a simple detection application in which the number of edges detected along the search line profile determines if a connector has been assembled properly. Detection of eight edges indicates that there are four wires. Any other edge count means that the part has not been assembled correctly.



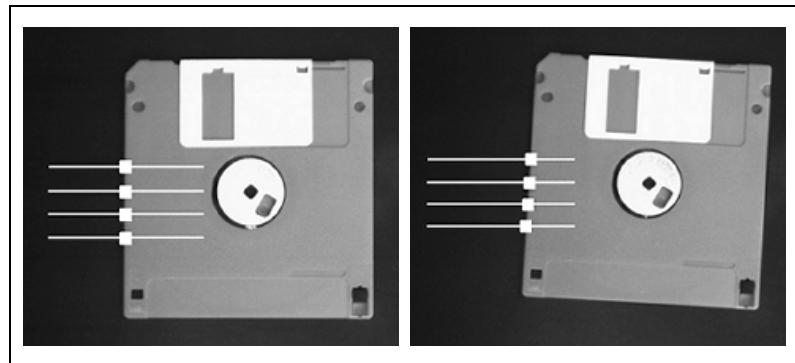
**Figure 11-2.** Connector Inspection Using Edge Detection

You can also use edge detection to detect structural defects such as cracks or cosmetic defects such as scratches on a part. If the part is of uniform intensity, then these defects show up as sharp changes in the intensity profile. Edge detection identifies these changes.

## Alignment

**Alignment** determines the position and orientation of a part. In many machine vision applications, the object that you want to inspect may be at different locations in the image. Edge detection finds the location of the object in the image before you perform the inspection, so that you can inspect only the regions of interest. The position and orientation of the part can also be used to provide feedback information to a positioning device, such as a stage.

Figure 11-3 shows an example of detecting the left boundary of a disk in the image. You can use the location of the edges to determine the orientation of the disk.



**Figure 11-3.** Alignment Using Edge Detection

## Edge Detection Concepts

### Definition of an Edge

An **edge** is a significant change in the grayscale values between adjacent pixels in an image. In IMAQ Vision, edge detection works on a one-dimensional profile of pixel values along a search region, as shown in Figure 11-4. The one-dimensional search region can take the form of a line, the perimeter of a circle or ellipse, the boundary of a rectangle or polygon, or a freehand region. The software analyzes the pixel values along the profile to detect significant intensity changes. You can specify characteristics of the intensity changes to determine which changes constitute an edge.

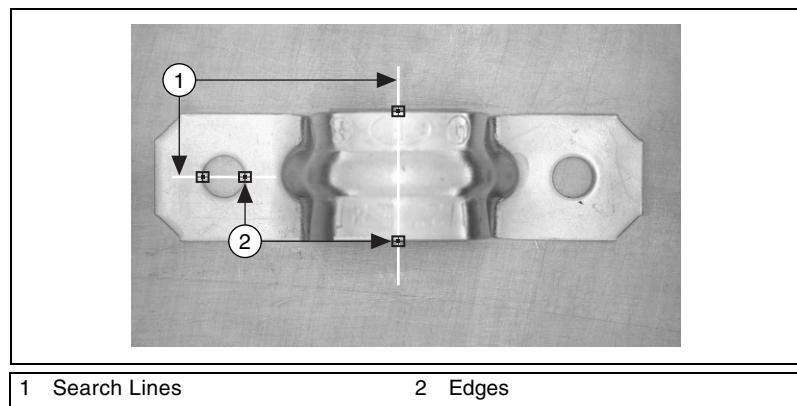


Figure 11-4. Examples of Edges Located on a Bracket

## Characteristics of an Edge

Figure 11-5 shows a common model that is used to characterize an edge.

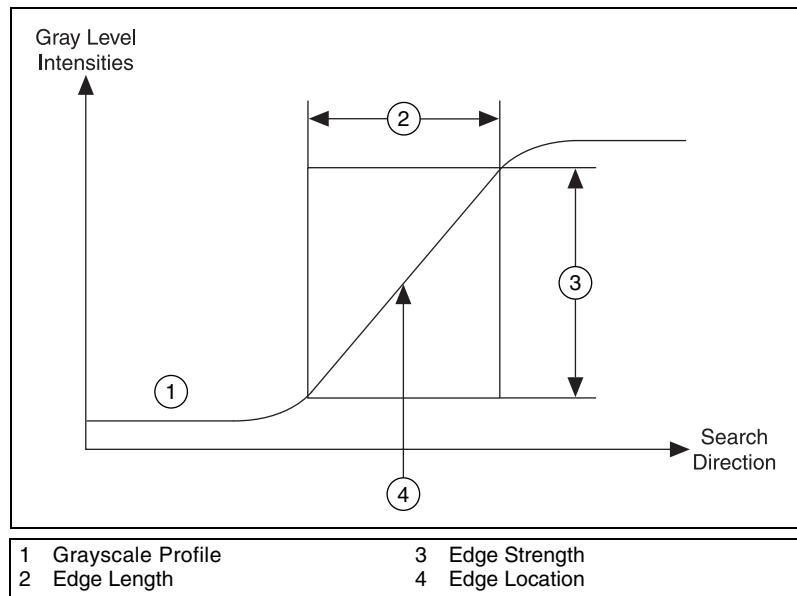
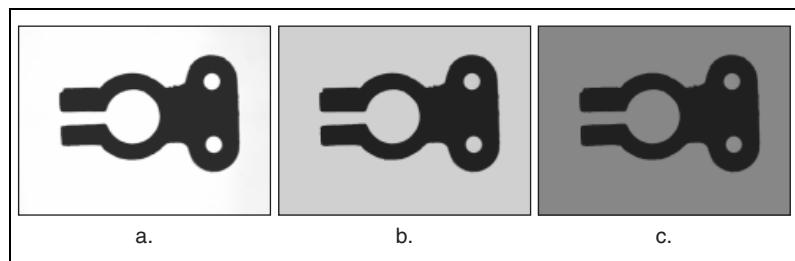


Figure 11-5. Edge Model

The main parameters of this model are:

- **Edge strength**—Defines the minimum difference in the grayscale values between the background and the edge. The edge strength is also called the edge contrast. Figure 11-6 shows an image that has different edge strengths. The strength of an edge can vary due to many reasons including:
  - Lighting conditions—If the overall light in the scene is low, the edges in image will have low strengths. Figure 11-6 illustrates a change in the edge strength along the boundary of an object relative to different lighting conditions.
  - Objects with different grayscale characteristics—The presence of a very bright object causes other objects in the image with a lower overall intensity to have edges with smaller strengths.



**Figure 11-6.** Examples of Edges with Different Strengths

- **Edge length**—Defines the maximum distance in which the desired grayscale difference between the edge and the background must occur. The length characterizes the slope of the edge. Use a longer length to detect edges with a gradual transition between the background and the edge.
- **Edge polarity**—Describes if an edge is rising or falling. A rising edge is characterized by an increase in grayscale values as you cross the edge. A falling edge is characterized by a decrease in grayscale values as you cross the edge. The polarity of an edge is linked to the search direction. Figure 11-7 shows examples of edge polarities.
- **Edge position**—The  $x, y$  location of an edge in the image. Figure 11-5 shows the edge position for the edge model.

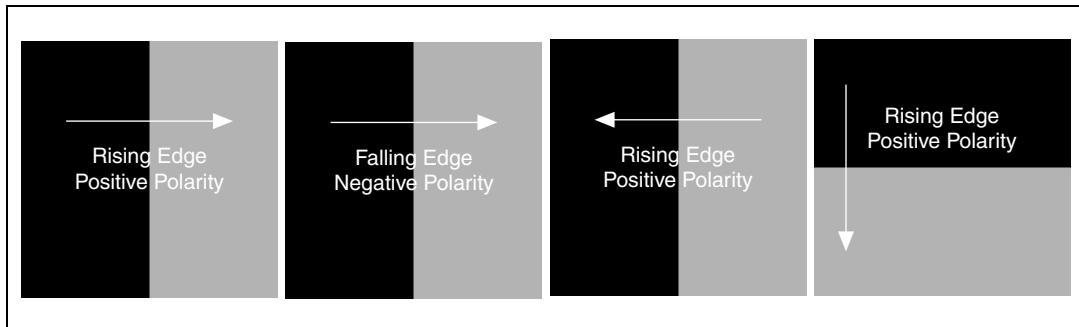


Figure 11-7. Edge Polarity

## Edge Detection Methods

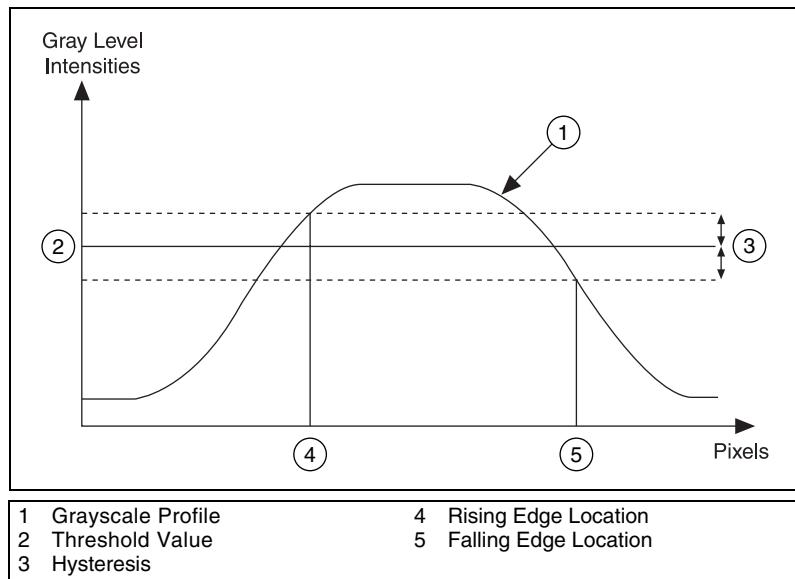
IMAQ Vision provides two ways to perform edge detection. Both methods compute the edge strength at each pixel along the one-dimensional profile. An edge occurs when the edge strength is greater than a minimum strength. Additional checks find the correct location of the edge. You can specify the minimum strength by using the *contrast* parameter in the software.

### Simple Edge Detection

The software uses the pixel value at any point along the pixel profile to define the edge strength at that point. To locate an edge point, the software scans the pixel profile pixel by pixel from the beginning to the end. A rising edge is detected at the first point at which the pixel value is greater than a threshold value plus a hysteresis value. Set this threshold value to define the minimum edge strength required for qualifying edges. Use the hysteresis value to declare different edge strengths for the rising and falling edges. Once a rising edge is detected, the software looks for a falling edge. A falling edge is detected when the pixel value falls below the specified threshold value. This process is repeated until the end of the pixel profile.

The first edge along the profile can be either a rising or falling edge. Figure 11-8 shows the simple edge model.

The simple edge detection method works very well when there is little noise in the image and when there is a distinct demarcation between the object and the background.



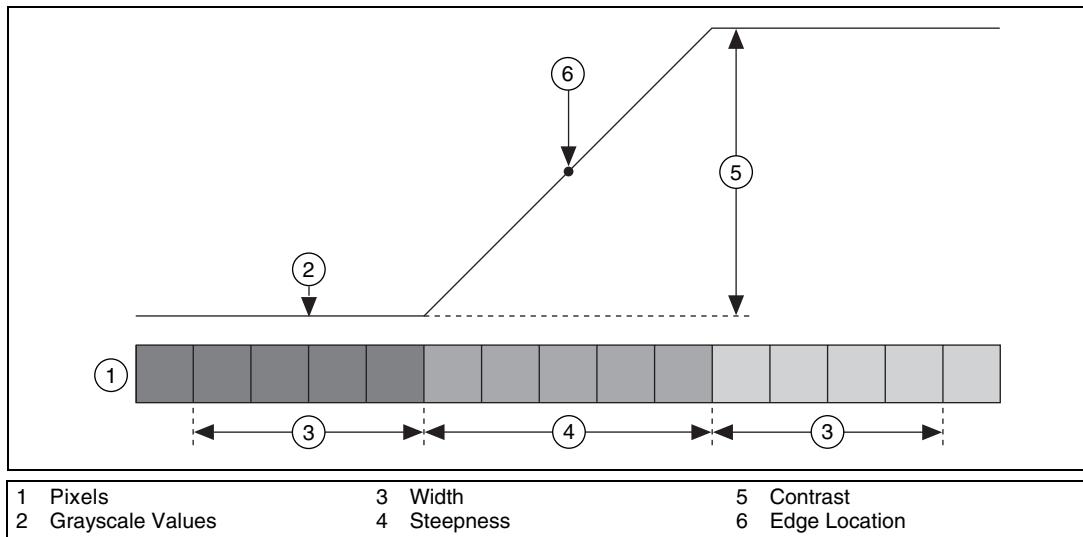
**Figure 11-8.** Simple Edge Detection

## Advanced Edge Detection

To compute the edge strength at a given point along the pixel profile, the software averages pixels before and after the analyzed point. The pixels that are averaged after the point can be a certain pixel distance from the point, which you define by setting the steepness parameter. This number corresponds to the expected transition region in the edge profile. Define the number of pixels averaged on each side by setting the width parameter. After computing the average, the software computes the difference between these averages to determine the contrast. Filtering reduces the effects of noise along the profile. If you expect the image to contain a lot of noise, use a large filter width. Figure 11-9 shows the relationship between the parameters and the edge profile.

To find the edge, the software scans across the one-dimensional grayscale profile pixel by pixel. At each point, the edge strength (contrast) is computed. If the contrast at the current point is greater than the user-set

value for the minimum contrast for an edge, the point is stored for further analysis. Starting from this point, successive points are analyzed until the contrast reaches a maximum value and then falls below that value. The point where the contrast reaches the maximum value is tagged as the start edge location. The value of the steepness parameter is added to the start edge location to obtain the end edge location. The first point between the start edge location and end edge location where the difference between the point's intensity value and the start edge value is greater than or equal to half the difference between the start edge value and end edge value is returned as the edge location.



**Figure 11-9.** Advanced Edge Detection

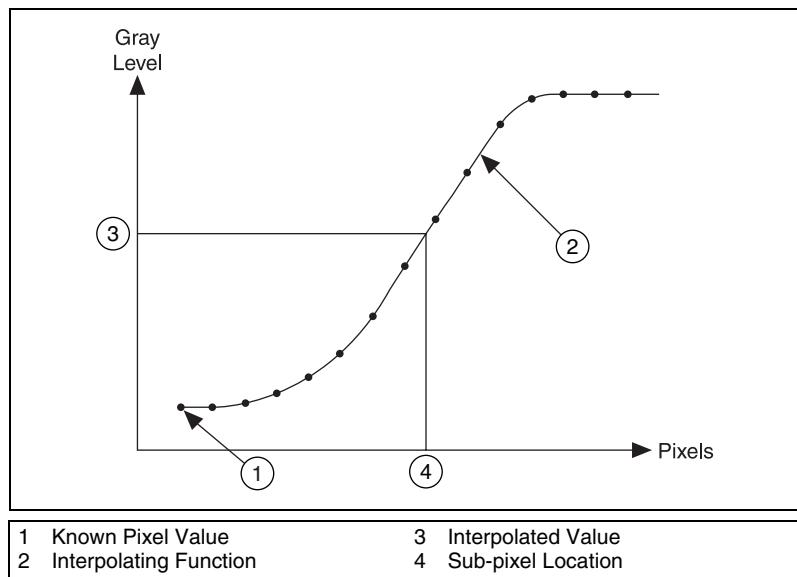
## Sub-pixel Accuracy

When the resolution of the image is high enough, most measurement applications make accurate measurements using pixel accuracy only. However, it is sometimes difficult to obtain the minimum image resolution needed by a machine vision application because of the limits on the size of the sensors available (or affordable). In these cases, you need to find edge positions with sub-pixel accuracy.

Sub-pixel analysis is a software method that estimates the pixel values that a higher resolution imaging system would have provided. To compute the location of an edge with sub-pixel precision, the edge detection software first fits a higher-order interpolating function, such as a quadratic or cubic function, to the pixel intensity data.

The interpolating function provides the edge detection algorithm with pixel intensity values between the original pixel values. The software then uses the intensity information to find the location of the edge with sub-pixel accuracy.

Figure 11-10 shows how a cubic spline function fits to a set of pixel values. Using this fit, values at locations in between pixels are estimated. The edge detection algorithms use these values to estimate the location of an edge with sub-pixel accuracy.



**Figure 11-10.** Obtaining Sub-pixel Information Using Interpolation

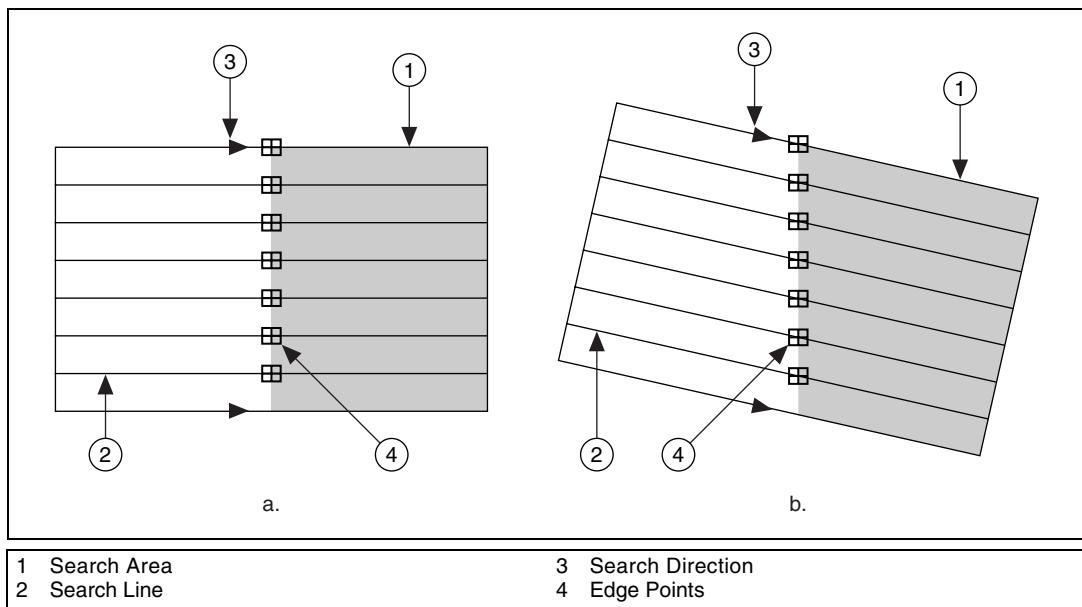
With the imaging system components and software tools available today, you can reliably estimate one-fourth sub-pixel accuracy. However, the results of the estimation depend heavily on the imaging setup, such as lighting conditions and the camera lens. Before resorting to sub-pixel information, try to improve the image resolution. For more information on improving your images, see the *IMAQ Vision User Manual*.

## Extending Edge Detection to Two-Dimensional Search Regions

The edge detection tool in IMAQ Vision works on a one-dimensional profile. You can extend the use of the edge detection tool to some other two-dimensional search areas with one of the following tools: Rake, Spoke and Concentric Rake. In these edge detection variations, the two-dimensional search area is covered by a number of search lines over which the edge detection is performed. You can control the number of the search lines used in the search region by defining the separation between the lines.

### Rake

Rake works on a rectangular search region. The search lines are drawn parallel to the orientation of the rectangle. Control the number of lines in the area by specifying the search direction as left to right or right to left for a horizontally oriented rectangle. Specify the search direction as top to bottom or bottom to top for a vertically oriented rectangle. Figure 11-11 illustrates the basics of the Rake function.



**Figure 11-11.** Rake Function

## Spoke

Spoke works on an annular search region, working along search lines that are drawn from the center of the region to the outer boundary and that fall within the search area. Control the number of lines in the region by specifying the angle between each line. Specify the search direction as either going from the center outward or from the outer boundary to the center. Figure 11-12 illustrates the basics of the Spoke function.

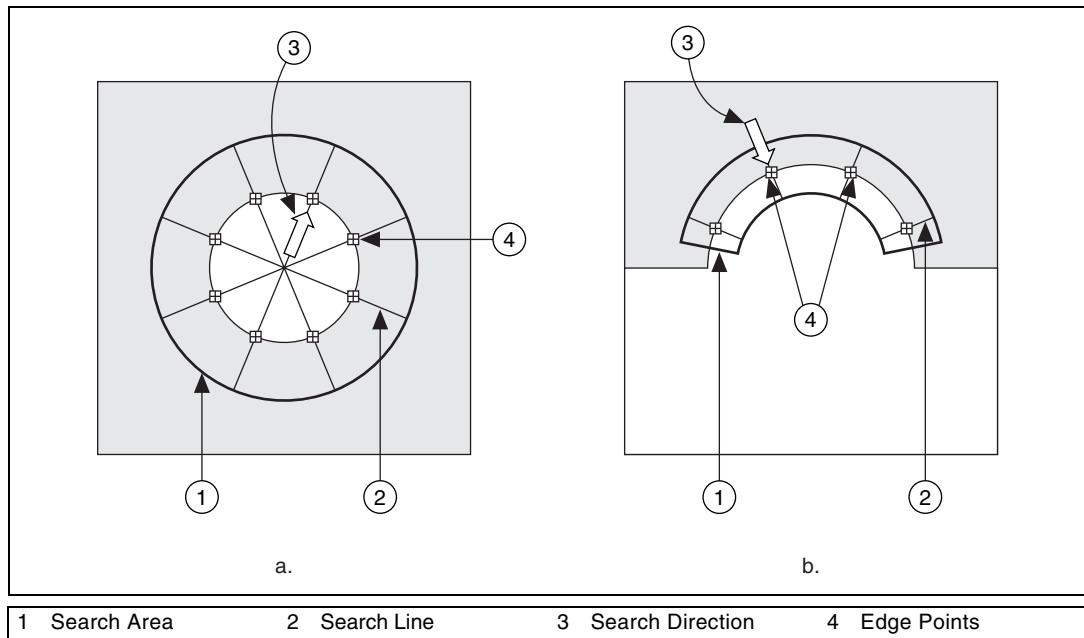
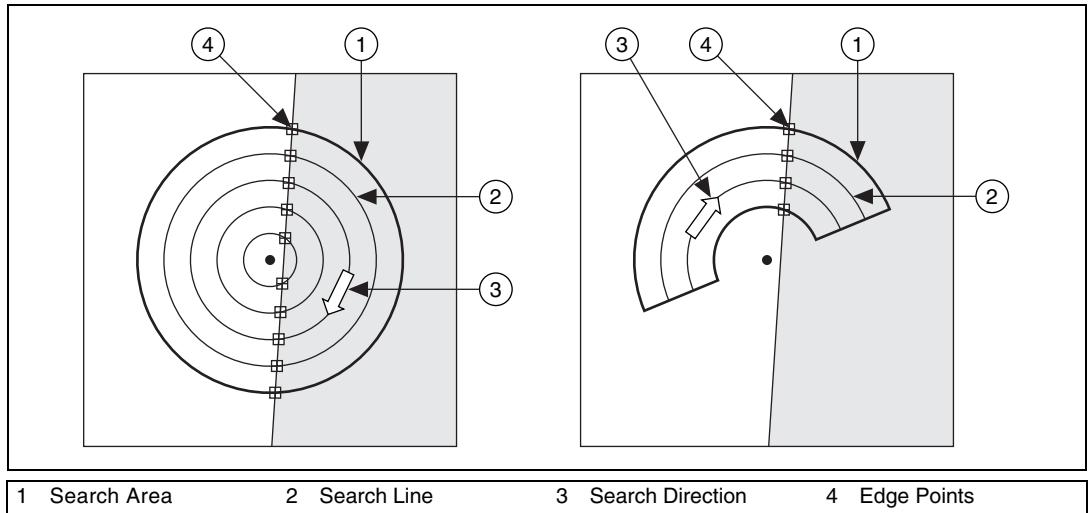


Figure 11-12. Spoke Function

## Concentric Rake

Concentric Rake works on an annular search region. The concentric rake is an adaptation of the Rake to an annular region. Figure 11-13 illustrates the basics of the concentric rake. Edge detection is performed along search lines that occur in the search region and that are concentric to the outer circular boundary. Control the number of concentric search lines that are used for the edge detection by specifying the radial distance between the concentric lines in pixels. Specify the direction of the search as either clockwise or anti-clockwise.



**Figure 11-13.** Concentric Rake Function

# Pattern Matching

This chapter contains information about pattern matching and shape matching.

## Introduction

*Pattern matching* locates regions of a grayscale image that match a predetermined template. Pattern matching finds template matches regardless of poor lighting, blur, noise, shifting of the template, or rotation of the template.

Use pattern matching to quickly locate known reference patterns, or fiducials, in an image. With pattern matching you create a model or template that represents the object for which you are searching. Then your machine vision application searches for the model in each acquired image, calculating a score for each match. The score relates how closely the model matches the pattern found.

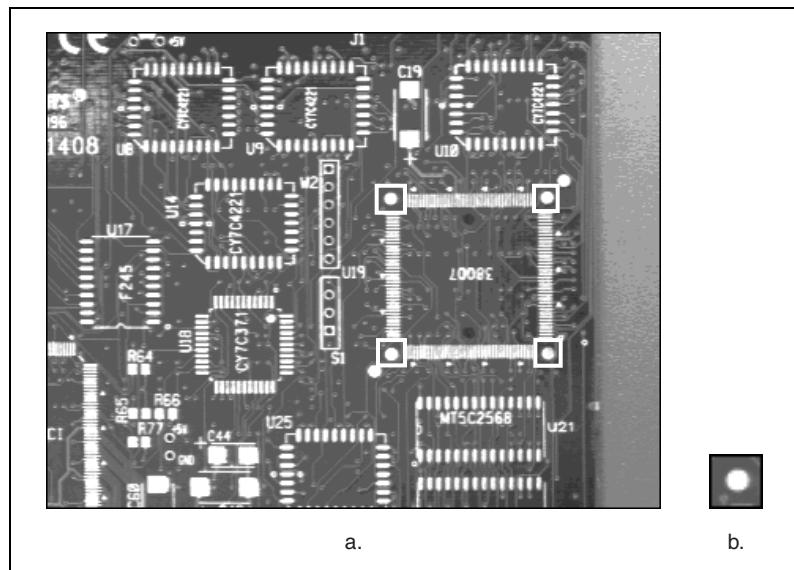
## When to Use

Pattern matching algorithms are some of the most important functions in image processing because of their use in varying applications. You can use pattern matching in the following three general applications:

- Alignment—Determines the position and orientation of a known object by locating fiducials. Use the fiducials as points of reference on the object.
- Gauging—Measures lengths, diameters, angles, and other critical dimensions. If the measurements fall outside set tolerance levels, the component is rejected. Use pattern matching to locate the object you want to gauge.
- Inspection—Detects simple flaws, such as missing parts or unreadable printing.

The pattern matching tools in IMAQ Vision measure the similarity between an idealized representation of a feature, called a model or template, and a feature that may be present in an image. A *feature* is defined as a specific pattern of pixels in an image.

Pattern matching is the key to many applications. Pattern matching provides your application with information about the number of instances and location of the template within an image. For example, you may search an image containing a printed circuit board for one or more alignment marks (fiducials). The machine vision application uses the marks to align the board for chip placement from a chip mounting device. Figure 12-1a shows part of a circuit board. Figure 12-1b shows a common fiducial used in Printed Circuit Board (PCB) inspections or chip pick-and-place applications.



**Figure 12-1.** Example of a Common Fiducial

Gauging applications locate and then measure, or gauge, the distance between these objects. If the measurement falls within a tolerance range, the part is considered good. If it falls outside the tolerance range, the component is rejected. Searching and finding a feature is the key processing task that determines the success of many gauging applications, such as inspecting the leads on a quad pack or inspecting an antilock-brake sensor. In real-time applications, search speed is critically important.

## Pattern Matching Concepts

### What to Expect from a Pattern Matching Tool

Because pattern matching is the first step in many machine vision applications, it should work reliably under various conditions.

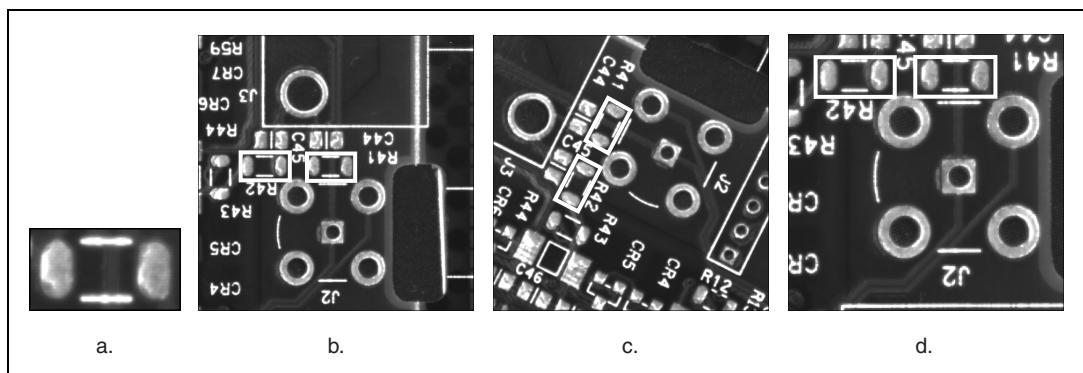
In automated machine vision applications, the visual appearance of materials or components under inspection can change due to factors such as orientation of the part, scale changes, and lighting changes. The pattern matching tool maintains its ability to locate the reference patterns despite these changes. The following are commonly occurring situations under which the pattern matching tool gives accurate results.

### Pattern Orientation and Multiple Instances

A pattern matching tool can locate the reference pattern in an image even if the pattern in the image is rotated or scaled. When a pattern is rotated or scaled in the image, the pattern matching tool can detect the following:

- The pattern in the image
- The position of the pattern in the image
- The orientation of the pattern
- Multiple instances of the pattern in the image (if applicable)

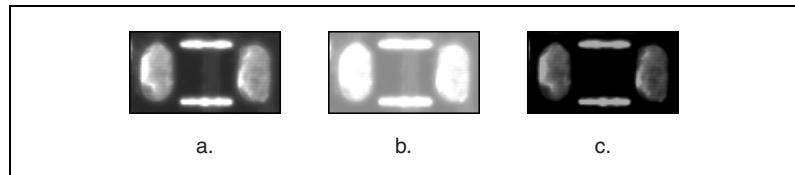
Figure 12-2a shows a template image, or pattern. Figure 12-2b shows the template shifted in the image. Figure 12-2c shows the template rotated in the image. Figure 12-2d shows the template scaled in the image. Figures 12-2b to 12-2d illustrate multiple instances of the template.



**Figure 12-2.** Pattern Orientation and Multiple Instances

## Ambient Lighting Conditions

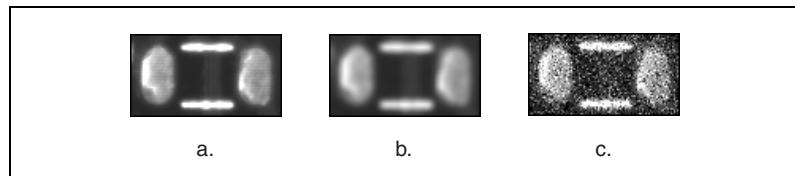
The pattern matching tool can find the reference pattern in an image under conditions of uniform changes in the lighting across the image. Figure 12-3 illustrates the typical conditions under which pattern matching works correctly. Figure 12-3a shows the original template image. Figure 12-3b shows the same pattern under bright light. Figure 12-3c shows the pattern under poor lighting.



**Figure 12-3.** Examples of Lighting Conditions

## Blur and Noise Conditions

Pattern matching can find patterns that have undergone some transformation because of blurring or noise. Blurring usually occurs because of incorrect focus or depth of field changes. Figure 12-4 illustrates typical blurring and noise conditions under which pattern matching works correctly. Figure 12-4a shows the original template image. Figure 12-4b shows changes on the image caused by blurring. Figure 12-4c shows changes on the image caused by noise.



**Figure 12-4.** Examples of Blur and Noise

## Pattern Matching Techniques

Pattern matching includes traditional techniques, newer techniques, and other techniques such as blob analysis.

## Traditional Pattern Matching

Traditional pattern matching techniques include normalized cross correlation, pyramidal matching, and scale-invariant matching.

### Cross Correlation

Normalized cross correlation is the most common way to find a template in an image. Because the underlying mechanism for correlation is based on a series of multiplication operations, the correlation process is time consuming. New technologies such as MMX allow you to do parallel multiplications and also reduce overall computation time. You can speed up the matching process by reducing the size of the image and restricting the region in the image where the matching is done. However, the basic normalized cross correlation operation does not meet the speed requirements of many applications.

### Pyramidal Matching

You can improve the computation time by reducing the size of the image and the template. Once such technique is called pyramidal matching. In this method, both the image and the template are sub-sampled to smaller spatial resolutions. The image and the template can be reduced to one-fourth their original size. Matching is performed first on the reduced images. Because the images are smaller, matching is faster. Once the matching is complete, only areas with a high match need to be considered as matching areas in the original image.

### Scale-Invariant Matching

Normalized cross correlation is a good technique for finding patterns in an image when the patterns in the image are not scaled or rotated.

Typically, cross correlation can detect patterns of the same size up to a rotation of 5° to 10°. Extending correlation to detect patterns that are invariant to scale changes and rotation is difficult.

For scale-invariant matching, you must repeat the process of scaling or resizing the template and then perform the correlation operation. This adds a significant amount of computation to your matching process. Normalizing for rotation is even more difficult. If a clue regarding rotation can be extracted from the image, you can simply rotate the template and do the correlation. However, if the nature of rotation is unknown, looking for the best match requires exhaustive rotations of the template.

You can also carry out correlation in the frequency domain using the FFT. If the image and the template are the same size, this approach is more efficient than performing correlation in the spatial domain. In the frequency domain, correlation is obtained by multiplying the FFT of the image by the complex conjugate of the FFT of the template. Normalized cross correlation is considerably more difficult to implement in the frequency domain.

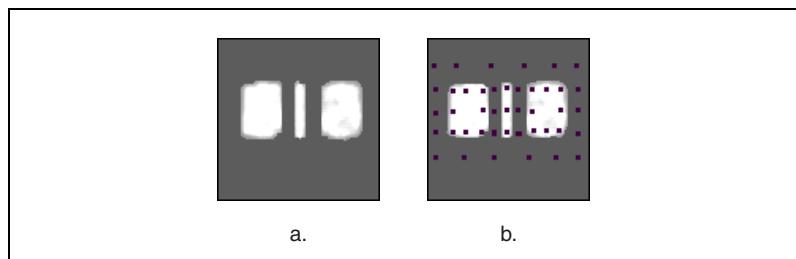
## New Pattern Matching Techniques

Classical pattern matching methods have their limitations. New methods (such as the ones used in IMAQ Vision) try to incorporate *image understanding* techniques to interpret the template information, and then use this information to find the template in the image. Image understanding refers to image processing techniques that generate information about the features of a template image. These methods include the following:

- Geometric modeling of images
- Efficient non-uniform sampling of images
- Extraction of template information that is rotation independent and scale independent.

These techniques reduce the amount of information needed to fully characterize an image or pattern, which speeds up the searching process. Also, extracting useful information from a template and removing the redundant and noisy information provides a more accurate search.

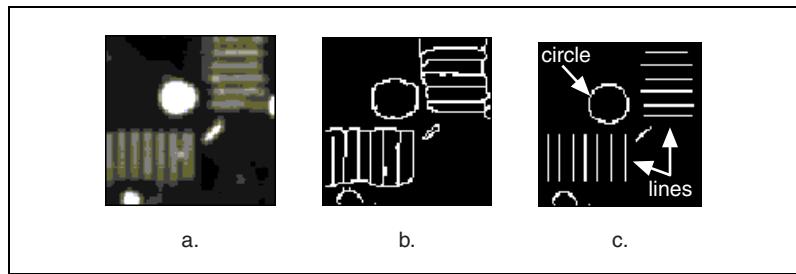
One new pattern matching technique uses non-uniform sampling. Since most images contain redundant information, using all the information in the image to match patterns is time-intensive and inaccurate. You can improve the speed and accuracy of the pattern matching tool by sampling the image and extracting a few points that represent the overall content of the image. Figure 12-5 shows an example of good sampling techniques for pattern matching. Figure 12-5a shows the original template, or reference image. The black dots on Figure 12-5b represent the points on the image that are used to represent the template.



**Figure 12-5.** Good Pattern Matching Sampling Techniques

Many of the new techniques for pattern matching use the image's edge information to provide information about the structure of the image. The amount of information in the image is reduced to contain only significant data about the edges.

You can process the edge image further to extract higher level geometric information about the image, such as the number of straight lines or circles present in the image. Pattern matching is then limited to the matching of edge and/or geometric information between the template and the image. Figure 12-6 illustrates the importance of edges and geometric modeling. Figure 12-6a shows the reference pattern, Figure 12-6b shows the edge information in the pattern, and Figure 12-6c shows the higher-level geometric interpretation of edges in the form of geometric objects, such as circles and lines.



**Figure 12-6.** Edge Detection and Pattern Matching Techniques

IMAQ Vision uses a combination of the edge information in the image and an intelligent image sampling technique to match patterns. In cases where the pattern can be rotated in the image, a similar technique is used, but with specially chosen template pixels whose values (or relative change in values) reflect the rotation of the pattern. The result is fast and accurate pattern matching. IMAQ Vision pattern matching is able to accurately locate objects where they vary in size ( $\pm 5\%$ ) and orientation (between  $0^\circ$  and  $360^\circ$ ) and when their appearance is degraded.

## In-Depth Discussion

This section provides additional information you may need for building successful pattern matching tools.

### Cross Correlation

The following is the basic concept of correlation: Consider a sub-image  $w(x,y)$  of size  $K \times L$  within an image  $f(x,y)$  of size  $M \times N$ , where  $K \leq M$  and  $L \leq N$ . The correlation between  $w(x,y)$  and  $f(x,y)$  at a point  $(i,j)$  is given by

$$C(i,j) = \sum_{x=0}^{L-1} \sum_{y=0}^{K-1} w(x,y)f(x+i, y+j)$$

where  $i = 0, 1, \dots, M-1$ ,  $j = 0, 1, \dots, N-1$ , and the summation is taken over the region in the image where  $w$  and  $f$  overlap.

Figure 12-7 illustrates the correlation procedure. Assume that the origin of the image  $f$  is at the top left corner. Correlation is the process of moving the template or subimage  $w$  around the image area and computing the value  $C$  in that area. This involves multiplying each pixel in the template by the image pixel that it overlaps and then summing the results over all the pixels of the template. The maximum value of  $C$  indicates the position where  $w$  best matches  $f$ . Correlation values are not accurate at the borders of the image.

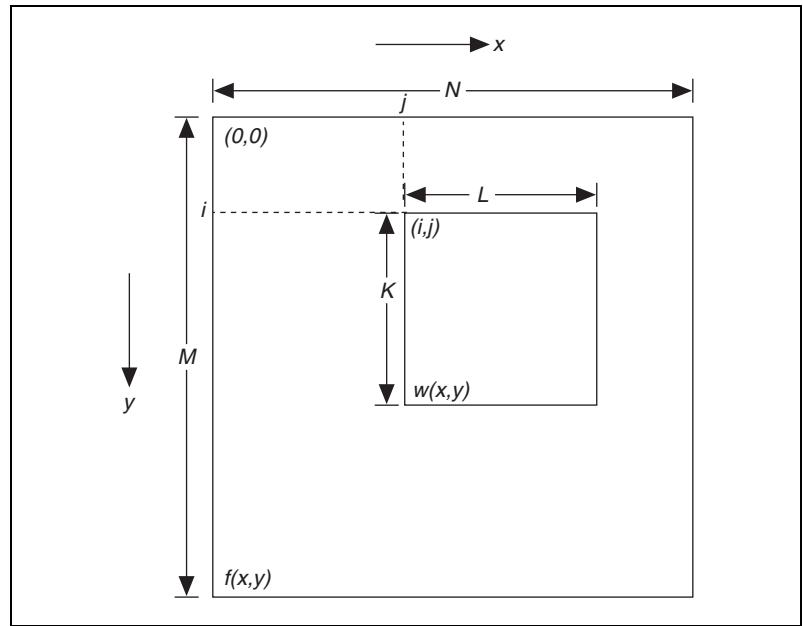


Figure 12-7. The Correlation Procedure

Basic correlation is very sensitive to amplitude changes in the image, such as intensity, and in the template. For example, if the intensity of the image  $f$  is doubled, so will the values of  $c$ . You can overcome sensitivity by computing the normalized correlation coefficient, which is defined as:

$$R(i,j) = \frac{\sum_{x=0}^{L-1} \sum_{y=0}^{K-1} (w(x,y) - \bar{w})(f(x+i,y+j) - \bar{f}(i,j))}{\left[ \sum_{x=0}^{L-1} \sum_{y=0}^{K-1} (w(x,y) - \bar{w})^2 \right]^{\frac{1}{2}} \left[ \sum_{x=0}^{L-1} \sum_{y=0}^{K-1} (f(x+i,y+j) - \bar{f}(i,j))^2 \right]^{\frac{1}{2}}}$$

where  $\bar{w}$  (calculated only once) is the average intensity value of the pixels in the template  $w$ . The variable  $\bar{f}$  is the average value of  $f$  in the region coincident with the current location of  $w$ . The value of  $R$  lies in the range  $-1$  to  $1$  and is independent of scale changes in the intensity values of  $f$  and  $w$ .

# Shape Matching

---

Shape matching searches for the presence of a shape in a binary image and specifies the location of each matching shape. IMAQ Vision detects the shape even if it is rotated or scaled.

Binary shape matching is performed by extracting parameters from a template object that represent the shape of the object and are invariant to the rotation and scale of the shape. These parameters are then compared with a similar set of parameters extracted from other objects. Binary shape matching has the benefit of finding features regardless of size and orientation.

## When to Use

If you are searching for a feature that has a known shape but unknown size and orientation, and the image can be thresholded, consider using binary shape matching.

Binary shape matching is useful in robot guidance applications such as a robot arm sorting parts into groups of similar shapes.

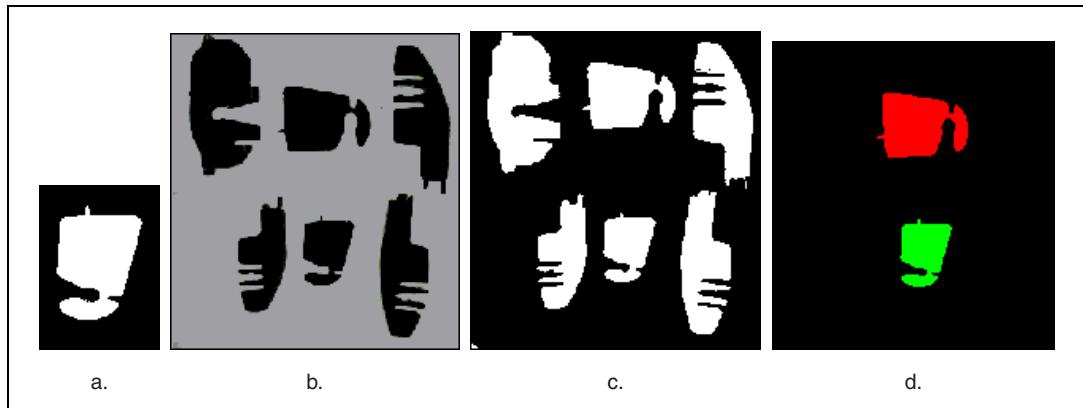
## Shape Matching Concepts

A shape matching function takes the following as inputs:

- An image of the shape (template) that you are looking for
- A binary image containing the parts that you want to sort
- A tolerance level that indicates the permitted degree of mismatch between the template and the parts

The output will be an image containing only the matched parts and a report detailing the location of each part, the centroid of the part, and a score that indicates the degree of match between the template and the part.

Figure 12-8 shows how binary shape matching is used to sort windshield wiper parts into different shapes. Figure 12-8a shows the shape template. Figure 12-8b shows the original grayscale image of different windshield parts. Figure 12-8c shows the binary (or thresholded) version of the original image. Figure 12-8d shows the output of the shape matching function. The shape matching function finds the desired shape in the image regardless of variations in size and orientation.



**Figure 12-8.** Using Shape Matching to Search for Windshield Wiper Parts

# Dimensional Measurements

This chapter contains information about coordinate systems, analytic tools, and clamps.

## Introduction

---

You can use dimensional measurement or *gauging* tools in IMAQ Vision to obtain quantifiable, critical distance measurements—such as distances, angles, areas, line fits, circular fits, and counts—to determine if a certain product was manufactured correctly.

Components such as connectors, switches, and relays are small and manufactured in high quantity. Human inspection of these components is tedious, time consuming, and inconsistent. IMAQ Vision can quickly and consistently measure certain features on these components and generate a report of the results. If the gauged distance or count does not fall within user-specified tolerance limits, the component or part fails to meet production specifications and should be rejected.

## When to Use

Use gauging for applications in which inspection decisions are made on critical dimensional information obtained from image of the part. Gauging is often used in both inline and offline production. During inline processes, each component is inspected as it is manufactured. Inline gauging inspection is often used in mechanical assembly verification, electronic packaging inspection, container inspection, glass vial inspection, and electronic connector inspection.

You can also use gauging to measure the quality of products off-line. A sample of products is extracted from the production line. Then, using measured distances between features on the object, IMAQ Vision determines if the sample falls within a tolerance range. Gauging techniques also allow you to measure the distance between blobs and edges in binary images and easily quantify image measurements.

## Dimensional Measurements Concepts

The gauging process consists of four steps:

1. Locate the component or part in the image.
2. Locate features in different areas of the part.
3. Make measurements using these features.
4. Compare the measurements to specifications to determine whether the part passes inspection.

### Locating the Part in the Image

A typical gauging application extracts measurements from ROIs rather than from an entire image. To use this technique, the necessary parts of the object must always appear inside the ROIs you define.

Usually, the object under inspection appears shifted or rotated within the images you want to process. When this occurs, the ROIs need to shift and rotate in the same way as the object. In order for the ROIs to move in relation to the object, you must locate the object in every image. Locating the object in the image involves determining the x, y position and the orientation of the object in the image using the reference coordinate system functions. You can build a coordinate reference using edge detection or pattern matching.

### Locating Features

To gauge an object, you need to find landmarks or object features on which you can base your measurements. In most applications, you can make measurements based on points detected in the image or geometric fits to the detected points. Object features that are useful for measurements fall into two categories:

- Edge points located along the boundary of an object located by edge detection method
- Shapes or patterns within the object located by pattern matching

## Making Measurements

You can make different types of measurements from the features found in the image. Typical measurements include the distance between points; the angle between two lines represented by three or four points; the best line, circular, or elliptical fits; and the areas of geometric shapes—such as circles, ellipses, and polygons—that fit detected points. For more information about the types of measurements you can make, see the *IMAQ Vision for LabVIEW User Manual* or the *IMAQ Vision for Measurement Studio User Manual*.

## Qualifying Measurements

The last step of a gauging application involves determining the quality of the part based on the measurements obtained from the image. You can determine the quality of the part using either relative comparisons or absolute comparisons.

In many applications, the measurements obtained from the inspection image can be compared to the same measurements obtained from a standard specification or a reference image. Because all the measurements are made on images of the part, you can compare them directly.

In other applications, the dimensional measurements obtained from the image must be compared with values that are specified in real units. In this case, convert the measurements from the image into real-world units using the calibration tools described in Chapter 3, *System Setup and Calibration*.

## Coordinate System

In a typical machine vision application, measurements are extracted from an ROI rather than from the entire image. The object under inspection must always appear in the defined ROI in order to extract measurements from that ROI.

When the location and orientation of the object under inspection is always the same in the inspection images, you can make measurements directly without locating the object in every inspection image.

In most cases, the object under inspection is not positioned in the camera's field of view consistently enough to use fixed search areas. If the object is shifted or rotated within an image, the search areas should shift and rotate with the object. The search areas are defined relative to a coordinate system. A coordinate system is defined by a reference point (origin) and a reference angle in the image or by the lines that make up its two axes.

## When to Use

Use coordinate systems in a gauging application when the object does not appear in the same position in every inspection image. You can also use a coordinate system to define search areas on the object relative to the location of the object in the image.

## Concepts

All measurements are defined with respect to a coordinate system. A coordinate system is based on a characteristic feature of the object under inspection, which is used as a reference for the measurements. When you inspect an object, first locate the reference feature in the inspection image. Choose a feature on the object that the software can reliably detect in every image. Do not choose a feature that may be affected by manufacturing errors that would make the feature impossible to locate in images of defective parts.

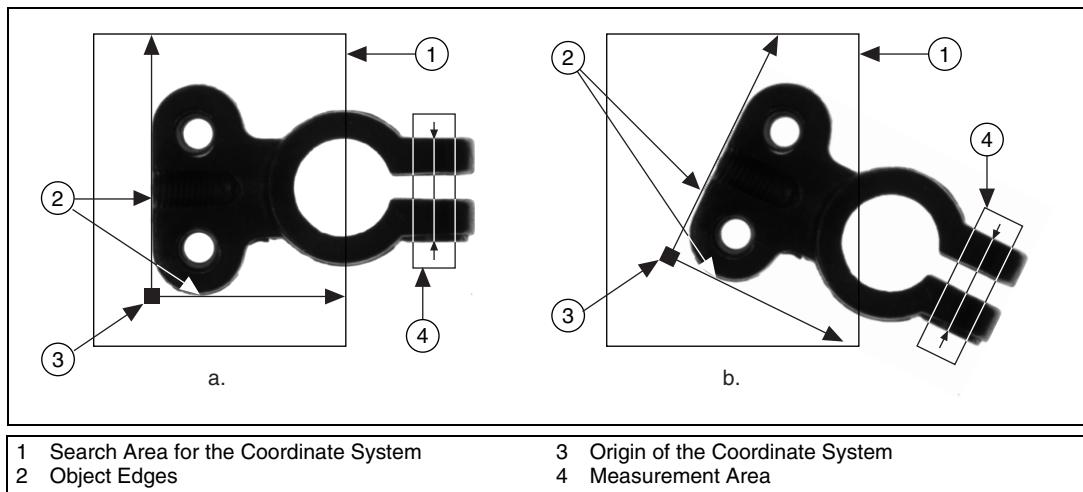
You can restrict the region in the image in which the software searches for the feature by specifying an ROI that encloses the feature. Defining an ROI in which you expect to find the feature can prevent mismatches if the feature appears in multiple regions of the image. A small ROI may also improve the locating speed.

Follow these general steps to define a coordinate system and make measurements based on the new coordinate system:

1. Define a reference coordinate system.
  - a. Define a search area that encompasses the reference feature or features on which you base your coordinate system. Make sure that the search area encompasses the features in all your inspection images.
  - b. Locate an easy-to-find reference feature of the object under inspection. That feature serves as the base for a reference coordinate system in a reference image. You can use two main techniques to locate the feature: edge detection or pattern matching.
  - c. The software builds a coordinate system to keep track of the location and orientation of the object in the image.
2. Set up measurement areas within the reference image in which you want to make measurements.
3. Acquire an image of the object to inspect or measure.

4. Update the coordinate system. During this step, IMAQ Vision locates the features in the search area and builds a new coordinate system based on the new location of the features.
5. Make measurements.
  - a. IMAQ Vision computes the difference between the reference coordinate system and the new coordinate system. Based on this difference, the software moves the new measurement areas with respect to the new coordinate system.
  - b. Make measurements within the updated measurement area.

Figure 13-1a shows a reference image with a defined reference coordinate system. Figure 13-1b shows an inspection image with an updated coordinate system.



**Figure 13-1.** Coordinate Systems of a Reference Image and Inspection Image

## In-Depth Discussion

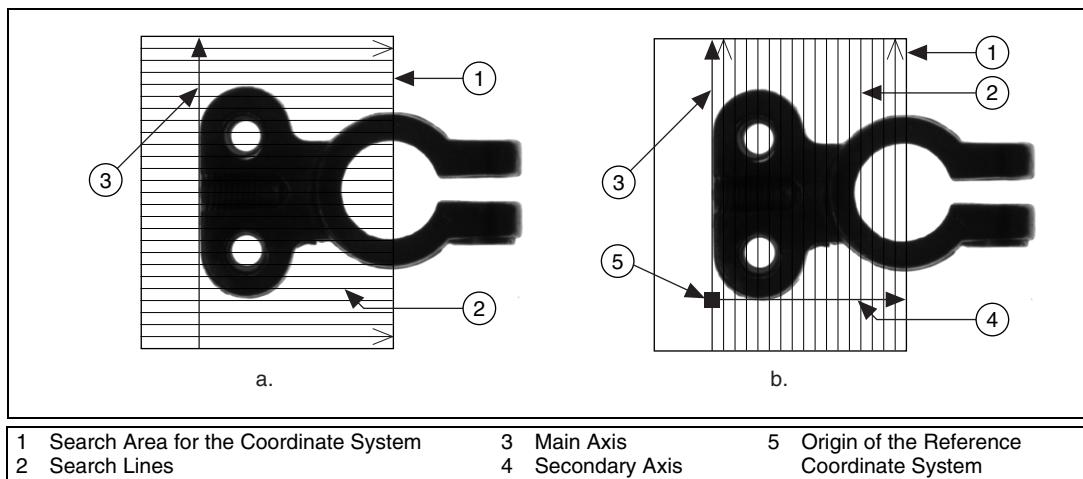
You can use four different strategies to build a coordinate system. Two strategies are based on detecting reference edges of the object under inspection. The other two strategies involve locating a specific pattern using a pattern matching algorithm.

## Edge-Based Coordinate System Functions

These functions determine the axis of the coordinate system by locating edges of the part under inspection. Use an edge-based method if you can identify two straight, distinct, non-parallel edges on the object you want to locate. Because the software uses these edges as references for creating the coordinate system, choose edges that are unambiguous and always present in the object under inspection.

### Single Search Area

This method involves locating the two axes of the coordinate system—the main axis and secondary axis—in a single search area based on an edge detection algorithm. First, the function determines the main axis of the coordinate system, as shown in Figure 13-2a. A rake function finds the intersection pixels between multiple search lines and the edge of the object. You can specify the search direction along these search lines. The intersection points are determined by their contrast, width, and steepness. For more information about detecting edges, see Chapter 11, *Edge Detection*. A line fitted through the intersection points defines the main axis. The function then searches for a secondary axis within the same search area, as shown in Figure 13-2b. The software uses multiple parallel lines that are parallel to the main axis to scan for edges, and then fits a line through the edge of the object closest to the search area and perpendicular to the main axis. This line defines the secondary axis of the coordinate system. The secondary axis must not be parallel to the main axis. The intersection between the main axis and secondary axis defines the origin of the reference coordinate system.

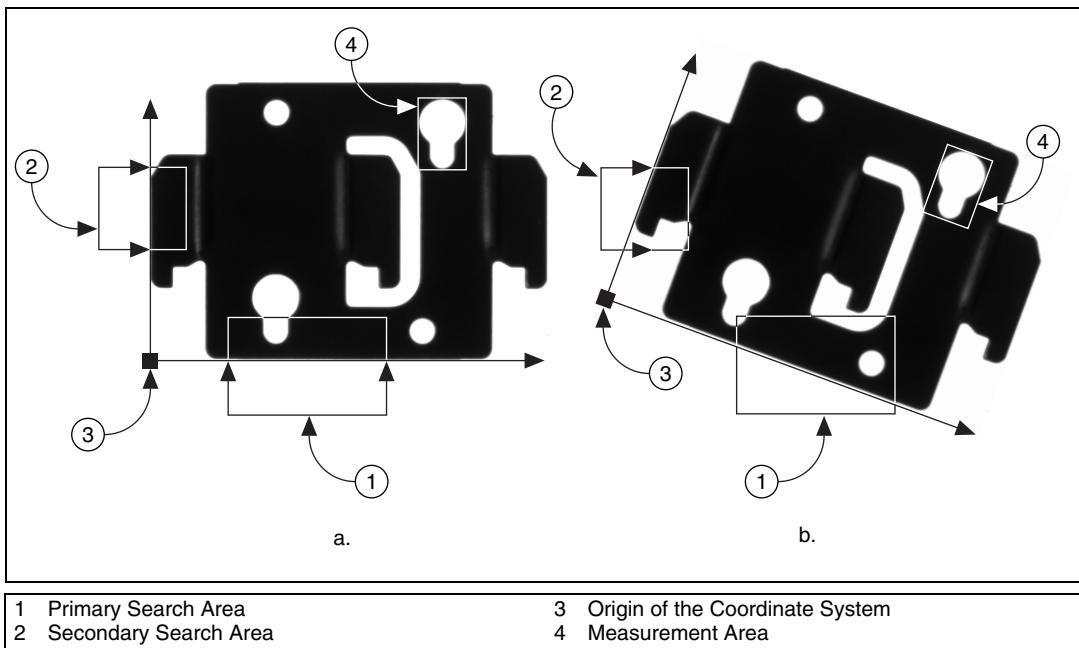
**Figure 13-2.** Locating A Coordinate System with One Search Area

### Two Search Areas

This method uses the same operating mode as the single search area method. However, the two edges used to define the coordinate system axes are located in two distinct search areas.

The function first determines the position of the main axis of the coordinate system. It locates the intersection points between a set of parallel search lines in the primary search area and a distinct straight edge of the object. The intersection points are determined based on their contrast, width, and steepness. For more information about detecting edges, see Chapter 11, *Edge Detection*. A line fitted through the intersection points defines the primary axis. The process is repeated perpendicularly in the secondary search area to locate the secondary axis. The intersection between the primary axis and secondary axis is the origin of the coordinate system.

Figure 13-3a shows a reference image with a defined reference coordinate system. Figure 13-3b shows an inspection image with an updated coordinate system.



**Figure 13-3.** Locating A Coordinate System with Two Search Areas

## Pattern Matching-Based Coordinate System Functions

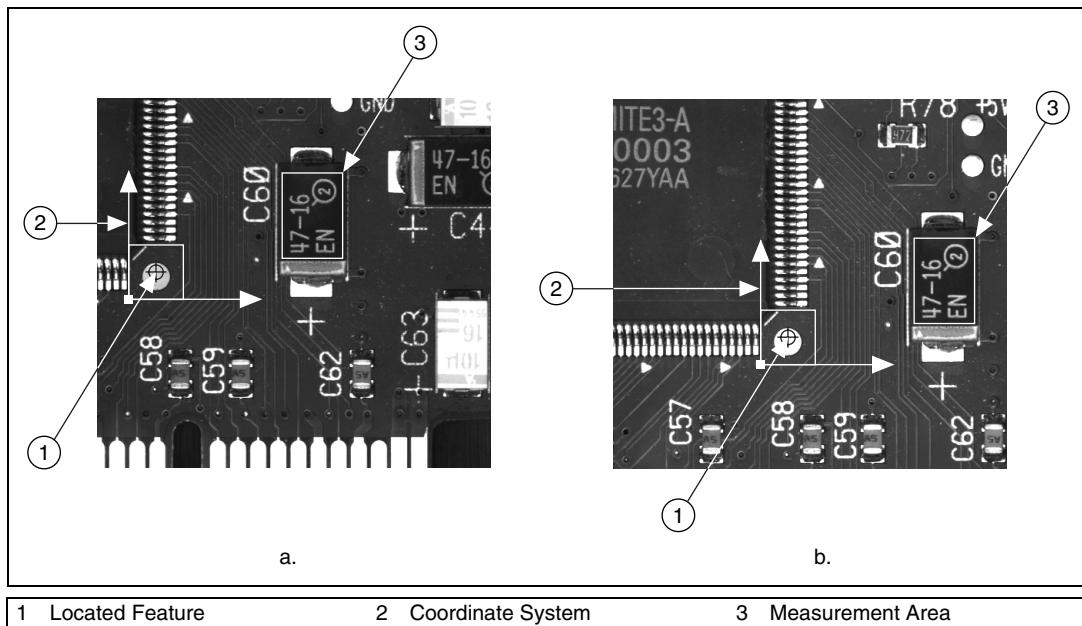
Using pattern matching techniques to locate a reference feature is a good alternative to edge detection when you cannot find straight, distinct edges in the image. The reference feature, or template, is the basis for the coordinate system.

The software searches for a template image in a rectangular search area of the reference image. The location and orientation of the located template is used to create the reference position of a coordinate system or to update the current location and orientation of an existing coordinate system.

The same constraints on feature stability and robustness that apply to the edge-detection techniques also apply to pattern matching. Pattern matching uses one of two strategies: shift-invariant pattern matching and rotation-invariant pattern matching. Shift-invariant pattern matching locates a template in an ROI or the entire image with a maximum tolerance in rotation of  $\pm 5^\circ$ . The rotation-invariant strategy locates a template in the

image even when the template varies in orientation (between 0° and 360°). For recommendations about the type of patterns to use for a template, see Chapter 12, *Pattern Matching*.

Figure 13-4 illustrates how to locate a coordinate system using a shift-invariant pattern matching strategy. Figure 13-4a shows a reference image with a defined reference coordinate system. Figure 13-4b shows an inspection image with an updated coordinate system.



**Figure 13-4.** Locating a Coordinate System with Shift-Invariant Pattern Matching

Figure 13-5 illustrates how to locate a coordinate system using a rotation-invariant pattern matching strategy. Figure 13-5a shows a reference image with a defined reference coordinate system. Figure 13-5b shows an inspection image with an updated coordinate system.

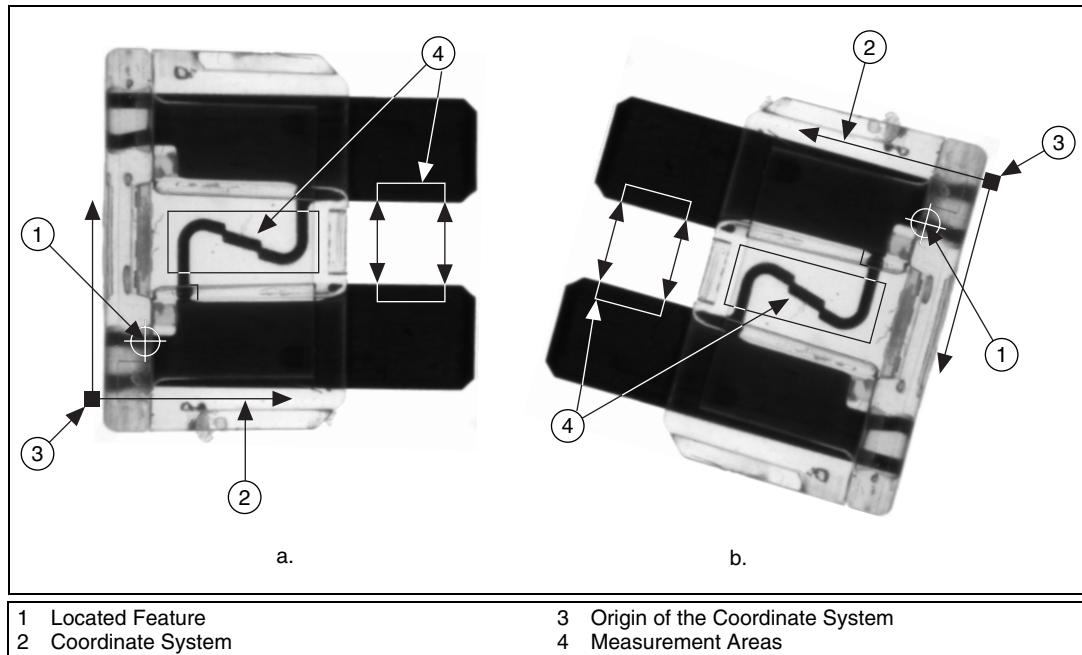


Figure 13-5. Locating a Coordinate System with Rotation-Invariant Pattern Matching

## Finding Features or Measurement Points

Before making measurements, you must locate features that you can use to make the measurements. There are many ways to find these features on an image. The most common features used to make measurements are points along the boundary of the part you want to gauge.

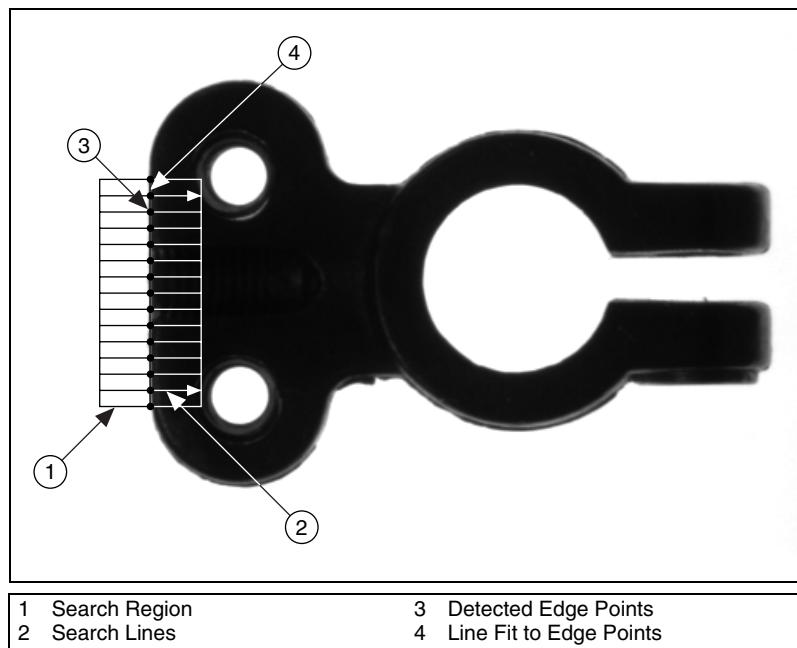
### Edge-Based Features

Use the edge detection techniques described in Chapter 11, [Edge Detection](#), to find edge points along a single search contour or along multiple search contours defined inside a 2D search area.

## Line and Circular Features

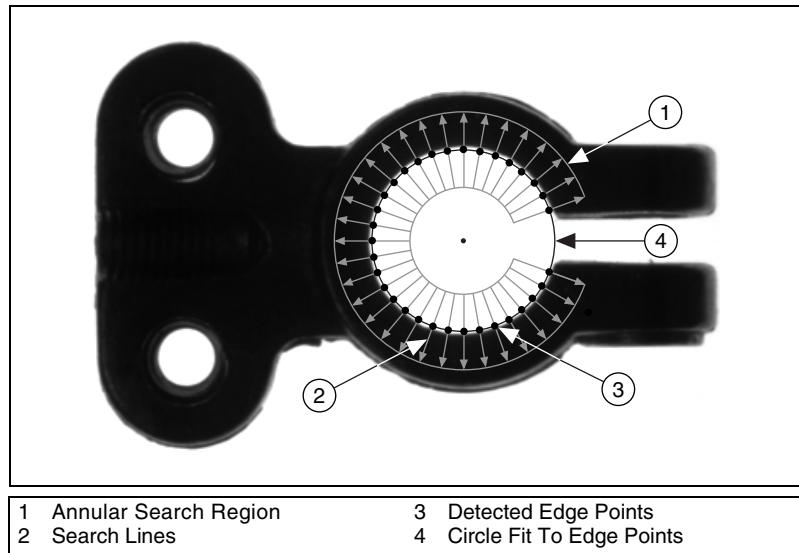
Use the line detection functions in IMAQ Vision to find vertically or horizontally oriented lines. These functions use the rake and concentric rake functions to find a set of points along the edge of an object and then fit a line through the edge.

Refer to Chapter 11, [Edge Detection](#), for more information on the rake and concentric rake functions. The line fitting method is described later in this chapter. Figure 13-6 illustrates how a rake finds a straight edge.



**Figure 13-6.** Finding a Straight Feature

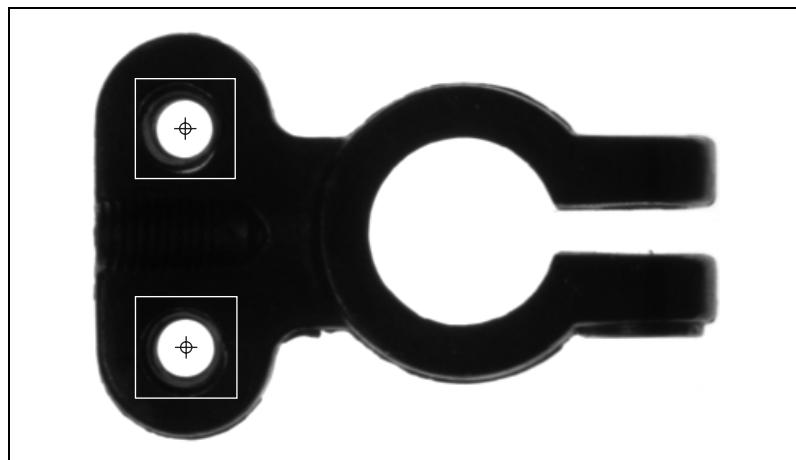
Use the circle detection function to locate circular edges. This function uses a spoke to find points on a circular edge, and then fits a circle on the detected points. Figure 13-7 illustrates how a spoke finds circular edges.



**Figure 13-7.** Circle Detection

## Shape-Based Features

Use pattern matching or color pattern matching to find features that are better described by the shape and grayscale or color content than the boundaries of the part.



**Figure 13-8.** Finding Shape Features

## Making Measurements on the Image

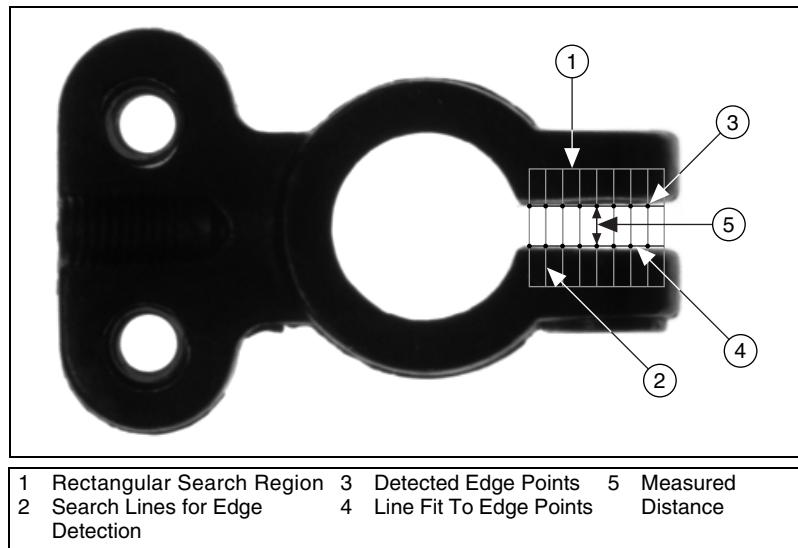
After you have located points in the image, you can make distance or geometrical measurements based on those points.

## Distance Measurements

Make distance measurements using one of the following methods:

- Measure the distance between points found by one of feature detection methods.
- Measure the distance between two edges of an object using the clamp functions available in IMAQ Vision. Clamp functions measure the separation between two edges in a rectangular region using the rake function. First, the clamp functions detect points along the two edges using the rake function. They then compute the distance between the detected points and return the largest or smallest distance. Use the clamp functions to do the following:
  - Find the smallest or largest horizontal separation between two vertically oriented edges.
  - Find the smallest or largest vertical separation between two horizontally oriented edges.

Figure 13-9 illustrates how a clamp function finds the minimum distance between edges of an object.



**Figure 13-9.** Clamp Function

## Analytic Geometry

You can make the following geometrical measurements from the feature points detected in the image:

- The area of a polygon specified by its vertex points
- The line that fits to a set of points and the equation of that line
- The circle that fits to a set of points and its area, perimeter, and radius
- The ellipse that fits to a set of points and its area, perimeter, and the lengths of its major and minor axis
- The intersection point of two lines specified by their start and end points
- The line bisecting the angle formed by two lines
- The line midway between a point and a line that is parallel to the line
- The perpendicular line from a point to line, which computes the perpendicular distance between the point and the line

## Line Fitting

The line fitting function in IMAQ Vision uses a robust algorithm to find a line that best fits a set of points. The line fitting function works specifically with the feature points obtained during gauging applications.

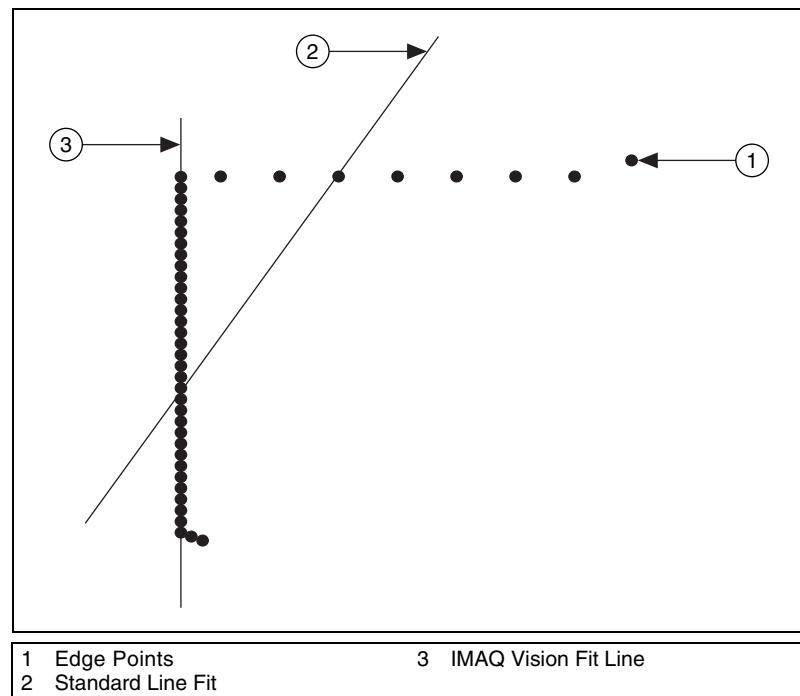
In a typical gauging application, a rake or a concentric rake function finds a set of points that lie along a straight edge of the object. In an ideal case, all the detected points would make a straight line. However, the points usually do not appear in a straight line for one of the following reasons:

- The edge of the object does not occupy the entire search region used by the rake.
- The edge of the object is not a continuous straight line.
- Noise in the image causes points along the edge to shift from their true positions.

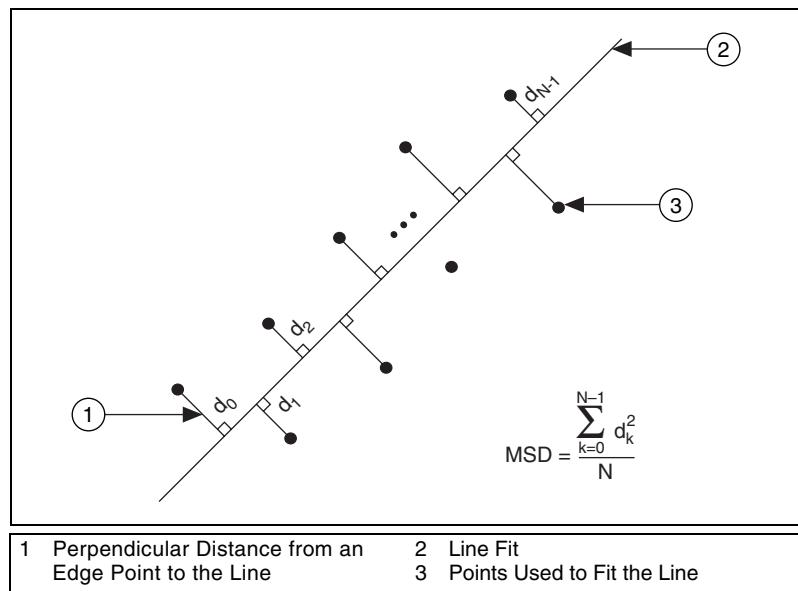
Figure 13-10 shows an example of a set of points located by the rake function. As shown in the figure, a typical line fitting algorithm that uses all of the points to fit a line returns inaccurate results. The line fitting function in IMAQ Vision compensates for outlying points in the dataset and returns a more accurate result.

IMAQ Vision uses the following process to fit a line. IMAQ Vision assumes that a point is part of a line if the point lies within a user-defined distance (pixel radius) from the fitted line. Then the line fitting algorithm fits a line to a subset of points that fall along an almost straight line. IMAQ Vision determines the quality of the line fit by measuring its mean square distance (MSD), which is the average of the squared distances between each point and the estimated line. Figure 13-11 shows how the MSD is calculated. Next, the line fitting function removes the subset of points from the original set. IMAQ Vision repeats these steps until all points have been fit. Then, the line fitting algorithm finds the line with the lowest MSD, which corresponds to the line with the best quality. The function then improves the quality of the line by successively removing the furthest points from the line until a user-defined minimum score is obtained or a user-specified maximum number of iterations is exceeded.

The result of the line fitting function is a line that is fit to the strongest subset of the points after ignoring the outlying points, as shown in Figure 13-12.

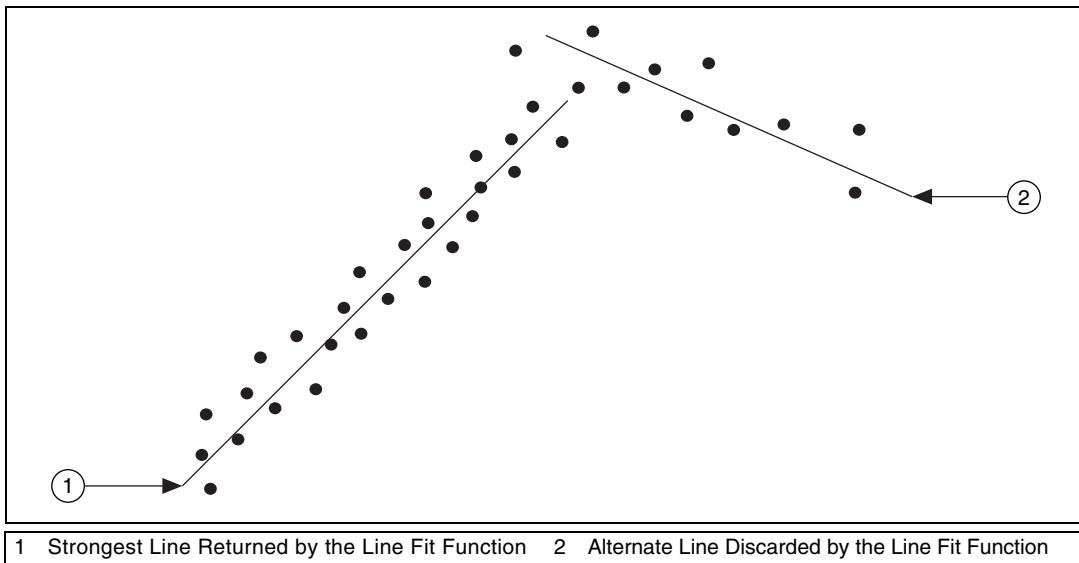


**Figure 13-10.** Data Set and Fitted Line Using Two Methods

**Figure 13-11.** Calculation of the Mean Square Distance (MSD)

The pixel radius, minimum score, and maximum iteration parameters control the behavior of the line fit function.

The pixel radius defines the maximum distance allowed, in pixels, between a valid point and the estimated line. The algorithm estimates a line where at least half the points in the set are within the pixel radius. If a set of points does not have such a line, the function attempts to return the line that has the most number of valid points.

**Figure 13-12.** Strongest Line Fit

Increasing the pixel radius increases the distance allowed between a point and the estimated line. Typically, you can use the imaging system resolution and the amount of noise in your system to gauge this parameter. If the resolution of the imaging system is very high, use a small pixel radius to minimize the use of outlying points in the line fit. Use a higher pixel radius if your image is noisy.

The minimum score allows you to improve the quality of the estimated line. The line fitting function removes the point furthest from the fit line, and then refits a line to the remaining points and computes the MSD of the line. Next, the function computes a line fit score (LFS) for the new fit using the following equation.

$$LFS = \left( \frac{1 - MSD}{PR^2} \right) \times 1000$$

where  $PR$  is the pixel radius.

IMAQ Vision repeats the entire process until the score is greater than or equal to the minimum score or until the number of iterations exceeds the user-defined maximum number of iterations.

Use a high minimum score to obtain the most accurate line fit. For example, combining a large pixel radius and a high minimum score produces an accurate fit within a very noisy data set. A small pixel radius and a small minimum score produces a robust fit in a standard data set.

The maximum number of iterations defines a limit in the search for a line that satisfies the minimum score. If you reach the maximum number of iterations before the algorithm finds a line matching the desired minimum score, the algorithm stops and returns the current line. If you do not need to improve the quality of the line in order to obtain the desired results, set the maximum iterations value to 0 in the line fit function.

# Color Inspection

This chapter contains information about the color spectrum, color matching, color location, and color pattern matching.

## The Color Spectrum

The color spectrum represents the three dimensional color information associated with an image or a region of an image in a concise one dimensional form that can be used by many of IMAQ Vision's color processing software. Use the color spectrum for color matching, color location and color pattern matching applications with IMAQ Vision.

The color spectrum is a one-dimensional representation of the three-dimensional color information in an image. The spectrum represents all the color information associated with that image or a region of the image in the HSL space. The information is packaged in a form that can be used by the color processing functions in IMAQ Vision.

## Color Space Used to Generate the Spectrum

The color spectrum represents the color distribution of an image in the HSL space, as shown in Figure 14-1. If the input image is in RGB format, the image is first converted to HSL format and the color spectrum is computed from the HSL space. Using HSL images directly, such as those acquired with an IMAQ PCI/PXI-1411 image acquisition device with an onboard RGB to HSL conversion for color matching, improves the operation speed.

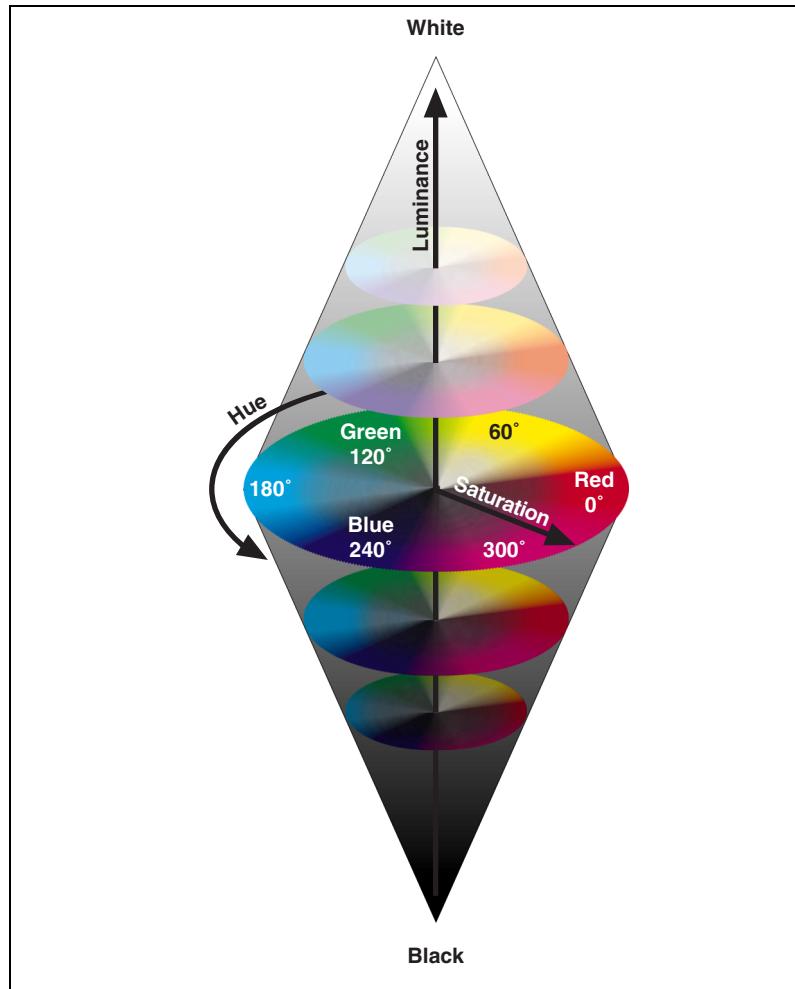
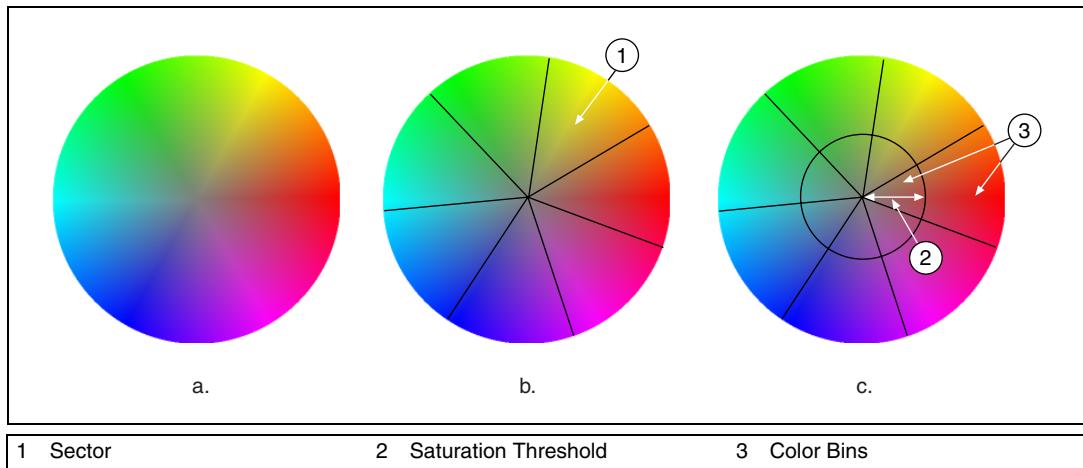


Figure 14-1. The HSL Color Space

Colors represented in the HSL model space are easy for humans to quantify. The luminance (intensity) component in the HSL space is separated from the color information. This feature leads to a more robust color representation independent of light intensity variation. However, the chromaticity (hue and saturation) plane cannot be used to represent the black and white colors that often are the background colors in many machine vision applications. See Chapter 1, *Digital Images*, for more information on color spaces.

## Generating the Color Spectrum

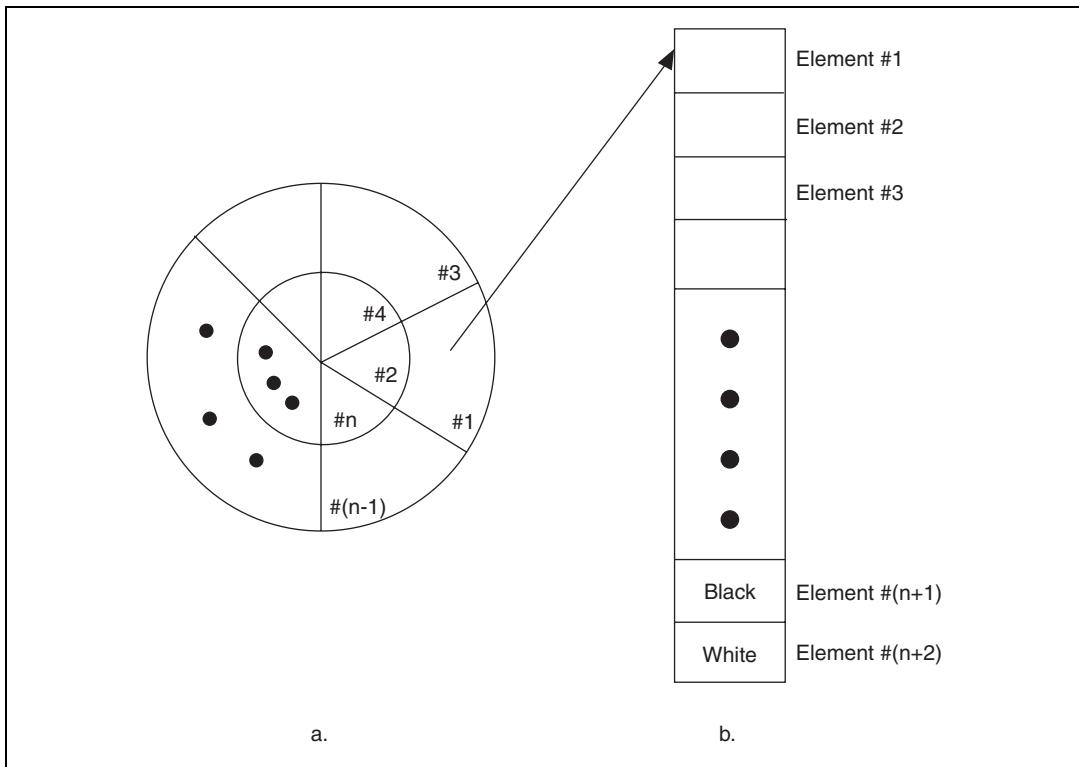
Each element in the color spectrum array corresponds to a bin of colors in the HSL space. The last two elements of the array represent black and white colors, respectively. Figure 14-2 illustrates how the HSL color space is divided into bins. The hue space is divided into a number of equal sectors, and each sector is further divided into two parts representing a part with high saturation values and another part with low saturation values. Each of these parts corresponds to a color bin (an element in the color spectrum array).



**Figure 14-2.** The HSL Space Divided into Bins and Sectors

The color sensitivity parameter determines the number of sectors the hue space is divided into. Figure 14-2a shows the hue color space when luminance is equal to 128. Figure 14-2b shows the hue space divided into a number of sectors, depending on the desired color sensitivity. Figure 14-2c shows each sector divided further into a high saturation bin and a low saturation bin. The saturation threshold determines the radius of the inner circle that separates each sector into bins.

Figure 14-3 shows the correspondence between the color spectrum elements and the bins in the color space. The first element in the color spectrum array represents the high saturation part in the first sector, the second element represents the low saturation part, the third element the high saturation part of the second sector and so on. If there are  $n$  bins in the color space, the color spectrum array contains  $n + 2$  elements. The last two components in the color spectrum represent the black and white color, respectively.



**Figure 14-3.** Hue Color Space and the Color Spectrum Array Relationship

A color spectrum with a larger number of bins (elements) represents the color information in an image with more detail, such as a higher color resolution, than a spectrum with fewer bins. In IMAQ Vision you can choose between three color sensitivity settings—low, medium, and high. **Low** divides the hue color space into seven sectors, giving a total of  $2 \times 7 + 2 = 16$  bins. **Medium** divides the hue color space into 14 sectors, giving a total of  $2 \times 14 + 2 = 30$  bins. **High** divides the hue color space into 28 sectors, giving a total of  $2 \times 28 + 2 = 58$  bins.

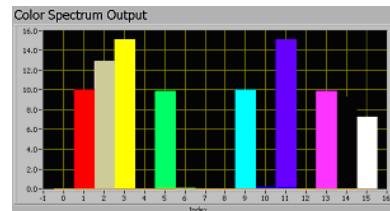
The value of each element in the color spectrum indicates the percentage of image pixels in each color bin. Once the number of bins is set according to the color sensitivity parameter, the machine vision software scans the image, counts the number of pixels that fall into each bin, and stores the ratio of the count and total number of pixels in the image in the appropriate element within the color spectrum array.

The software also applies a special adaptive learning algorithm to determine if pixels are either black or white before assigning it to a color bin. Figure 14-4b represents the low sensitivity color spectrum of Figure 14-4a. The height of each bar corresponds to the percentage of pixels in the image that fall into the corresponding bin.

The color spectrum contains useful information about the color distribution in the image. You can analyze the color spectrum to get information such as the most dominant color in the image which is the element with the highest value in the color spectrum. You can also use the array of the color spectrum to directly analyze the color distribution and for color matching applications.



a.



b.

**Figure 14-4.** Color Spectrum Associated with an Image

# Color Matching

---

Color matching quantifies which colors and how much of each color exist in a region of an image and uses this information to check if another image contains the same colors in the same ratio.

Use color matching to compare the color content of an image or regions within an image to a reference color information. With color matching you create an image or select regions in an image that contain the color information you want to use as a reference. The color information in the image may consist of one or more colors. The machine vision software then learns the three-dimensional color information in the image and represents it as a one-dimensional color spectrum. Your machine vision application compares the color information in the entire image or regions in the image to the learned color spectrum, calculating a score for each region. The score relates how closely the color information in the image region matches the information represented by the color spectrum.

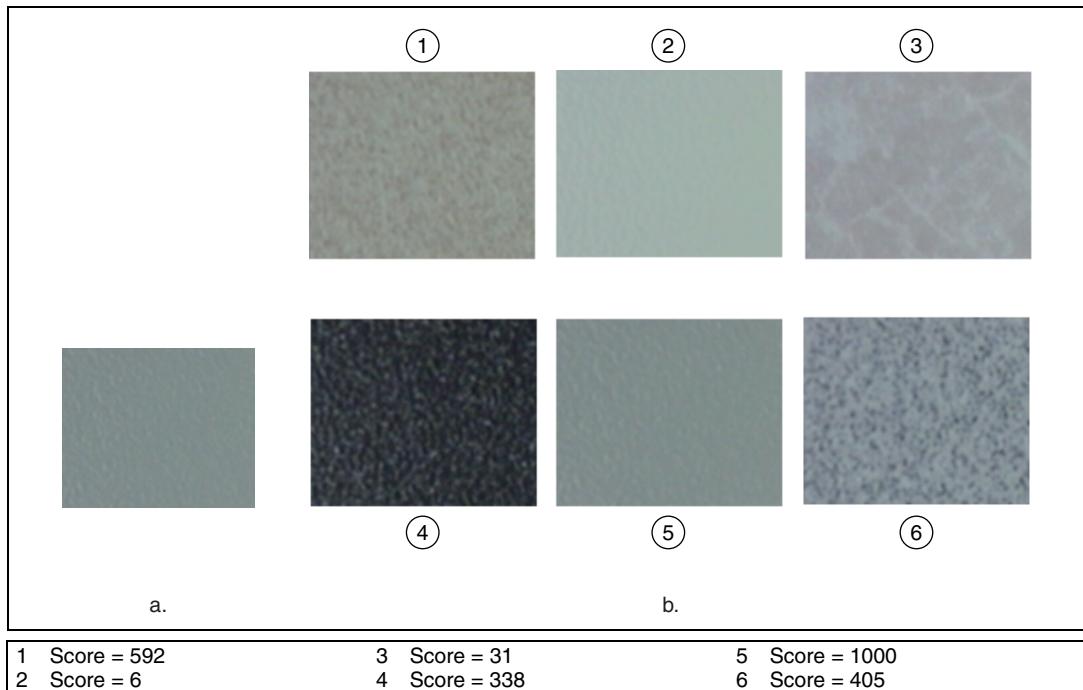
## When to Use

Color matching can be used for applications such as color identification, color inspection, color object location and other applications that require the comparison of color information to make decisions.

## Color Identification

Color identification identifies an object by comparing the color information in the image of the object to a database of reference colors which correspond to pre-defined object types. The object is assigned a label corresponding to the object type with closest reference color in the database. Use color matching to first learn the color information of all the pre-defined object types. The color spectrums associated with each of the pre-defined object types become the reference colors. Your machine vision application then uses color matching to compare the color information in the image of the object to the reference color spectrums. The object receives the label of the color spectrum with the highest match score.

Figure 14-5 shows an example of a tile identification application. Figure 14-5a shows the image of a tile that needs to be identified. Figure 14-5b shows the scores obtained using color matching with a set of the reference tiles.



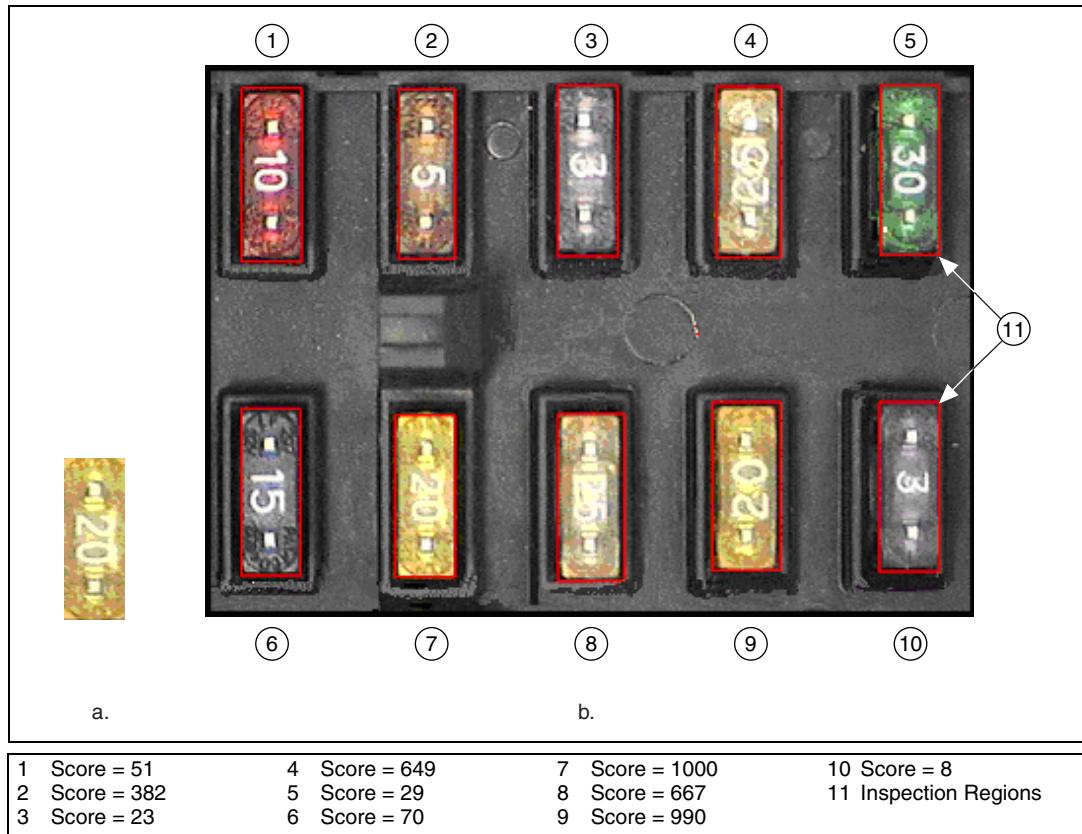
**Figure 14-5.** Color Matching

Use color matching to verify the presence of correct components in automotive assemblies. An example of a color identification task is to ensure that the color of the fabric in the interior of a car adheres to specifications.

## Color Inspection

Color inspection detects simple flaws such as missing or misplaced color components, defects on the surfaces of color objects or printing errors on color labels. You can use color matching for these applications if known regions of interest predefine the object or areas to be inspected in the image. You can define these regions or they can be the output of some other machine vision tool, such as pattern matching used to locate the components to be inspected.

The layout of the fuses in junction boxes in automotive assemblies is easily defined by regions of interest. Color matching determines if all of the fuses are present and in the correct locations. Figure 14-6 shows an example of a fuse box inspection application in which the exact location of the fuses in the image can be specified by regions of interest. Color matching compares the color of the fuse in each region to the color that is expected to be in that region.



**Figure 14-6.** Fuse Box Inspection Using Color Matching

Color matching can be used to inspect printed circuit boards containing a variety of components including diodes, resistors, integrated circuits, and capacitors. In a manufacturing environment, color matching can find flaws in a manufactured product when the flaws are accompanied by a color change.

## Color Matching Concepts

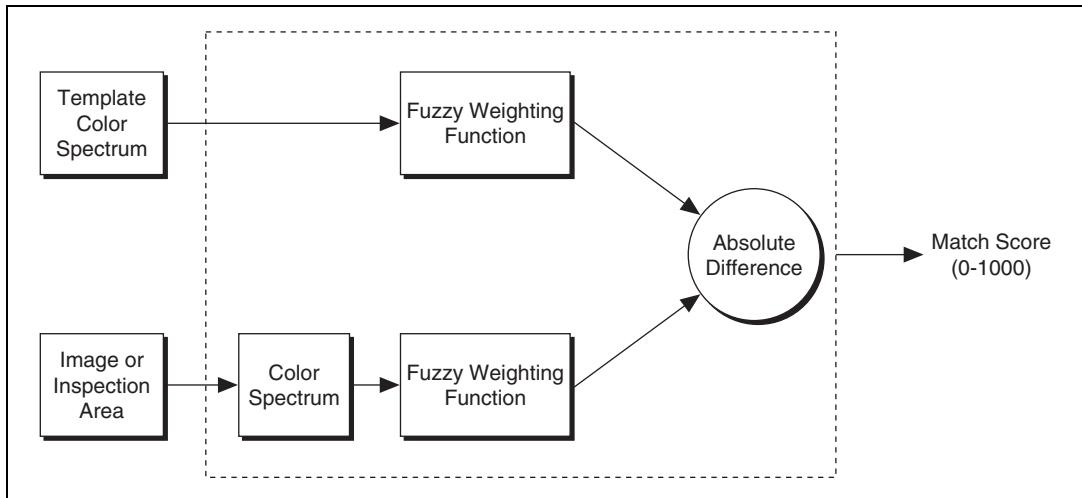
Color matching is performed in two steps. In the first step, the machine vision software learns a reference color distribution. In the second step, the software compares color information from other images to the reference image and returns a score as an indicator of similarity.

### Learning Color Distribution

The machine vision software learns a color distribution by generating a color spectrum. You provide the software with an image or regions in the image containing the color information that you want to use as a reference in your application. The machine vision software then generates a color spectrum based on the information you provide. The color spectrum becomes the basis of comparison during the matching phase.

### Comparing Color Distributions

During the matching phase, the color spectrum obtained from the target image or region in the target image is compared to the reference color spectrum taken during the learning step. A match score is computed based on the similarity between these two color spectrums using the Manhattan distance between two vectors. A fuzzy membership weighting function is applied to both the color spectrums before computing the distance between them. The weighting function compensates for some errors that may occur during the binning process in the color space. The fuzzy color comparison approach provides a robust and accurate quantitative match score. The match score, ranging from 0 to 1000, defines the similarity between the color spectrums. A score of zero represents no similarity between the color spectrums, while a score of 1000 represents a perfect match. Figure 14-7 shows the comparison process.

**Figure 14-7.** Comparing Two Spectrums for Similarity

## Color Location

Use color location to quickly locate known color regions in an image. With color location, you create a model or template that represents the colors that you are searching. Your machine vision application then searches for the model in each acquired image, and calculates a score for each match. The score indicates how closely the color information in the model matches the color information in the found regions.

### When to Use

Color can simplify a monochrome visual inspection problem by improving contrast or separating the object from the background. Color location algorithms provide a quick way to locate regions in an image with specific colors.

Use color location when your application:

- Requires the location and the number of regions in an image with their specific color information
- Relies on the cumulative color information in the region, instead of how the colors are arranged in the region
- Does not require the orientation of the region
- Does not require the location with sub-pixel accuracy

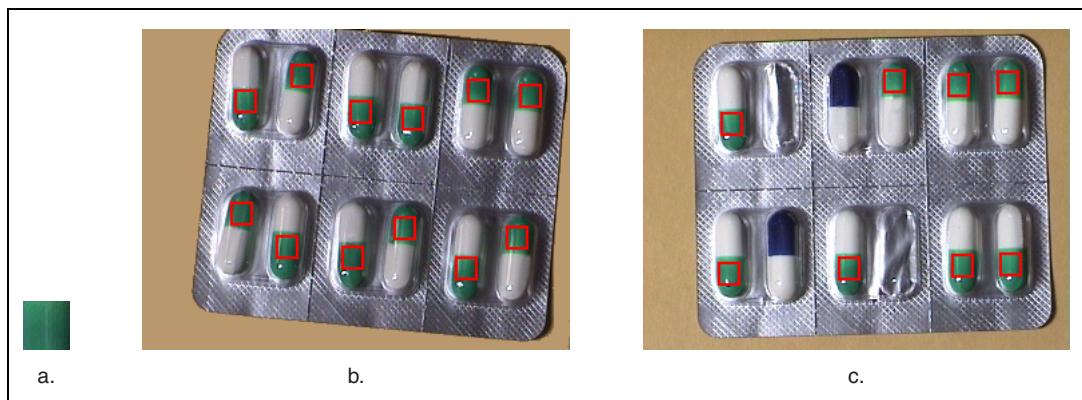
The color location tools in IMAQ Vision measure the similarity between an idealized representation of a feature, called a model, and a feature that may be present in an image. A feature for color location is defined as a region in an image with specific colors.

Color location is useful in many applications. Color location provides your application with information about the number of instances and locations of the template within an image. Use color location in the following general applications—inspection, identification, and sorting.

## Inspection

*Inspection* detects flaws like missing components, incorrect printing and incorrect fibers on textiles. A common pharmaceutical inspection application is inspecting a blister pack for the correct pills. Blister pack inspection involves checking that all the pills are of the correct type, which is easily performed by checking that all the pills have the same color information. Since your task is to determine if there are a fixed number of the correct pills in the pack, color location is a very effective tool.

Figure 14-8a shows the template image of the part of the pill that contains the color information that you want to locate. Figure 14-8b shows the pills located in a good blister pack. Figure 14-8c shows the pills located when a blister pack contains the wrong type of pills or missing pills. Since the exact locations of the pills is not necessary for the inspection, the number of matches returned by color location indicates whether a blister pack passes inspection.



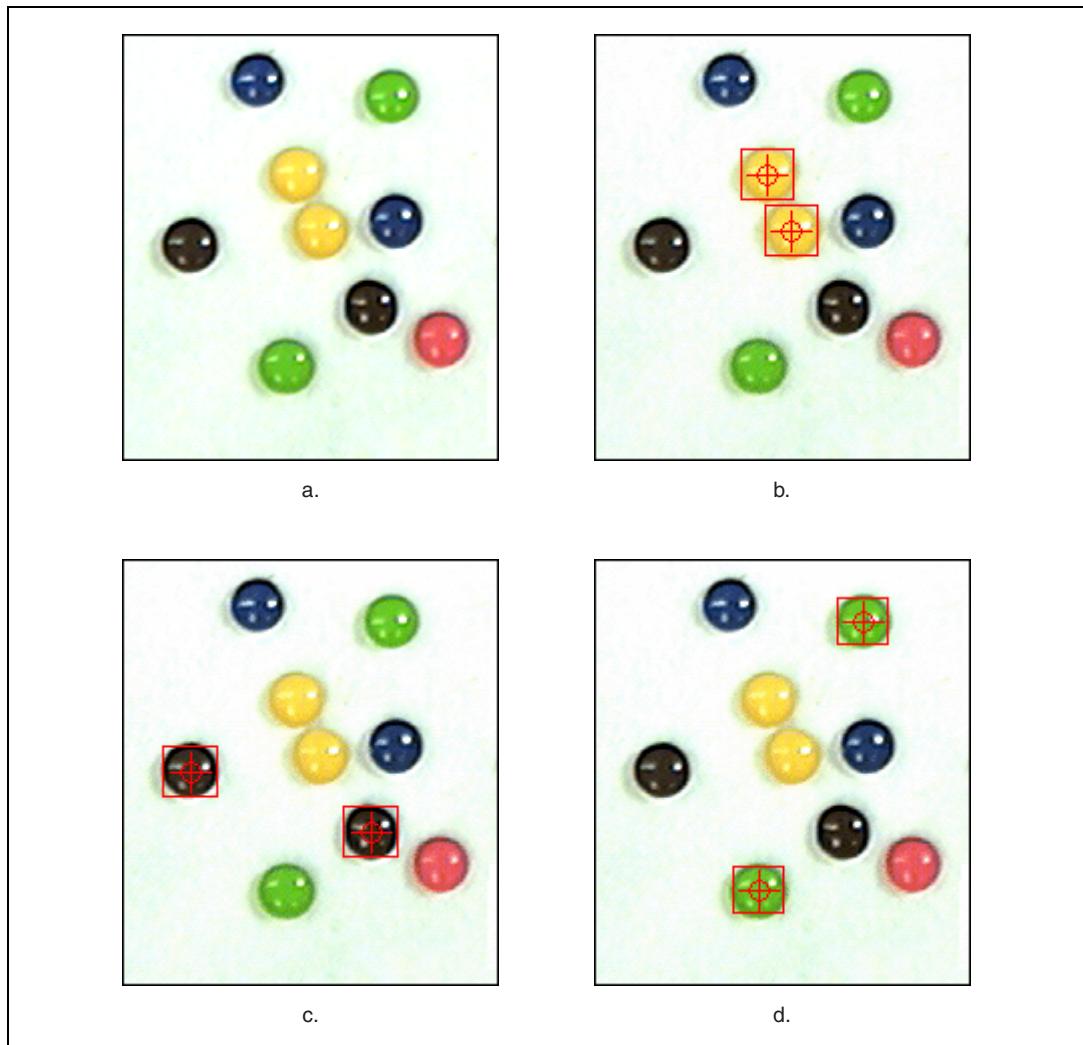
**Figure 14-8.** Blister Pack Inspection Using Color Matching

## Identification

Identification assigns a label to an object based on its features. In many applications, the color-coded identification marks are placed on the objects. In these applications, color matching locates the color code and identifies the object. In a spring identification application, different types of springs are identified by a collection of color marks painted on the coil. If you know the different types of color patches that are used to mark the springs, color location can find which color marks appear in the image. You can then use this information to identify the type of spring.

## Sorting

Sorting separates objects based on attributes such as color, size, and shape. In many applications, especially in the pharmaceutical and plastic industry, objects are sorted according to color, such as pills and plastic pellets. Figure 14-9 shows a simple example of how to sort different colored candies. Using color templates of the different candies in the image, color location quickly locates the positions of the different candies.



**Figure 14-9.** Sorting Candy by Color Information

## What to Expect from a Color Location Tool

In automated machine vision applications, the visual appearance of inspected materials or components changes because of factors such as orientation of the part, scale changes, and lighting changes. The color location tool maintains its ability to locate the reference patterns despite these changes. The color location tool provides accurate results during the following common situations: pattern orientation and multiple instances, ambient lighting conditions, and blur and noise conditions.

### Pattern Orientation and Multiple Instances

A color location tool locates the reference pattern in an image even if the pattern in the image is rotated or scaled. When a pattern is rotated or slightly scaled in the image, the color location tool can detect the following:

- The pattern in the image
- The position of the pattern in the image
- Multiple instances of the pattern in the image (if applicable)

Since color location only works on the color information of a region and does not use any kind of shape information from the template, it does not find the angle of the rotation of the match. It only locates the position of a region in the image whose size matches a template containing similar color information.

Refer to Figure 14-8 for an example of pattern orientation and multiple instances. Figure 14-8a shows a template image. Figures 14-8b and 14-8c show multiple shifted and rotated occurrences of the template.

## Ambient Lighting Conditions

The color location tool finds the reference pattern in an image under conditions of uniform changes in the lighting across the image. Color location also finds patterns under conditions of non-uniform light changes, such as shadows.

Figure 14-10 illustrates typical conditions under which the color location tool works correctly. Figure 14-10a shows the original template image. Figure 14-10b shows the same pattern under bright light. Figure 14-10c shows the pattern under poor lighting.



**Figure 14-10.** Examples of Lighting Conditions

## Blur and Noise Conditions

Color location finds patterns that have undergone some transformation because of blurring or noise. Blurring usually occurs because of incorrect focus or depth of field changes.

## Color Location Concept

Color location is built upon the color matching software to quickly locate regions with specific color information in an image. Refer to the [Color Matching](#) section of this chapter for more information.

The color location software extends the capabilities of color matching to applications in which the location of the objects in the image is unknown. Color location uses the color information in a template image to look for occurrences of the template in the search image. The basic operation moves the template across the image pixel by pixel and comparing the color

information at the current location in the image to the color information in the template using the color matching algorithm.

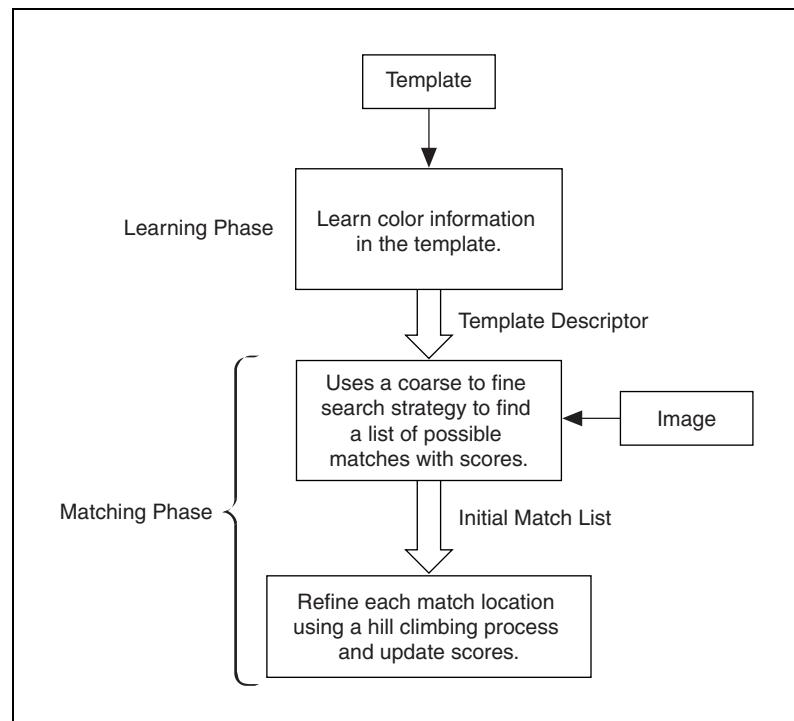
The color location process consists of two main steps—learning template information and searching for the template in an image. Figure 14-11 illustrates the general flow of the color location process. During the learning phase, the software extracts the color spectrum from the template image. This color spectrum is used to compare the color information of the template with the color information in the image.

During the search step, a region the size of the template is moved across the image pixel by pixel from the top of the image to the bottom. At each pixel, the function computes the color spectrum of the region under consideration. This color spectrum is then compared with the template's color spectrum to compute a match score.

The search step is divided into two phases. First, the software performs a coarse-to-fine search phase that identifies all possible locations, even those with very low match scores. The objective of this phase is to quickly find possible locations in the image that may be potential matches to the template information. Since stepping through the image pixel by pixel and computing match scores is time consuming, some techniques are used to speed up the search process. The following techniques allow for a quick search:

- Sub-sampling—When stepping through the image, the color information is taken from only a few sample points in the image to use for comparison with the template. This reduces the amount of data used to compute the color spectrum in the image, which speeds up the search process.
- Step size—Instead of moving the template across the image pixel by pixel, the search process skips a few pixels between each color comparison, speeding up the search process. The step size indicates the number of pixels to skip. For color location, the initial step size can be as large as half the size of the template.

The initial search phase generates a list of possible match locations in the image. In the second step, that list is searched for the location of the best match using a hill-climbing algorithm.



**Figure 14-11.** Overview of the Color Location Process

# Color Pattern Matching

---

Use color pattern matching to quickly locate known reference patterns, or fiducials, in a color image. With color pattern matching, you create a model or template that represents the object you are searching for. Then your machine vision application searches for the model in each acquired image, calculating a score for each match. The score indicates how closely the model matches the color pattern found. Use color pattern matching to locate reference patterns that are fully described by the color and spatial information in the pattern.

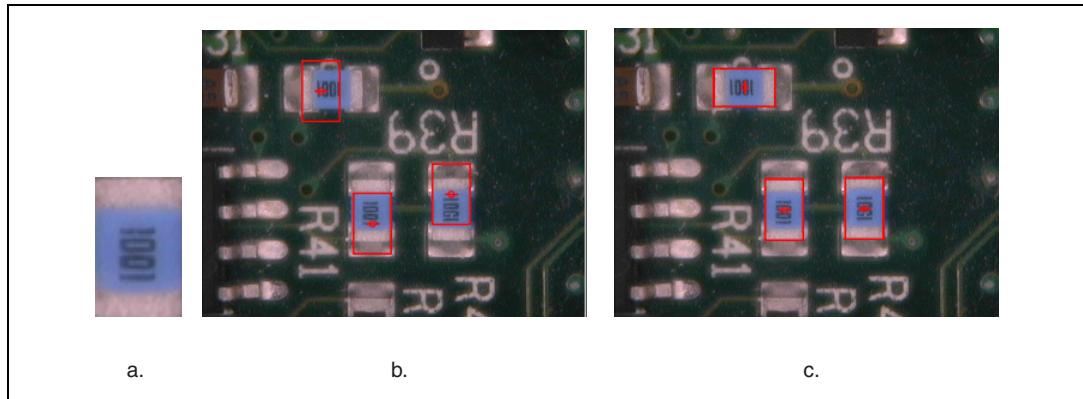
## When to Use

Grayscale, or monochrome, pattern matching is a well-established tool for alignment, gauging, and inspection applications. (See Chapter 12, *Pattern Matching*, for more information on pattern matching.) In all of these application areas, color simplifies a monochrome problem by improving contrast or separation of the object from the background. Color pattern matching algorithms provide a quick way to locate objects when color is present.

Use color pattern matching if:

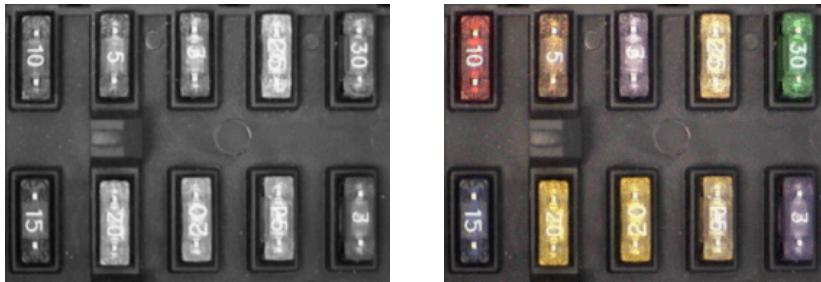
- The object you want to locate contains color information that is very different from the background, and you want to find the location of the object in the image very precisely. For these applications, color pattern matching provides a more accurate solution than color location—since color location does not use shape information during the search phase, finding the locations of the matches with pixel accuracy is very difficult.
- The object you want to locate has grayscale properties that are very difficult to characterize or that are very similar to other objects in the search image. In such cases, grayscale pattern matching may not give accurate results. If the object has some color information that differentiates it from the other objects in the scene, color provides the machine vision software with the additional information to locate the object.

Figure 14-12 illustrates the advantage of using color pattern matching over color location to locate the resistors in an image. Although color location finds the resistors in the image, the matches are not very accurate because they are limited to color information. Color pattern matching uses color matching first to locate the objects, and then pattern matching to refine the locations, providing more accurate results.



**Figure 14-12.** Comparison Between Color Location and Color Pattern Matching

Figure 14-13 shows the advantage of using color information when locating color-coded fuses on a fuse box. Figure 14-13a shows a grayscale image of the fuse box. In the image of the fuse box in Figure 14-13a, the grayscale pattern matching tool has difficulty clearly differentiating between fuse 20 and fuse 25 and will return close match scores because of similar grayscale intensities and the translucent nature of the fuses. In the color image. In the color image, Figure 14-13b, color helps to separate the fuses. The addition of color helps to improve the accuracy and reliability of the pattern matching tool.



**Figure 14-13.** Benefit of Adding Color to Fuse Box Inspection

The color pattern matching tools in IMAQ Vision measure the similarity between an idealized representation of a feature, called a model, and the feature that may be present in an image. A feature is defined as a specific pattern of color pixels in an image.

Color pattern matching is the key to many applications. Color pattern matching provides your application with information about the number of instances and location of the template within an image. Use color pattern matching in the following three general applications: gauging, inspection, and alignment.

## Gauging

Many gauging applications locate and then measure or gauge the distance between objects. Searching and finding a feature is the key processing task that determines the success of many gauging applications. If the components you want to gauge are uniquely identified by their color, then color pattern matching provides a fast way to locate the components.

## Inspection

Inspection detects simple flaws, such as missing parts or unreadable printing. A common application is inspecting the labels on consumer product bottles for printing defects. Since most of the labels are in color, color pattern matching is used to locate the labels in the image before a detailed inspection of the label is performed. The score returned by the color pattern matching tool can also be used to decide whether a label is acceptable.

## Alignment

Alignment determines the position and orientation of a known object by locating fiducials. Use the fiducials as points of reference on the object. Grayscale pattern matching is sufficient for most applications, but some alignment applications require color pattern matching for more reliable results.

## What to Expect from a Color Pattern Matching Tool

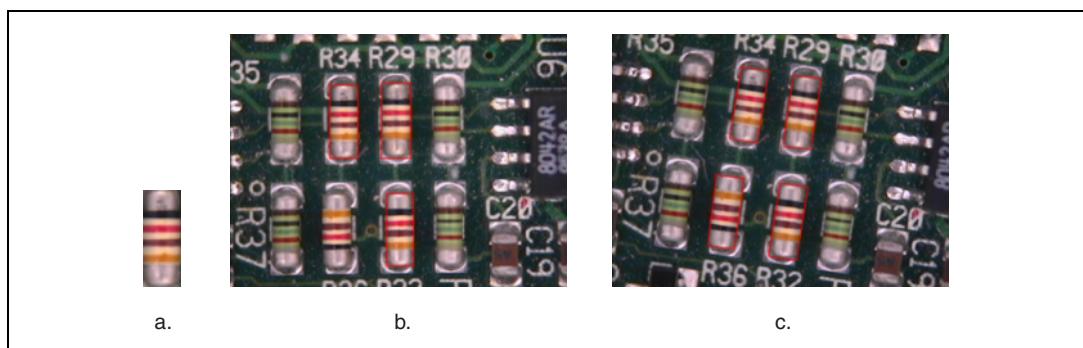
In automated machine vision applications, the visual appearance of materials or components under inspection can change due to factors such as orientation of the part, scale changes, and lighting changes. The color pattern matching tool maintains its ability to locate the reference patterns and gives accurate results despite these changes.

## Pattern Orientation and Multiple Instances

A color pattern matching tool locates the reference pattern in an image even when the pattern in the image is rotated and slightly scaled. When a pattern is rotated or scaled in the image, the color pattern matching tool detects the following:

- The pattern in the image
- The position of the pattern in the image
- The orientation of the pattern
- Multiple instances of the pattern in the image (if applicable)

Figure 14-14a shows a template image, or pattern. Figures 14-14b and 14-14c illustrate multiple occurrences of the template. Figure 14-14b shows the template shifted in the image. Figure 14-14c shows the template rotated in the image.

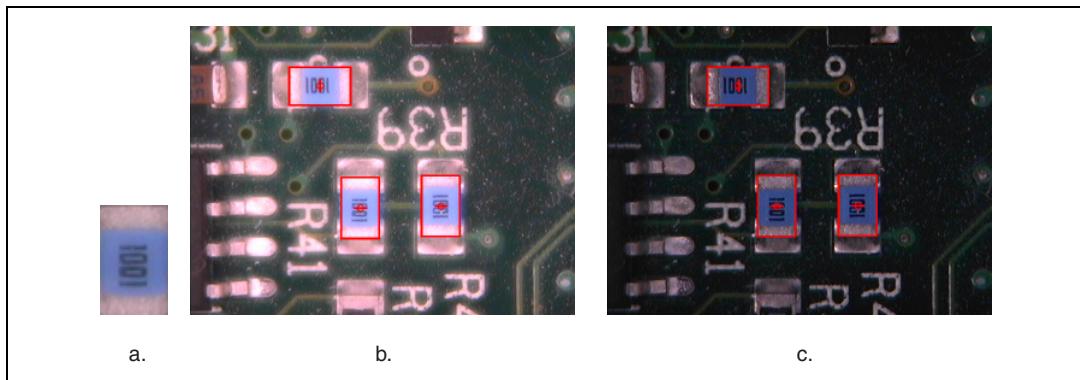


**Figure 14-14.** Pattern Orientation

## Ambient Lighting Conditions

The color pattern matching tool finds the reference pattern in an image under conditions of uniform changes in the lighting across the image. Because color analysis is more robust when dealing with variations in lighting than grayscale processing, color pattern matching performs better under conditions of non-uniform light changes, such as in the presence of shadows, than grayscale pattern matching.

Figure 14-15a shows the original template image. Figure 14-15b shows the same pattern under bright light. Figure 14-15c shows the pattern under poor lighting.



**Figure 14-15.** Examples of Lighting Conditions

## Blur and Noise Conditions

Color pattern matching finds patterns that have undergone some transformation because of blurring or noise. Blurring usually occurs because of incorrect focus or depth of field changes.

## Color Pattern Matching Concepts

Color pattern matching is a unique approach that combines color and spatial information to quickly find color patterns in an image. It uses the technologies behind color matching and grayscale pattern matching in a synergistic way to locate color patterns in color images.

## Color Matching and Color Location

Color matching compares the color content of an image or regions in an image to existing color information. The color information in the image may consist of one or more colors. To use color matching, define regions in an image that contain the color information you want to use as a reference. The machine vision software then learns the three-dimensional color information in the image and represents it as a one-dimensional color spectrum. Your machine vision application compares the color information in the entire image or regions in the image to the learned color spectrum, calculating a score for each region. This score relates how closely the color information in the image region matches the information represented by the color spectrum. To use color matching, you need to know the location of the objects in the image before performing the match.

The color location software extends the capabilities of color matching to applications where you do not know the location of the objects in the image. Color location uses the color information from a template image to look for occurrences of the template in the search image. The basic operation moves the template across the image pixel by pixel and compares the color information at the current location in the image to the color information in the template, using the color matching algorithm. Because searching an entire image for color matches is time consuming, the color location software uses some techniques to speed up the location process. A coarse-to-fine search strategy finds the rough locations of the matches in the image. A more refined search using a hill climbing algorithm is then performed around each match to get the accurate location of the match. Color location is an efficient way to look for occurrences of regions in an image with specific color attributes.

Refer to the [Color Matching](#) and [Color Location](#) sections of this chapter for more information.

## Grayscale Pattern Matching

Grayscale pattern matching methods used in IMAQ Vision incorporate image understanding techniques to interpret the template information and use that information to find the template in the image. Image understanding refers to image processing techniques that generate information about the features of a template image. These methods include the following:

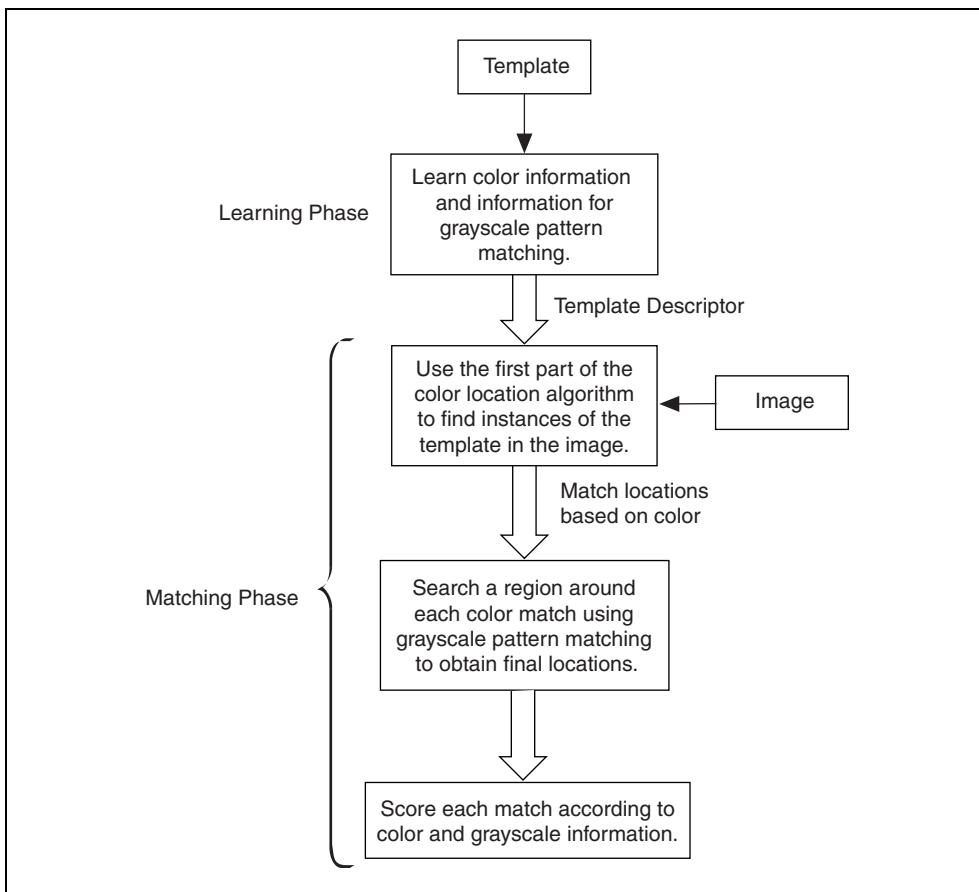
- Geometric modeling of images
- Efficient non-uniform sampling of images
- Extraction of rotation-independent template information

IMAQ Vision uses a combination of the edge information in the image and an intelligent image sampling technique to match patterns. The image's edge content provides information about the structure of the image in a compact form. The intelligent sampling technique extracts points from the template that represent the overall content of the image. The edge information and the smart sampling method reduce the inherently redundant information in an image and improve the speed and accuracy of the pattern matching tool. In cases where the pattern can be rotated in the image, a similar technique is used, but with specially chosen template pixels whose values (or relative change in values) reflect the rotation of the pattern. The result is fast and accurate grayscale pattern matching. IMAQ Vision pattern matching accurately locates objects in conditions where they vary in size ( $\pm 5\%$ ) and orientation (between  $0^\circ$  and  $360^\circ$ ) and when their appearance is degraded.

Refer to Chapter 12, [Pattern Matching](#), for more information on grayscale pattern matching.

## Combining Color Location and Grayscale Pattern Matching

Color pattern matching uses a combination of color location and grayscale pattern matching to search for the template. When you use color pattern matching to search for a template, the software uses the color information in the template to look for occurrences of the template in the image. The software then applies grayscale pattern matching in a region around each of these occurrences to find the exact position of the template in the image. Figure 14-16 shows the general flow of the color pattern matching algorithm. The size of the searchable region is determined by the software, based on the inputs you provide such as search strategy and color sensitivity.



**Figure 14-16.** Overview of the Color Pattern Matching Process

# Instrument Readers

This chapter contains information about instrument readers that read meters, liquid crystal displays (LCDs), and barcodes.

## Introduction

Instrument readers are functions you can use to accelerate the development of applications that require reading meters, seven segment displays, and barcodes.

## When to Use

Use instrument readers when you need to obtain information from images of simple meters, LCD displays, and barcodes.

## Meter Functions

Meter functions simplify and accelerate the development of applications that require reading values from meters or gauges. These functions provide high-level vision processes to extract the position of a meter or gauge needle.

You can use this information to build different applications such as the calibration of a gauge. Use the functions to compute the base of the needle and its extremities from an area of interest indicating the initial and the full-scale position of the needle. You then can use these VIs to read the position of the needle using parameters computed earlier.

The recognition process consists of two phases:

- A learning phase during which the user must specify the extremities of the needle
- An analysis phase during which the current position of the needle is determined

The meter functions are designed to work with meters or gauges that have either a dark needle on a light background or a light needle on a dark background.

## Meter Algorithm Limits

This section explains the limit conditions of the algorithm used for the meter functions. The algorithm is fairly insensitive to light variations.

The position of the base of the needle is very important in the detection process. Carefully draw the lines that indicate the initial and the full-scale position of the needle. The coordinates of the base and of the points of the arc curved by the tip of the needle are computed during the setup phase. These coordinates are used to read the meter during inspection.

## LCD Functions

LCD functions simplify and accelerate the development of applications that require reading values from seven-segment displays.

Use these functions to extract seven-segment digit information from an image.

The reading process consists of two phases:

- A learning phase during which the user specifies an area of interest in the image to locate the seven-segment display
- A reading phase during which the area specified by the user is analyzed to read the seven-segment digit

IMAQ Vision's LCD functions provide the high-level vision processes required for recognizing and reading seven-segment digit indicators. The VIs in this library are designed for seven-segment displays that use either LCDs or LEDs (seven segments composed of light-emitting diodes or electroluminescent indicators).

The functions in this library can perform the following tasks:

- Detect the area around each seven-segment digit from a rectangular area that contains multiple digits
- Read the value of a single digit
- Read the value, sign, and decimal separator of the displayed number

## LCD Algorithm Limits

Four factors can cause a bad detection:

- Very high horizontal or vertical light drift
- Very low contrast between the background and the segments
- Very high level of noise
- Very low resolution of the image

Each of these factors is quantified to indicate when the algorithm might not give accurate results.

Light drift is quantified by the difference between the average pixel values at the top left and the bottom right of the background of the LCD screen. Detection results might be inaccurate when light drift is greater than 90 in 8-bit images.

Contrast is measured as the difference between the average pixel values in a rectangular region in the background and a rectangular region in a segment. This difference must be greater than 30 in 8-bit images (256 gray levels) to obtain accurate results.

Noise is defined as the standard deviation of the pixel values contained in a rectangular region in the background. This value must be less than 15 for 8-bit images (256 gray levels) to obtain accurate results.

Each digit must be larger than  $18 \times 12$  pixels to obtain accurate results.

## Barcode

The barcode concept applies to applications that require reading values encoded into 1D barcodes. IMAQ Vision currently supports the following barcodes: Code 25, Code 39, Code 93, Code 128, EAN 8, EAN 13, Codabar, MSI, and UPC A.

The process used to recognize the barcodes consists of two phases:

- A learning phase in which the user specifies an area of interest in the image which helps to localize the region occupied by the barcode
- The recognition phase during which the region specified by the user is analyzed to decode the barcode

## Barcode Algorithm Limits

The following factors can cause errors in the decoding process:

- Very low resolution of the image
- Very high horizontal or vertical light drift
- The contrast along the bars of the image
- High level of noise

The limit conditions are different for barcodes that have two different widths of bars and spaces (Code 39, Codabar, Code 25, and MSI code) and for barcodes that have four different widths of bars and spaces (Code 93, Code 128, EAN 13, EAN 8, and UPC A).

The resolution of an image is determined by the width of the smallest bar and space. These widths must be at least 3 pixels for all barcodes.

Light drift is quantified by the difference between the average of the gray level of the left (upper) line and the right (bottom) line of the background of the barcode. Decoding inaccuracies can occur if the light drift is greater than 120 for barcodes with two different widths of bars and spaces and greater than 100 for barcodes with four different widths of bars and spaces.

In overexposed images, the gray levels of the wide and narrow bars in the barcode tend to differ. Decoding results may not be accurate when the difference in gray levels is less than 80 for barcodes with two different widths of bars and spaces, and less than 100 for barcodes with four different widths of bars and spaces.

Consider the difference in gray levels between the narrow bars and the wide bars. The narrow bars are scarcely visible. If this difference of gray level exceeds 115 on 8-bit images (256 gray levels) for barcodes with two different widths of bars and spaces and 100 for barcodes with four different widths of bars and spaces, the results may be inaccurate.

Noise is defined as the standard deviation of a rectangular region of interest drawn in the background. It must be less than 57 for barcodes with two different widths of bars and spaces and less than 27 for barcodes with four different widths of bars and spaces.

Reflections on the barcode can introduce errors in the value read from the barcode. Bars and spaces that are masked by the reflection produce errors.

# Kernels

A kernel is a structure that represents a pixel and its relationship to its neighbors.

## Predefined Gradient Kernels

The tables in this section list the predefined gradient kernels.

### Prewitt Filters

The *Prewitt filters* have the following kernels. The notations West (W), South (S), East (E), and North (N) indicate which edges of bright regions they outline.

**Table A-1.** Prewitt Filters

#0 W/Edge	#1 W/Image	#2 SW/Edge	#3 SW/Image
-1 0 1 -1 <b>0</b> 1 -1 0 1	-1 0 1 -1 <b>1</b> 1 -1 0 1	0 1 1 -1 <b>0</b> 1 -1 -1 0	0 1 1 -1 <b>1</b> 1 -1 -1 0
#4 S/Edge	#5 S/Image	#6 SE/Edge	#7 SE/Image
1 1 1 0 <b>0</b> 0 -1 -1 -1	1 1 1 0 <b>1</b> 0 -1 -1 -1	1 1 0 1 <b>0</b> -1 0 -1 -1	1 1 0 1 <b>1</b> -1 0 -1 -1
#8 E/Edge	#9 E/Image	#10 NE/Edge	#11 NE/Image
1 0 -1 1 <b>0</b> -1 1 0 -1	1 0 -1 1 <b>1</b> -1 1 0 -1	0 -1 -1 1 <b>0</b> -1 1 1 0	0 -1 -1 1 <b>1</b> -1 1 1 0
#12 N/Edge	#13 N/Image	#14 NW/Edge	#15 NW/Image
-1 -1 -1 0 <b>0</b> 0 1 1 1	-1 -1 -1 0 <b>1</b> 0 1 1 1	-1 -1 0 -1 <b>0</b> 1 0 1 1	-1 -1 0 -1 <b>1</b> 1 0 1 1

## Sobel Filters

The *Sobel filters* are very similar to the Prewitt filters, except that they highlight light intensity variations along a particular axis that is assigned a stronger weight. The Sobel filters have the following kernels. The notations West (W), South (S), East (E), and North (N) indicate which edges of bright regions they outline.

**Table A-2.** Sobel Filters

#16 W/Edge	#17 W/Image	#18 SW/Edge	#19 SW/Image
-1 0 1 -2 0 2 -1 0 1	-1 0 1 -2 1 2 -1 0 1	0 1 2 -1 0 1 -2 -1 0	0 1 2 -1 1 1 -2 -1 0
#20 S/Edge	#21 S/Image	#22 SE/Edge	#23 SE/Image
1 2 1 0 0 0 -1 -2 -1	1 2 1 0 1 0 -1 -2 -1	2 1 0 1 0 -1 0 -1 -2	2 1 0 1 1 -1 0 -1 -2
#24 E/Edge	#25 E/Image	#26 NE/Edge	#27 NE/Image
1 0 -1 2 0 -2 1 0 -1	1 0 -1 2 1 -2 1 0 -1	0 -1 -2 1 0 -1 2 1 0	0 -1 -2 1 1 -1 2 1 0
#28 N/Edge	#29 N/Image	#30 NW/Edge	#31 NW/Image
-1 -2 -1 0 0 0 1 2 1	-1 -2 -1 0 1 0 1 2 1	-2 -1 0 -1 0 1 0 1 2	-2 -1 0 -1 1 1 0 1 2

The following table lists the predefined gradient  $5 \times 5$  kernel.

**Table A-3.** Gradient  $5 \times 5$

#0 W/Edge	#1 W/Image	#2 SW/Edge	#3 SW/Image
0 -1 0 1 0	0 -1 0 1 0	0 0 1 1 1	0 0 1 1 1
-1 -2 0 2 1	-1 -2 0 2 1	0 0 2 2 1	0 0 2 2 1
-1 -2 0 2 1	-1 -2 1 2 1	-1 -2 0 2 1	-1 -2 1 2 1
-1 -2 0 2 1	-1 -2 0 2 1	-1 -2 -2 0 0	-1 -2 -2 0 0
0 -1 0 1 0	0 -1 0 1 0	-1 -1 -1 0 0	-1 -1 -1 0 0
#4 S/Edge	#5 S/Image	#6 SE/Edge	#7 SE/Image
0 1 1 1 0	0 1 1 1 0	1 1 1 0 0	1 1 1 0 0
1 2 2 2 1	1 2 2 2 1	1 2 2 0 0	1 2 2 0 0
0 0 0 0 0	0 0 1 0 0	1 2 0 -2 -1	1 2 1 -2 -1
-1 -2 -2 -2 -1	-1 -2 -2 -2 -1	0 0 -2 -2 -1	0 0 -2 -2 -1
0 -1 -1 -1 0	0 -1 -1 -1 0	0 0 -1 -1 -1	0 0 -1 -1 -1
#8 E/Edge	#9 E/Image	#10 NE/Edge	#11 NE/Image
0 1 0 -1 0	0 1 0 -1 0	0 0 -1 -1 -1	0 0 -1 -1 -1
1 2 0 -2 -1	1 2 0 -2 -1	0 0 -2 -2 -1	0 0 -2 -2 -1
1 2 0 -2 -1	1 2 1 -2 -1	1 2 0 -2 -1	1 2 1 -2 -1
1 2 0 -2 -1	1 2 0 -2 -1	1 2 2 0 0	1 2 2 0 0
0 1 0 -1 0	0 1 0 -1 0	1 1 1 0 0	1 1 1 0 0
#12 N/Edge	#13 N/Image	#14 NW/Edge	#15 NW/Image
0 -1 -1 -1 0	0 -1 -1 -1 0	-1 -1 -1 0 0	-1 -1 -1 0 0
-1 -2 -2 -2 -1	-1 -2 -2 -2 -1	-1 -2 -2 0 0	-1 -2 -2 0 0
0 0 0 0 0	0 0 1 0 0	-1 -2 0 2 1	-1 -2 1 2 1
1 2 2 2 1	1 2 2 2 1	0 0 2 2 1	0 0 2 2 1
0 1 1 1 0	0 1 1 1 0	0 0 1 1 1	0 0 1 1 1

The following table lists the predefined gradient  $7 \times 7$  kernel.

**Table A-4.** Gradient  $7 \times 7$ 

#0 W/Edge	#1 W/Image	#2 S/Edge	#3 S/Image
0 -1 -1 0 1 1 0	0 -1 -1 0 1 1 0	0 1 1 1 1 1 0	0 1 1 1 1 1 0
-1 -2 -2 0 2 2 1	-1 -2 -2 0 2 2 1	1 2 2 2 2 2 1	1 2 2 2 2 2 1
-1 -2 -3 0 3 2 1	-1 -2 -3 0 3 2 1	1 2 3 3 3 2 1	1 2 3 3 3 2 1
-1 -2 -3 0 3 2 1	-1 -2 -3 1 3 2 1	0 0 0 0 0 0 0	0 0 0 1 0 0 0
-1 -2 -3 0 3 2 1	-1 -2 -3 0 3 2 1	-1 -2 -3 -3 -3 -2 -1	-1 -2 -3 -3 -3 -2 -1
-1 -2 -2 0 2 2 1	-1 -2 -2 0 2 2 1	-1 -2 -2 -2 -2 -2 -1	-1 -2 -2 -2 -2 -2 -1
0 -1 -1 0 1 1 0	0 -1 -1 0 1 1 0	0 -1 -1 -1 -1 -1 0	0 -1 -1 -1 -1 -1 0
#4 E/Edge	#5 E/Image	#6 N/Edge	#7 N/Image
0 1 1 0 -1 -1 0	0 1 1 0 -1 -1 0	0 -1 -1 -1 -1 -1 0	0 -1 -1 -1 -1 -1 0
1 2 2 0 -2 -2 -1	1 2 2 0 -2 -2 -1	-1 -2 -2 -2 -2 -2 -1	-1 -2 -2 -2 -2 -2 -1
1 2 3 0 -3 -2 -1	1 2 3 0 -3 -2 -1	-1 -2 -3 -3 -3 -2 -1	-1 -2 -3 -3 -3 -2 -1
1 2 3 0 -3 -2 -1	1 2 3 1 -3 -2 -1	0 0 0 0 0 0 0	0 0 0 1 0 0 0
1 2 3 0 -3 -2 -1	1 2 3 0 -3 -2 -1	1 2 3 3 3 2 1	1 2 3 3 3 2 1
1 2 2 0 -2 -2 -1	1 2 2 0 -2 -2 -1	1 2 2 2 2 2 1	1 2 2 2 2 2 1
0 1 1 0 -1 -1 0	0 1 1 0 -1 -1 0	0 1 1 1 1 1 0	0 1 1 1 1 1 0

## Predefined Laplacian Kernels

The following tables list the predefined Laplacian kernels.

**Table A-5.** Laplacian  $3 \times 3$

#0 Contour 4	#1 +Image×1	#2 +Image×2
0 -1 0 -1 <b>4</b> -1 0 -1 0	0 -1 0 -1 <b>5</b> -1 0 -1 0	0 -1 0 -1 <b>6</b> -1 0 -1 0
#3 Contour 8	#4 +Image×1	#5 +Image×2
-1 -1 -1 -1 <b>8</b> -1 -1 -1 -1	-1 -1 -1 -1 <b>9</b> -1 -1 -1 -1	-1 -1 -1 -1 <b>10</b> -1 -1 -1 -1
#6 Contour 12	#7 +Image×1	
-1 -2 -1 -2 <b>12</b> -2 -1 -2 -1	-1 -2 -1 -2 <b>13</b> -2 -1 -2 -1	

**Table A-6.** Laplacian  $5 \times 5$

#0 Contour 24	#1 +Image×1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 <b>24</b> -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 <b>25</b> -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

**Table A-7.** Laplacian  $7 \times 7$

#0 Contour 48	#1 +Image×1
-1 <b>48</b> -1	-1 <b>49</b> -1

## Predefined Smoothing Kernels

The following tables list the predefined smoothing kernels.

**Table A-8.** Smoothing  $3 \times 3$

0	1	0	0	1	0	0	2	0	0	4	0
1	<b>0</b>	1	1	<b>1</b>	1	2	<b>1</b>	2	4	<b>1</b>	4
0	1	0	0	1	0	0	2	0	0	4	0
1	1	1	1	1	1	2	2	2	4	4	4
1	<b>0</b>	1	1	<b>1</b>	1	2	<b>1</b>	2	4	<b>1</b>	4
1	1	1	1	1	1	2	2	2	4	4	4

**Table A-9.** Smoothing  $5 \times 5$

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	<b>0</b>	1	1	1	1	<b>1</b>	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

**Table A-10.** Smoothing  $7 \times 7$

1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	<b>0</b>	1	1	1	1	<b>1</b>	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1

## Predefined Gaussian Kernels

The following tables list the predefined Gaussian kernels.

**Table A-11.** Gaussian  $3 \times 3$

0	1	0	0	1	0	1	1	1
1	<b>2</b>	1	1	<b>4</b>	1	1	<b>2</b>	1
0	1	0	0	1	0	1	1	1
1	1	1	1	2	1	1	4	1
1	<b>4</b>	1	2	<b>4</b>	2	4	<b>16</b>	4
1	1	1	1	2	1	1	4	1

**Table A-12.** Gaussian  $5 \times 5$

1	2	4	2	1
2	4	8	4	2
4	8	<b>16</b>	8	4
2	4	8	4	2
1	2	4	2	1

**Table A-13.** Gaussian  $7 \times 7$

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	<b>16</b>	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

# Technical Support Resources

## Web Support

---

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of [ni.com](http://ni.com).

## NI Developer Zone

---

The NI Developer Zone at [ni.com/zone](http://ni.com/zone) is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

## Customer Education

---

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of [ni.com](http://ni.com) for online course schedules, syllabi, training centers, and class registration.

## System Integration

---

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of [ni.com](http://ni.com).

## **Worldwide Support**

---

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of [ni.com](http://ni.com). Branch office Web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.

# Glossary

---

Prefix	Meaning	Value
p-	pico-	$10^{-12}$
n-	nano-	$10^{-9}$
$\mu$ -	micro-	$10^{-6}$
m-	milli-	$10^{-3}$
k-	kilo-	$10^3$
M-	mega-	$10^6$
G-	giga-	$10^9$
t-	tera-	$10^{12}$

## Numbers/Symbols

- 1D                    One-dimensional.
- 2D                    Two-dimensional.
- 3D                    Three-dimensional.
- 3D view              Displays the light intensity of an image in a three-dimensional coordinate system, where the spatial coordinates of the image form two dimensions and the light intensity forms the third dimension.

## A

- AIPD                National Instruments internal image file format used for saving complex images and calibration information associated with an image (extension APD).
- alignment            The process by which a machine vision application determines the location, orientation, and scale of a part being inspected.

alpha channel	Channel used to code extra information, such as gamma correction, about a color image. The alpha channel is stored as the first byte in the four-byte representation of an RGB pixel.
area	A rectangular portion of an acquisition window or frame that is controlled and defined by software.
area threshold	Detects objects based on their size, which can fall within a user-specified range.
arithmetic operators	The image operations multiply, divide, add, subtract, and remainder.
array	Ordered, indexed set of data elements of the same type.
auto-median function	A function that uses dual combinations of opening and closing operations to smooth the boundaries of objects.

## B

b	Bit. One binary digit, either 0 or 1.
B	Byte. Eight related bits of data, an eight-bit binary number. Also denotes the amount of memory required to store one byte of data.
barycenter	The grayscale value representing the centroid of the range of an image's grayscale values in the image histogram.
binary image	An image in which the objects usually have a pixel intensity of 1 (or 255) and the background has a pixel intensity of 0.
binary morphology	Functions that perform morphological operations on a binary image.
binary threshold	Separation of an image into objects of interest (assigned a pixel value of 1) and background (assigned pixel values of 0) based on the intensities of the image pixels.
bit depth	The number of bits ( $n$ ) used to encode the value of a pixel. For a given $n$ , a pixel can take $2^n$ different values. For example, if $n$ equals 8-bits, a pixel can take 256 different values ranging from 0 to 255. If $n$ equals 16 bits, a pixel can take 65,536 different values ranging from 0 to 65,535 or -32,768 to 32,767.
black reference level	The level that represents the darkest an image can get. <i>See also white reference level.</i>

blob	Binary large object. A connected region or grouping of pixels in an image in which all pixels have the same intensity level.
blob analysis	A series of processing operations and analysis functions that produce some information about the blobs in an image.
blurring	Reduces the amount of detail in an image. Blurring commonly occurs because the camera is out of focus. You can blur an image intentionally by applying a lowpass frequency filter.
BMP	Bitmap. Image file format commonly used for 8-bit and color images (extension BMP).
border function	Removes objects (or particles) in a binary image that touch the image border.
brightness	(1) A constant added to the red, green, and blue components of a color pixel during the color decoding process. (2) The perception by which white objects are distinguished from gray and light objects from dark objects.
buffer	Temporary storage for acquired data.

## C

caliper	(1) A function in IMAQ Vision Builder that calculates distances, angles, circular fits, and the center of mass based on positions given by edge detection, particle analysis, centroid, and search functions. (2) A measurement function that finds edge pairs along a specified path in the image. This function performs an edge extraction and then finds edge pairs based on specified criteria such as the distance between the leading and trailing edges, edge contrasts, and so forth.
center of mass	The point on an object where all the mass of the object could be concentrated without changing the first moment of the object about any axis
character recognition	The ability of a machine to read human-readable text.
chroma	The color information in a video signal.
chromaticity	The combination of hue and saturation. The relationship between chromaticity and brightness characterizes a color.
chrominance	<i>See</i> chroma.

circle function	Detects circular objects in a binary image.
closing	A dilation followed by an erosion. A closing fills small holes in objects and smooths the boundaries of objects.
clustering	Technique where the image is sorted within a discrete number of classes corresponding to the number of phases perceived in an image. The gray values and a barycenter are determined for each class. This process is repeated until a value is obtained that represents the center of mass for each phase or class.
CLUT	Color lookup table. Table for converting the value of a pixel in an image into a red, green, and blue (RGB) intensity.
color images	Images containing color information, usually encoded in the RGB form.
color space	The mathematical representation for a color. For example, color can be described in terms of red, green, and blue; hue, saturation, and luminance; or hue, saturation, and intensity.
complex image	Stores information obtained from the FFT of an image. The complex numbers that compose the FFT plane are encoded in 64-bit floating-point values: 32 bits for the real part and 32 bits for the imaginary part.
connectivity	Defines which of the surrounding pixels of a given pixel constitute its neighborhood.
connectivity-4	Only pixels adjacent in the horizontal and vertical directions are considered neighbors.
connectivity-8	All adjacent pixels are considered neighbors.
contrast	A constant multiplication factor applied to the luma and chroma components of a color pixel in the color decoding process.
convex function	Computes the convex regions of objects in a binary image.
convex hull	The smallest convex polygon that can encapsulate a particle.
convolution	<i>See</i> linear filter.

**convolution kernel** 2D matrices (or templates) used to represent the filter in the filtering process. The contents of these kernels are a discrete two-dimensional representation of the impulse response of the filter that they represent.

**cross correlation** The most common way to perform pattern matching.

## D

**Danielsson function** Similar to the distance functions, but with more accurate results.

**dB** Decibel. The unit for expressing a logarithmic measure of the ratio of two signal levels:  $\text{dB} = 20\log_{10} V_1/V_2$ , for signals in volts.

**default setting** A default parameter value recorded in the driver. In many cases, the default input of a control is a certain value (often 0).

**definition** The number of values a pixel can take on, which is the number of colors or shades that you can see in the image.

**dendrite** Branches of the skeleton of an object.

**densitometry** Determination of optical or photographic density.

**density function** For each gray level in a linear histogram, the function gives the number of pixels in the image that have the same gray level.

**differentiation filter** Extracts the contours (edge detection) in gray level.

**digital image** An image  $f(x, y)$  that has been converted into a discrete number of pixels. Both spatial coordinates and brightness are specified.

**dilation** Increases the size of an object along its boundary and removes tiny holes in the object.

**distance calibration** Determination of the physical dimensions of a pixel by defining the physical dimensions of a line in the image.

**distance function** Assigns to each pixel in an object a gray-level value equal to its shortest Euclidean distance from the border of the object.

**driver** Software that controls a specific hardware device, such as an IMAQ or DAQ device.

**dynamic range** The ratio of the largest signal level a circuit can handle to the smallest signal level it can handle (usually taken to be the noise level), normally expressed in decibels.

## E

**edge** Defined by a sharp change (transition) in the pixel intensities in an image or along an array of pixels.

**edge contrast** The difference between the average pixel intensity before and the average pixel intensity after the edge.

**edge detection** Any of several techniques to identify the edges of objects in an image.

**edge hysteresis** The difference in threshold level between a rising and a falling edge.

**edge steepness** The number of pixels that corresponds to the slope or transition area of an edge.

**energy center** The center of mass of a grayscale image. *See* center of mass.

**entropy** A measure of the randomness in an image. An image with high entropy contains more pixel value variation than an image with low entropy.

**equalize function** *See* histogram equalization.

**erosion** Reduces the size of an object along its boundary and eliminates isolated points in the image.

**Euclidean distance** The shortest distance between two points in a Cartesian system.

**exponential and gamma corrections** Expand the high gray-level information in an image while suppressing low gray-level information.

**exponential function** Decreases brightness and increases contrast in bright regions of an image, and decreases contrast in dark regions of an image.

**F**

FFT	Fast Fourier Transform. A method used to compute the Fourier transform of an image.
fiducial	A reference pattern on a part that helps a machine vision application find the part's location and orientation in an image.
form	Window or area on the screen on which you place controls and indicators to create the user interface for your program.
Fourier spectrum	The magnitude information of the Fourier transform of an image.
Fourier transform	Transforms an image from the spatial domain to the frequency domain.
frequency filters	Counterparts of spatial filters in the frequency domain. For images, frequency information is in the form of spatial frequency.
ft	Feet.
function	A set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed.

**G**

gamma	The nonlinear change in the difference between the video signal's brightness level and the voltage level needed to produce that brightness.
gauging	Measurement of an object or distances between objects.
Gaussian filter	A filter similar to the smoothing filter, but using a Gaussian kernel in the filter operation. The blurring in a Gaussian filter is more gentle than a smoothing filter.
gradient convolution filter	<i>See</i> gradient filter.
gradient filter	Extracts the contours (edge detection) in gray-level values. Gradient filters include the Prewitt and Sobel filters.
gray level	The brightness of a pixel in an image.
gray-level dilation	Increases the brightness of pixels in an image that are surrounded by other pixels with a higher intensity.

gray-level erosion Reduces the brightness of pixels in an image that are surrounded by other pixels with a lower intensity.

grayscale image An image with monochrome information.

grayscale morphology Functions that perform morphological operations on a gray-level image.

## H

h Hour.

highpass attenuation Inverse of lowpass attenuation.

highpass FFT filter Removes or attenuates low frequencies present in the FFT domain of an image.

highpass filter Emphasizes the intensity variations in an image, detects edges (or object boundaries), and enhances fine details in an image.

highpass frequency filter Attenuates or removes (truncates) low frequencies present in the frequency domain of the image. A highpass frequency filter suppresses information related to slow variations of light intensities in the spatial image.

highpass truncation Inverse of lowpass truncation.

histogram Indicates the quantitative distribution of the pixels of an image per gray-level value.

histogram equalization Transforms the gray-level values of the pixels of an image to occupy the entire range (0 to 255 in an 8-bit image) of the histogram, increasing the contrast of the image.

histogram inversion Finds the photometric negative of an image. The histogram of a reversed image is equal to the original histogram flipped horizontally around the center of the histogram.

histograph In LabVIEW, a histogram that can be wired directly into a graph.

hit-miss function Locates objects in the image similar to the pattern defined in the structuring element.

hole filling function Fills all holes in objects that are present in a binary image.

HSI Color encoding scheme in Hue, Saturation, and Intensity.

HSL	Color encoding scheme using Hue, Saturation, and Luminance information where each image in the pixel is encoded using 32 bits: 8 bits for hue, 8 bits for saturation, 8 bits for luminance, and 8 unused bits.
HSV	Color encoding scheme in Hue, Saturation, and Value.
hue	Represents the dominant color of a pixel. The hue function is a continuous function that covers all the possible colors generated using the R, G, and B primaries. <i>See also</i> RGB.
hue offset angle	The value added to all hue values so that the discontinuity occurs outside the values of interest during analysis.
Hz	Hertz. Frequency in units of 1/second.

**I**

I/O	Input/output. The transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces.
image	A two-dimensional light intensity function $f(x, y)$ where $x$ and $y$ denote spatial coordinates and the value $f$ at any point $(x, y)$ is proportional to the brightness at that point.
image border	A user-defined region of pixels surrounding an image. Functions that process pixels based on the value of the pixel neighbors require image borders.
Image Browser	An image that contains thumbnails of images to analyze or process in a vision application.
image buffer	Memory location used to store images.
image definition	<i>See</i> pixel depth.
image enhancement	The process of improving the quality of an image that you acquire from a sensor in terms of signal-to-noise ratio, image contrast, edge definition, and so on.
image file	A file containing pixel data and additional information about the image.
image format	Defines how an image is stored in a file. Usually composed of a header followed by the pixel data.

image mask	A binary image that isolates parts of a source image for further processing. A pixel in the source image is processed if its corresponding mask pixel has a non-zero value. A source pixel whose corresponding mask pixel has a value of 0 is left unchanged.
image palette	The gradation of colors used to display an image on screen, usually defined by a color lookup table.
image processing	Encompasses various processes and analysis functions that you can apply to an image.
image source	Original input image.
image understanding	A technique that interprets the content of the image at a symbolic level rather than a pixel level.
image visualization	The presentation (display) of an image (image data) to the user.
imaging	Any process of acquiring and displaying images and analyzing image data.
IMAQ	Image Acquisition.
in.	Inches.
INL	Integral nonlinearity. A measure in LSB of the worst-case deviation from the ideal A/D or D/A transfer characteristic of the analog I/O circuitry.
inner gradient	Finds the inner boundary of objects.
inspection	The process by which parts are tested for simple defects such as missing parts or cracks on part surfaces.
inspection function	Analyzes groups of pixels within an image and returns information about the size, shape, position, and pixel connectivity. Typical applications include quality of parts, analyzing defects, locating objects, and sorting objects.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that control specific plug-in computer boards. Instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW.
intensity	The sum of the Red, Green, and Blue primary colors divided by three. $(Red+Green+Blue)/3$

intensity calibration	Assigning user-defined quantities such as optical densities or concentrations to the gray-level values in an image.
intensity profile	The gray-level distribution of the pixels along an ROI in an image.
intensity range	Defines the range of gray-level values in an object of an image.
intensity threshold	Characterizes an object based on the range of gray-level values in the object. If the intensity range of the object falls within the user-specified range, it is considered an object. Otherwise it is considered part of the background.
interpolation	The technique used to find values in between known values when resampling an image or array of pixels.
IRE	A relative unit of measure (named for the Institute of Radio Engineers). 0 IRE corresponds to the blanking level of a video signal, 100 IRE to the white level. Note that for CCIR/PAL video, the black level is equal to the blanking level or 0 IRE, while for RS-170/NTSC video, the black level is at 7.5 IRE.

**J**

JPEG	Joint Photographic Experts Group. Image file format for storing 8-bit and color images with lossy compression (extension JPG).
------	--

**K**

k	Kilo. The standard metric prefix for 1,000, or $10^3$ , used with units of measure such as volts, hertz, and meters.
K	Kilo. The prefix for 1,024, or $2^{10}$ , used with B in quantifying data or computer memory.
kernel	Structure that represents a pixel and its relationship to its neighbors. The relationship is specified by weighted coefficients of each neighbor.

## L

labeling	The process by which each object in a binary image is assigned a unique value. This process is useful for identifying the number of objects in the image and giving each object a unique identity.
LabVIEW	Laboratory Virtual Instrument Engineering Workbench. Program development environment application based on the programming language G used commonly for test and measurement applications.
Laplacian filter	Extracts the contours of objects in the image by highlighting the variation of light intensity surrounding a pixel.
line gauge	Measures the distance between selected edges with high-precision subpixel accuracy along a line in an image. For example, this function can be used to measure distances between points and edges. This function also can step and repeat its measurements across the image.
line profile	Represents the gray-level distribution along a line of pixels in an image.
linear filter	A special algorithm that calculates the value of a pixel based on its own pixel value as well as the pixel values of its neighbors. The sum of this calculation is divided by the sum of the elements in the matrix to obtain a new pixel value.
logarithmic and inverse gamma corrections	Expand low gray-level information in an image while compressing information from the high gray-level ranges.
logarithmic function	Increases the brightness and contrast in dark regions of an image and decreases the contrast in bright regions of the image.
logic operators	The image operations AND, NAND, OR, XOR, NOR, XNOR, difference, mask, mean, max, and min.
lossless compression	Compression in which the decompressed image is identical to the original image.
lossy compression	Compression in which the decompressed image is visually similar but not identical to the original image.
lowpass attenuation	Applies a linear attenuation to the frequencies in an image, with no attenuation at the lowest frequency and full attenuation at the highest frequency.

lowpass FFT filter	Removes or attenuates high frequencies present in the FFT domain of an image.
lowpass filter	Attenuates intensity variations in an image. You can use these filters to smooth an image by eliminating fine details and blurring edges.
lowpass frequency filter	Attenuates high frequencies present in the frequency domain of the image. A lowpass frequency filter suppresses information related to fast variations of light intensities in the spatial image.
lowpass truncation	Removes all frequency information above a certain frequency.
LSB	Least significant bit.
L-skeleton function	Uses an L-shaped structuring element in the skeleton function.
luma	The brightness information in the video picture. The luma signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.
luminance	<i>See</i> luma.
LUT	Lookup table. Table containing values used to transform the gray-level values of an image. For each gray-level value in the image, the corresponding new value is obtained from the lookup table.

## M

m	Meters.
M	(1) Mega, the standard metric prefix for 1 million or $10^6$ , when used with units of measure such as volts and hertz (2) Mega, the prefix for 1,048,576, or $2^{20}$ , when used with B to quantify data or computer memory.
machine vision	An automated application that performs a set of visual inspection tasks.
mask FFT filter	Removes frequencies contained in a mask (range) specified by the user.
match score	A number ranging from 0 to 1000 that indicates how closely an acquired image matches the template image. A match score of 1000 indicates a perfect match. A match score of 0 indicates no match.
MB	Megabyte of memory.

median filter	A lowpass filter that assigns to each pixel the median value of its neighbors. This filter effectively removes isolated pixels without blurring the contours of objects.
memory buffer	<i>See</i> buffer.
mile	An Imperial unit of length equal to 5,280 feet or 1,609.344 meters. Also known as a statute mile to discern from a nautical mile. <i>See also</i> nautical mile.
MMX	Multimedia Extensions. Intel chip-based technology that allows parallel operations on integers, which results in accelerated processing of 8-bit images.
morphological transformations	Extract and alter the structure of objects in an image. You can use these transformations for expanding (dilating) or reducing (eroding) objects, filling holes, closing inclusions, or smoothing borders. They are used primarily to delineate objects and prepare them for quantitative inspection analysis.
MSB	Most significant bit.
M-skeleton function	Uses an M-shaped structuring element in the skeleton function.

## N

nautical mile	International unit of length used for sea and air navigation equal to 6,076.115 feet or 1,852 meters. <i>See also</i> mile.
neighbor	A pixel whose value affects the value of a nearby pixel when an image is processed. The neighbors of a pixel are usually defined by a kernel or a structuring element.
neighborhood operations	Operations on a point in an image that take into consideration the values of the pixels neighboring that point.
NI-IMAQ	Driver software for National Instruments IMAQ hardware.
nonlinear filter	Replaces each pixel value with a nonlinear function of its surrounding pixels.
nonlinear gradient filter	A highpass edge-extraction filter that favors vertical edges.

nonlinear Prewitt filter	A highpass, edge-extraction filter based on two-dimensional gradient information.
nonlinear Sobel filter	A highpass, edge-extraction filter based on two-dimensional gradient information. The filter has a smoothing effect that reduces noise enhancements caused by gradient operators.
Nth order filter	Filters an image using a nonlinear filter. This filter orders (or classifies) the pixel values surrounding the pixel being processed. The pixel being processed is set to the Nth pixel value, where N is the order of the filter.
number of planes (in an image)	The number of arrays of pixels that compose the image. A gray-level or pseudo-color image is composed of one plane, while an RGB image is composed of three planes (one for the red component, one for the blue, and one for the green).

**O**

offset	The coordinate position in an image where you want to place the origin of another image. Setting an offset is useful when performing mask operations.
opening	An erosion followed by a dilation. An opening removes small objects and smooths boundaries of objects in the image.
operators	Allow masking, combination, and comparison of images. You can use arithmetic and logic operators in IMAQ Vision.
optical character verification	A machine vision application that inspects the quality of printed characters.
optical representation	Contains the low-frequency information at the center and the high-frequency information at the corners of an FFT-transformed image.
outer gradient	Finds the outer boundary of objects.

**P**

palette	The gradation of colors used to display an image on screen, usually defined by a color lookup table.
pattern matching	The technique used to locate quickly a grayscale template within a grayscale image

picture aspect ratio	The ratio of the active pixel region to the active line region. For standard video signals like RS-170 or CCIR, the full-size picture aspect ratio normally is 4/3 (1.33).
picture element	An element of a digital image. Also called pixel.
pixel	Picture element. The smallest division that makes up the video scan line. For display on a computer monitor, a pixel's optimum dimension is square (aspect ratio of 1:1, or the width equal to the height).
pixel aspect ratio	The ratio between the physical horizontal size and the vertical size of the region covered by the pixel. An acquired pixel should optimally be square, thus the optimal value is 1.0, but typically it falls between 0.95 and 1.05, depending on camera quality.
pixel calibration	Directly calibrating the physical dimensions of a pixel in an image.
pixel depth	The number of bits used to represent the gray level of a pixel.
PNG	Portable Network Graphic. Image file format for storing 8-bit, 16-bit, and color images with lossless compression (extension PNG).
power 1/Y function	Similar to a logarithmic function but with a weaker effect.
power Y function	<i>See</i> exponential function.
Prewitt filter	Extracts the contours (edge detection) in gray-level values using a 3 x 3 filter kernel.
probability function	Defines the probability that a pixel in an image has a certain gray-level value.
proper-closing	A finite combination of successive closing and opening operations that you can use to fill small holes and smooth the boundaries of objects.
proper-opening	A finite combination of successive opening and closing operations that you can use to remove small particles and smooth the boundaries of objects.
pts	Points.
pyramidal matching	A technique used to increase the speed of a pattern matching algorithm by matching subsampled versions of the image and the reference pattern.

**Q**

**quantitative analysis**      Obtaining various measurements of objects in an image.

**R**

**real time**      A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.

**relative accuracy**      A measure in LSB of the accuracy of an ADC; it includes all nonlinearity and quantization errors but does not include offset and gain errors of the circuitry feeding the ADC.

**resolution**      The number of rows and columns of pixels. An image composed of  $m$  rows and  $n$  columns has a resolution of  $m \times n$ .

**reverse function**      Inverts the pixel values in an image, producing a photometric negative of the image.

**RGB**      Color encoding scheme using red, green, and blue (RGB) color information where each pixel in the color image is encoded using 32 bits: 8 bits for red, 8 bits for green, 8 bits for blue, and 8 bits for the alpha value (unused).

**Roberts filter**      Extracts the contours (edge detection) in gray level, favoring diagonal edges.

**ROI**      Region of interest. (1) An area of the image that is graphically selected from a window displaying the image. This area can be used focus further processing. (2) A hardware-programmable rectangular portion of the acquisition window.

**ROI tools**      Collection of tools from the LabVIEW Tools palette that enable you to select a region of interest from an image. These tools let you select a point or line; polygon, rectangle, and oval regions; and freehand lines and areas.

**rotational shift**      The amount by which one image is rotated with respect to a reference image. This rotation is computed with respect to the center of the image.

**rotation-invariant matching**      A pattern matching technique in which the reference pattern can be located at any orientation in the test image as well as rotated at any degree.

## S

s	Seconds.
saturation	The amount of white added to a pure color. Saturation relates to the richness of a color. A saturation of zero corresponds to a pure color with no white added. Pink is a red with low saturation.
scale-invariant matching	A pattern matching technique in which the reference pattern can be any size in the test image.
segmentation function	Fully partitions a labeled binary image into non-overlapping segments, with each segment containing a unique object.
separation function	Separates objects that touch each other by narrow isthmuses.
shape matching	Finds objects in an image whose shape matches the shape of the object specified by a shape template. The matching process is invariant to rotation and can be set to be invariant to the scale of the objects.
shift-invariant matching	A pattern matching technique in which the reference pattern can be located anywhere in the test image but cannot be rotated or scaled.
Sigma filter	A highpass filter that outlines edges.
skeleton function	Applies a succession of thinning operations to an object until its width becomes one pixel.
skiz function	Obtains lines in an image that separate each object from the others and are equidistant from the objects that they separate.
smoothing filter	Blurs an image by attenuating variations of light intensity in the neighborhood of a pixel.
Sobel filter	Extracts the contours (edge detection) in gray-level values using a 3 x 3 filter kernel.
spatial calibration	Assigning physical dimensions to the area of a pixel in an image.
spatial filters	Alter the intensity of a pixel with respect to variations in intensities of its neighboring pixels. You can use these filters for edge detection, image enhancement, noise reduction, smoothing, and so forth.
spatial resolution	The number of pixels in an image, in terms of the number of rows and columns in the image.

square function	<i>See</i> exponential function.
square root function	<i>See</i> logarithmic function.
standard representation	Contains the low-frequency information at the corners and high-frequency information at the center of an FFT-transformed image.
statute mile	<i>See</i> mile.
structuring element	A binary mask used in most morphological operations. A structuring element is used to determine which neighboring pixels contribute in the operation.
sub-pixel analysis	Finds the location of the edge coordinates in terms of fractions of a pixel.

**T**

template	Color, shape, or pattern that you are trying to match in an image using the color matching, shape matching, or pattern matching functions. A template can be a region selected from an image or it can be an entire image.
thickening	Alters the shape of objects by adding parts to the object that match the pattern specified in the structuring element.
thinning	Alters the shape of objects by eliminating parts of the object that match the pattern specified in the structuring element.
threshold	Separates objects from the background by assigning all pixels with intensities within a specified range to the object and the rest of the pixels to the background. In the resulting binary image, objects are represented with a pixel intensity of 255 and the background is set to 0.
threshold interval	Two parameters, the lower threshold gray-level value and the upper threshold gray-level value.
TIFF	Tagged Image File Format. Image format commonly used for encoding 8-bit, 16-bit, and color images (extension TIF).
Tools palette	Collection of tools that enable you to select regions of interest, zoom in and out, and change the image palette.
truth table	A table associated with a logic operator that describes the rules used for that operation.

## V

V

Volts.

value

The grayscale intensity of a color pixel computed as the average of the maximum and minimum red, green, and blue values of that pixel.

VI

Virtual Instrument. (1) A combination of hardware and/or software elements, typically used with a PC, that has the functionality of a classic stand-alone instrument (2) A LabVIEW software module (VI), which consists of a front panel user interface and a block diagram program.

## W

watershed

A technique used to segment an image into multiple regions.

white reference level

The level that defines what is white for a particular video system. *See also* black reference level.

## Z

zones

Areas in a displayed image that respond to user clicks. You can use these zones to control events which can then be interpreted within LabVIEW.

# Index

---

## Numbers

16-bit image display, mapping methods for, 2-3 to 2-4

## A

Absolute Difference operator (table), 6-2

Add operator (table), 6-2

advanced binary morphology functions. *See* binary morphology.

AIPD (National Instruments internal image file format), 1-6

alignment application

    color pattern matching, 14-21

    edge detection, 11-4

    pattern matching, 12-1

ambient lighting conditions

    color location tool, 14-15

    color pattern matching, 14-22

    pattern matching, 12-4

analysis of images. *See* image analysis.

AND operator (table), 6-3

area of digital particles, 10-1 to 10-2

    holes' area, 10-2

    number of holes, 10-2

    number of pixels, 10-1

    particle area, 10-1

    ratio, 10-1

    scanned area, 10-1

    total area, 10-2

arithmetic operators (table), 6-2

attenuation

    highpass FFT filters, 7-9

    lowpass FFT filters, 7-7

auto-median function

    binary morphology, 9-22

    grayscale morphology

concept and mathematics, 5-43

overview, 5-40

automatic thresholding, 8-3 to 8-6

    clustering

        examples, 8-4 to 8-5

        in-depth discussion, 8-7

        overview, 8-3

    entropy

        in-depth discussion, 8-7 to 8-8

        overview, 8-5

    interclass variance, 8-6, 8-8

    metric

        in-depth discussion, 8-8

        overview, 8-5

    moments

        in-depth discussion, 8-9

        overview, 8-5

    overview, 8-3

    techniques, 8-6 to 8-7

axes. *See* chords and axes.

axis of symmetry, gradient filter, 5-16 to 5-17

## B

barcode reader

    algorithm limits, 15-4

    purpose and use, 15-3

binary image, 9-1

binary morphology, 9-1 to 9-32

    advanced binary functions, 9-22 to 9-32

    basic concepts, 9-23

    border, 9-23

    circle, 9-30 to 9-31

    convex, 9-31 to 9-32

    Danielsson, 9-30

    distance, 9-29

    hole filling, 9-23

- labeling, 9-23 to 9-24
  - lowpass and highpass filters, 9-24 to 9-25
  - segmentation, 9-28 to 9-29
  - separation, 9-25 to 9-26
  - skeleton, 9-26 to 9-28
  - when to use, 9-22
  - connectivity, 9-7 to 9-10
    - basic concepts and examples, 9-7 to 9-8
  - Connectivity-4, 9-9
  - Connectivity-8, 9-9 to 9-10
    - in-depth discussion, 9-9 to 9-10
    - when to use, 9-7
  - overview, 9-1
  - pixel frame shape, 9-4 to 9-7
    - examples (figures), 9-4 to 9-6
    - hexagonal, 9-7
    - overview, 9-4
    - square frame, 9-6
  - primary morphology functions, 9-10 to 9-22
    - auto-median, 9-22
    - erosion and dilation, 9-11 to 9-13
    - hit-miss, 9-15 to 9-17
    - inner gradient, 9-14
    - opening and closing, 9-13 to 9-14
    - outer gradient, 9-14 to 9-15
    - proper-closing, 9-21
    - proper-opening, 9-21
    - thickening, 9-19 to 9-20
    - thinning, 9-17 to 9-19
    - when to use, 9-10
  - structuring elements, 9-1 to 9-7
    - basic concepts, 9-2
    - pixel frame shape, 9-4 to 9-7
    - size, 9-2 to 9-3
    - values, 9-3
    - when to use, 9-1 to 9-2
- B**
- Binary palette
    - gray-level values (table), 2-8
    - periodic palette (figure), 2-9
  - binary shape matching, 12-10 to 12-11
  - bit depth (image definition), 1-2
  - bitmap (BMP) file format, 1-5
  - blob analysis, III-1 to III-4
    - basic concepts, III-2 to III-4
    - parameters, III-3 to III-4
    - when to use, III-2
  - blobs, definition, III-1
  - blur and noise conditions
    - color location tool, 14-15
    - color pattern matching, 14-22
    - pattern matching, 12-4
  - BMP (bitmap) file format, 1-5
  - border function, binary morphology, 9-23
  - borders. *See* image borders.
  - breadth parameter, digital particles, 10-3
  - brightness, definition, 1-17

## C

- calibration. *See* spatial calibration.
- center of mass (X, Y), digital particle coordinate, 10-3
- chords and axes, digital particles, 10-4 to 10-5
  - max chord length, 10-4
  - max intercept, 10-5
  - mean chord X, 10-4
  - mean chord Y, 10-4
  - mean intercept perpendicular, 10-5
  - particle orientation, 10-5
- chromaticity, definition, 1-17
- CIE-Lab color space
  - overview, 1-17
  - transforming RGB to CIE L\*a\*b\*, 1-20 to 1-21
- circle detection functions, in dimensional measurements, 13-11 to 13-12

- circle function, binary morphology, 9-30 to 9-31
- closing function
  - binary morphology
    - basic concepts, 9-13
    - examples, 9-14
  - grayscale morphology
    - description, 5-39
    - examples, 5-39 to 5-40
- clustering technique, in automatic thresholding
  - examples, 8-4 to 8-5
  - in-depth discussion, 8-7
  - overview, 8-3
- CMY color space
  - description, 1-18
  - transforming RGB to CMY, 1-21
- color distribution
  - comparing, 14-9 to 14-10
  - learning, 14-9
- color identification
  - example, 14-7
  - purpose and use, 14-6 to 14-7, 14-12
  - sorting objects, 14-12 to 14-13
- color images
  - bytes per pixel (table), 1-4
  - encoding, 1-5
  - histogram of color image, 4-5
- color inspection, 14-7 to 14-8
- color location, 14-10 to 14-17
  - basic concepts, 14-15 to 14-17
  - identification of objects, 14-12 to 14-13
  - inspection, 14-11
  - overview, 14-10
  - sorting objects, 14-12 to 14-13
  - what to expect, 14-14
    - ambient lighting conditions, 14-15
    - blur and noise conditions, 14-15
    - pattern orientation and multiple instances, 14-14
  - when to use, 14-10 to 14-11
- color matching, 14-6 to 14-10
  - basic concepts, 14-9
  - comparing color distributions, 14-9 to 14-10
  - learning color distribution, 14-9
  - overview, 14-6
  - when to use, 14-6 to 14-8
    - color identification, 14-6 to 14-7
    - color inspection, 14-7 to 14-8
- color pattern matching, 14-18 to 14-25
  - basic concepts, 14-22 to 14-25
  - color matching and color location, 14-23
  - combining color location and grayscale pattern matching, 14-24 to 14-25
  - grayscale pattern matching, 14-23 to 14-24
  - what to expect
    - ambient lighting conditions, 14-22
    - blur and noise conditions, 14-22
    - pattern orientation and multiple instances, 14-21
  - when to use, 14-18 to 14-21
    - alignment, 14-21
    - gauging, 14-20
    - inspection, 14-20
- color spaces, 1-13 to 1-21
  - CIE-Lab color space, 1-17
  - CMY color space, 1-18
  - color sensations, 1-14
  - common types of color spaces, 1-13
  - definition, 1-13
  - generating color spectrum, 14-1 to 14-5
  - HSL color space, 1-17
  - RGB color space, 1-15 to 1-16
  - transformations
    - RGB and CIE L\*a\*b\*, 1-20 to 1-21
    - RGB and CMY, 1-21
    - RGB and HSL, 1-19 to 1-20

- RGB and YIQ, 1-21
- RGB to grayscale, 1-18
- when to use, 1-13 to 1-14
- YIQ color space, 1-19
- color spectrum, 14-1 to 14-5
  - generating, 14-3 to 14-5
  - HSL color space, 14-1 to 14-2
  - overview, 14-1
- color thresholding, 8-9 to 8-11
  - ranges
    - HSL image (figure), 8-11
    - RGB image (figure), 8-10
    - when to use, 8-9
- compactness factor parameter, 10-8
- comparison operators. *See* logic and comparison operators.
- complex images
  - bytes per pixel (table), 1-4
  - definition, 1-5
- Concentric Rake function, 11-13
- connectivity, 9-7 to 9-10
  - basic concepts and examples, 9-7 to 9-8
  - Connectivity-4, 9-9
  - Connectivity-8, 9-9 to 9-10
  - in-depth discussion, 9-9 to 9-10
  - when to use, 9-7
- contours
  - extracting and highlighting, 5-21 to 5-22
  - thickness, 5-23
- contrast
  - histogram for determining lack of contrast, 4-2
  - setting, 3-5
- conventions used in manual, xv
- convex function, binary morphology, 9-31 to 9-32
- convolution
  - definition, 5-13
  - types of (families), 5-13
- convolution kernels. *See also* linear filters.
  - basic concepts, 5-10 to 5-12
  - examples of kernels (figure), 5-11
  - filtering border pixels (figure), 5-12
  - mechanics of filtering (figure), 5-11
  - size of, 5-13
  - when to use, 5-10
- coordinate system
  - dimensional measurements, 13-3 to 13-5
  - edge-based functions, 13-6 to 13-8
  - pattern matching-based functions, 13-8 to 13-10
  - steps for defining, 13-4 to 13-5
  - when to use, 13-4
- spatial calibration, 3-9 to 3-11
  - axis direction (figure), 3-10
  - origin and angle (figure), 3-10
  - redefining, 3-17 to 3-18
- coordinates of digital particles, 10-3 to 10-4
  - center of mass X and center of mass Y, 10-3
  - max chord X and max chord Y, 10-4
  - max X, max Y, 10-4
  - min X, min Y, 10-4
- Corrected Projection X parameter, particle measurement, 10-9
- Corrected Projection Y parameter, particle measurement, 10-9
- correction region, in calibration, 3-15 to 3-16
- cross correlation, in pattern matching
  - correlation procedure (figure), 12-9
  - in-depth discussion, 12-8 to 12-9
  - overview, 12-5
- cumulative histogram, 4-3 to 4-5
- customer education, B-1

**D**

- Danielsson function, 9-30
- densitometry parameters, 4-7
- depth of field
  - definition, 3-3
  - setting, 3-5

- derived measurements for digital particles (table), 10-10 to 10-11
- detection application. *See* edge detection.
- differentiation filter
  - definition, 5-29
  - mathematical concepts, 5-34
- digital image processing, definition, 1-1
- digital images, 1-1 to 1-21
  - color spaces, 1-13 to 1-21
    - CIE-Lab color space, 1-17
    - CMY color space, 1-18
    - color sensations, 1-14
    - common types of color spaces, 1-13
    - definition, 1-13
    - HSL color space, 1-17
    - RGB color space, 1-15 to 1-16
    - transformations
      - RGB and CIE L\*a\*b\*, 1-20 to 1-21
      - RGB and CMY, 1-21
      - RGB and HSL, 1-19 to 1-20
      - RGB and YIQ, 1-21
      - RGB to grayscale, 1-18
    - when to use, 1-13 to 1-14
    - YIQ color space, 1-19
  - definitions, 1-1
  - image borders, 1-8 to 1-10
  - image file formats, 1-5 to 1-6
  - image masks, 1-10 to 1-12
  - image types, 1-3 to 1-5
    - bytes per pixel (table), 1-3 to 1-4
    - color images, 1-5
    - complex images, 1-5
    - grayscale images, 1-4
  - internal representation of IMAQ Vision
    - image, 1-6
  - properties
    - image definition, 1-2
    - image resolution, 1-2
    - number of planes, 1-3
    - overview, 1-1
  - digital particles, 10-1 to 10-11
    - areas, 10-1 to 10-2
    - chords and axes, 10-4 to 10-5
    - coordinates, 10-3 to 10-4
    - derived measurements (table), 10-10 to 10-11
    - diverse measurements, 10-9
    - lengths, 10-2 to 10-3
    - primary measurement definitions, 10-9 to 10-10
    - shape equivalence, 10-6 to 10-7
    - shape features, 10-8 to 10-9
    - when to use, 10-1
  - dilation function
    - binary morphology
      - basic concepts, 9-11
      - examples, 9-12
      - structure element effects (table), 9-13
    - grayscale morphology
      - concept and mathematics, 5-41
      - examples, 5-37 to 5-38
      - purpose and use, 5-37
  - dimensional measurements, 13-1 to 13-19
    - coordinate system, 13-3 to 13-5
    - edge-based functions, 13-6 to 13-8
    - pattern matching-based
      - functions, 13-8 to 13-10
    - steps for defining, 13-4 to 13-5
    - when to use, 13-4
  - finding features or measurement
    - points, 13-10 to 13-13
    - edge-based features, 13-10
    - line and circular features, 13-11 to 13-12
    - overview, 13-2
    - shape-based features, 13-13
  - finding part of image in region of interest, 13-2
  - making measurements
    - analytic geometry, 13-14

- distance measurements,
  - 13-13 to 13-14
  - line fitting, 13-15 to 13-19
  - overview, 13-3
- overview, 13-1
  - qualifying measurements, 13-3
  - when to use, 13-1
- direction, gradient filter, 5-16 to 5-17
- display, 2-1 to 2-11
  - image display, 2-1 to 2-4
    - basic concepts, 2-1
    - display modes, 2-2
    - mapping methods for 16-bit image display, 2-3 to 2-4
    - when to use, 2-1
  - nondestructive overlay, 2-11
  - palettes, 2-4 to 2-9
    - basic concepts, 2-5
    - Binary palette, 2-8 to 2-9
    - Gradient palette, 2-7
    - Gray palette, 2-5
    - Rainbow palette, 2-7
    - Temperature palette, 2-6
    - when to use, 2-4
  - regions of interest, 2-9 to 2-10
    - defining, 2-10
    - types of contours (table), 2-10
    - when to use, 2-9
- distance function, binary morphology, 9-29
- distance measurements, 13-13 to 13-14
- distortion
  - description, 3-7
  - perspective and distortion errors (figure), 3-6
- diverse measurements, digital particles, 10-9
- Divide operator (table), 6-2
- dltd gradient filters
  - predefined kernels
    - Prewitt filters, A-1
    - Sobel filters, A-2 to A-4
- documentation
  - conventions used in manual, xv
  - related documentation, xvi
- E**
- edge detection, 11-1 to 11-13
  - characteristics of edge, 11-5 to 11-7
    - common model (figure), 11-5
    - edge length parameter, 11-6
    - edge polarity parameter, 11-6
    - edge position parameter, 11-6 to 11-7
    - edge strength parameter, 11-6
  - definition of edge, 11-4 to 11-5
  - methods for edge detection, 11-7 to 11-10
    - advanced, 11-8 to 11-9
    - simple, 11-7 to 11-8
    - sub-pixel accuracy, 11-9 to 11-10
  - overview, 11-1
  - in pattern matching, 12-7
  - two-dimensional search regions, 11-11 to 11-13
    - Concentric rake, 11-13
    - Rake, 11-11
    - Spoke, 11-12
  - when to use, 11-1 to 11-4
    - alignment, 11-4
    - detection, 11-3
    - dimensional measurements, 13-10
    - gauging, 11-2
- edge outlining with gradient filters
  - edge extraction and highlighting, 5-17 to 5-18
  - edge thickness, 5-19
- edge-based coordinate system functions, 13-6 to 13-8
  - single search area, 13-6 to 13-7
  - two search areas, 13-7 to 13-8
- ellipse major axis parameter, 10-6
- ellipse minor axis parameter, 10-6

ellipse ratio parameter, 10-7  
 elongation factor parameter, 10-8  
 entropy technique, in automatic thresholding  
     in-depth discussion, 8-7 to 8-8  
     overview, 8-5  
**Equalize function**  
     basic concepts, 5-8  
     examples, 5-9 to 5-10  
     summary (table), 5-3  
**equivalent ellipse minor axis parameter**, 10-6  
**erosion function**  
     binary morphology  
         basic concepts, 9-11  
         examples, 9-12  
         structure element effects (table), 9-12  
     grayscale morphology  
         concept and mathematics, 5-41  
         examples, 5-37 to 5-38  
         purpose and use, 5-37  
**error map output of calibration**  
     function, 3-12 to 3-13  
**exponential and gamma correction**  
     basic concepts, 5-6  
     examples, 5-7 to 5-8  
     summary (table), 5-3  
**external edge function, binary morphology**, 9-14 to 9-15

**F**

**Fast Fourier Transform (FFT).** *See also frequency filters.*  
     definition, 7-1  
     FFT display, 7-13  
     FFT representation, 7-3 to 7-6  
         optical representation, 7-5 to 7-6  
         standard representation, 7-3 to 7-4  
     Fourier Transform concepts, 7-12  
**feature, in pattern matching**, 12-1

**fiducials**  
     definition, 12-1  
     example of common fiducial (figure), 12-2  
**field of view**  
     definition, 3-3  
     relationship with pixel resolution, 3-4  
**filters.** *See convolution kernels; frequency filters; spatial filters.*  
**Fourier Transform**, 7-12. *See also Fast Fourier Transform (FFT).*  
**frame.** *See pixel frame shape.*  
**frequency filters**, 7-1 to 7-13  
     definition, 7-1  
     Fast Fourier Transform concepts, 7-3 to 7-6  
         FFT display, 7-13  
         FFT representation, 7-3 to 7-6  
         Fourier Transform, 7-12  
         overview, 7-3  
     FFT representation  
         optical representation, 7-5 to 7-6  
         standard representation, 7-3 to 7-4  
     highpass FFT filters, 7-9 to 7-11  
         attenuation, 7-9  
         examples, 7-10 to 7-11  
         overview, 7-2  
         truncation, 7-10  
     lowpass FFT filters, 7-6 to 7-8  
         attenuation, 7-7  
         examples, 7-8  
         overview, 7-2  
         truncation, 7-7  
     mask FFT filters  
         overview, 7-2  
         purpose and use, 7-11  
         overview, 7-1 to 7-2  
         when to use, 7-3  
     frequency processing, 7-1

**G**

gauging application. *See also* dimensional measurements.

- color pattern matching, 14-20
- edge detection, 11-2
- pattern matching, 12-1

Gaussian filters, 5-26 to 5-27

- example, 5-26
- kernel definition, 5-26 to 5-27
- predefined kernels, A-7

geometric measurements, 13-14 to 13-19

gradient filters

- linear, 5-15 to 5-19
  - definition, 5-15
  - edge extraction and edge highlighting, 5-17 to 5-18
  - edge thickness, 5-19
  - example, 5-15 to 5-16
  - filter axis and direction, 5-16 to 5-17
  - kernel definition, 5-16
- nonlinear
  - definition, 5-29
  - mathematical concepts, 5-34

Gradient palette, 2-7

Gray palette, 2-5

gray-level values

- in Binary palette (table), 2-8
- of pixels, 1-1

grayscale images

- bytes per pixel (table), 1-3 to 1-4
- pixel encoding, 1-4
- transforming RGB to grayscale, 1-18 to 1-19

grayscale morphology functions, 5-36 to 5-43

- auto-median
  - concepts and mathematics, 5-42
  - overview, 5-40
- basic concepts, 5-36 to 5-37

**closing**

- opening and closing examples, 5-39 to 5-40
- overview, 5-39

concepts and mathematics, 5-41 to 5-43

dilation

- concepts and mathematics, 5-41
- erosion and dilation
  - examples, 5-37 to 5-38
  - overview, 5-37

erosion

- concepts and mathematics, 5-41
- erosion and dilation examples, 5-37 to 5-38
- overview, 5-37

opening

- opening and closing
  - examples, 5-39 to 5-40
  - overview, 5-38

proper-closing

- concepts and mathematics, 5-42
- overview, 5-40

proper-opening

- concepts and mathematics, 5-42
- overview, 5-40
- when to use, 5-36

grayscale pattern matching

- combining color location and grayscale pattern matching, 14-24 to 14-25
- methods, 14-23 to 14-24

**H**

height parameter, digital particles, 10-3

hexagonal pixel frame, 9-7

Heywood circularity factor, 10-8

highpass filters

- binary morphology
  - basic concepts, 9-24
  - effects (table), 9-24
  - example, 9-25

- classes (table), 5-14
- definition, 5-14
- highpass frequency (FFT) filters, 7-9 to 7-11
  - attenuation, 7-9
  - examples, 7-10 to 7-11
  - overview, 7-2
  - truncation, 7-10
- histogram, 4-1 to 4-5
  - basic concepts, 4-2 to 4-3
  - color image histogram, 4-5
  - cumulative histogram, 4-3 to 4-5
  - definition, 4-1
  - interpretation, 4-4
  - linear histogram, 4-3
  - scale of histogram, 4-4 to 4-5
  - when to use, 4-1 to 4-2
- hit-miss function, binary
  - morphology, 9-15 to 9-17
    - basic concepts, 9-15
    - examples, 9-16
    - strategies for using (table), 9-17
- hole filling function, binary morphology, 9-23
- hole parameters
  - area of digital particle, 10-2
  - length of digital particle, 10-2
- HSL color space
  - basic concepts, 1-17
  - generating color spectrum, 14-1 to 14-5
  - mapping RGB to HSL color
    - space, 1-19 to 1-20
- hue, definition, 1-17
- hydraulic radius parameter, 10-8 to 10-9
  
- |
- image analysis, 4-1 to 4-7
  - histogram, 4-1 to 4-5
    - basic concepts, 4-2 to 4-3
    - color images, 4-5
    - cumulative histogram, 4-3 to 4-4
    - interpretation, 4-4
- linear histogram, 4-3
- scale, 4-4 to 4-5
  - when to use, 4-1 to 4-2
- intensity measurements, 4-6 to 4-7
- line profile, 4-6
- image borders, 1-8 to 1-10
  - definition, 1-8
  - size of border, 1-8
  - specifying pixel values, 1-8 to 1-10
- image correction, in calibration, 3-14
- image definition (bit depth), 1-2
- image display, 2-1 to 2-4
  - basic concepts, 2-1
  - display modes, 2-2
  - mapping methods for 16-bit image display, 2-3 to 2-4
  - when to use, 2-1
- image files and formats, 1-5 to 1-6
- image masks, 1-10 to 1-12
  - applying with different offsets (figure), 1-12
  - definition, 1-10
  - effect of mask (figure), 1-11
  - offset for limiting image mask (figure), 1-11
  - using image masks, 1-11 to 1-12
  - when to use, 1-10
- image processing, 5-1 to 5-43
  - convolution kernels, 5-10 to 5-12
  - definition, 2-1
  - grayscale morphology
    - functions, 5-36 to 5-43
      - auto-median, 5-40
      - basic concepts, 5-36 to 5-37
      - closing, 5-39
      - concepts and mathematics, 5-41 to 5-43
      - dilation, 5-37
      - erosion, 5-37
      - opening, 5-38
      - proper-closing, 5-40

- proper-opening, 5-40
    - when to use, 5-36
  - lookup tables, 5-1 to 5-10
    - basic concepts, 5-1 to 5-2
    - Equalize, 5-8 to 5-10
    - exponential and gamma correction, 5-6 to 5-8
    - logarithmic and inverse gamma correction, 5-4 to 5-6
    - predefined, 5-3
    - when to use, 5-1
  - spatial filters, 5-13 to 5-43
    - classification summary (table), 5-14
    - in-depth discussion, 5-32 to 5-36
    - linear filters, 5-15 to 5-27
    - nonlinear filters, 5-27 to 5-31
    - when to use, 5-13 to 5-14
  - image types
    - bytes per pixel (table), 1-3 to 1-4
    - color images, 1-5
    - complex images, 1-5
    - grayscale images, 1-4
  - image understanding, in pattern matching, 12-6
  - image visualization, definition, 2-1
  - images. *See also* digital images.
    - definition, 1-1
    - internal representation of IMAQ Vision image, 1-6 to 1-7
  - inner gradient function, binary morphology, 9-14
  - inspection
    - color inspection, 14-7 to 14-8
    - color location, 14-11
    - color pattern matching, 14-20
    - pattern matching, 12-1
  - instrument readers, 15-1 to 15-4
    - barcode, 15-3 to 15-4
  - LCD functions, 15-2 to 15-3
  - meter functions, 15-1 to 15-2
    - when to use, 15-1
  - intensity measurements
    - densitometry parameters, 4-7
    - when to use, 4-6 to 4-7
  - intensity threshold, 8-2
  - interclass variance technique, in automatic thresholding, 8-6, 8-8
  - internal edge function, binary morphology, 9-14
  - internal representation of IMAQ Vision image, 1-6 to 1-7
  - interpretation of histogram, 4-4
  - inverse gamma correction. *See* logarithmic and inverse gamma correction.
- ## J
- JPEG (Joint Photographic Experts Group) format, 1-6
- ## K
- kernel definition
    - Gaussian filters, 5-26 to 5-27
    - gradient filters, 5-16
    - Laplacian filters, 5-21
    - smoothing filters, 5-24 to 5-25
  - kernels, predefined. *See* predefined kernels.
- ## L
- L-skeleton function, 9-26 to 9-27
  - labeling function, binary morphology, 9-23 to 9-24
  - Laplacian filters, 5-20 to 5-23
    - contour extraction and highlighting, 5-21 to 5-22
    - contour thickness, 5-23
    - example, 5-20
    - kernel definition, 5-21
    - predefined kernels, A-5

LCD functions  
 algorithm limits, 15-3  
 purpose and use, 15-2

length parameters, digital particles, 10-2 to 10-3  
 breadth, 10-3  
 height, 10-3  
 holes' perimeter, 10-2  
 particle perimeter, 10-2

lens focal length, setting, 3-5

lighting conditions, in pattern matching, 12-4

line detection functions, in dimensional measurements, 13-11

line fitting function, in dimensional measurements, 13-15 to 13-19  
 calculation of mean square distance (figure), 13-17  
 data set and fitted line (figure), 13-16  
 strongest line fit (figure), 13-18

line profile, 4-6

linear filters, 5-15 to 5-27  
 classes (table), 5-14  
 Gaussian filters, 5-26 to 5-27  
 example, 5-26  
 kernel definition, 5-26 to 5-27  
 predefined kernels, A-7

gradient filters, 5-15 to 5-19  
 edge extraction and edge highlighting, 5-17 to 5-18  
 edge thickness, 5-19  
 example, 5-15 to 5-16  
 filter axis and direction, 5-16 to 5-17  
 kernel definition, 5-16  
 predefined kernels, A-1 to A-4

in-depth discussion, 5-32 to 5-33

Laplacian filters, 5-20 to 5-23  
 contour extraction and highlighting, 5-21 to 5-22  
 contour thickness, 5-23  
 example, 5-20  
 kernel definition, 5-21

predefined kernels, A-5  
 overview, 5-15  
 smoothing filters, 5-24 to 5-25  
 example, 5-24  
 kernel definition, 5-24 to 5-25  
 predefined kernels, A-6

linear histogram, 4-3

logarithmic and inverse gamma correction  
 basic concepts, 5-4  
 examples, 5-4 to 5-6  
 summary (table), 5-3

logic and comparison operators  
 examples, 6-5 to 6-7  
 list of operators (table), 6-3  
 purpose and use, 6-2  
 truth tables, 6-4 to 6-5  
 using with binary image masks (table), 6-4

Logic Difference operator (table), 6-3

lookup table transformations  
 basic concepts, 5-1 to 5-2  
 examples, 5-2 to 5-3  
 when to use, 5-1

lookup tables  
 Equalize, 5-8 to 5-10  
 exponential and gamma correction, 5-6 to 5-8  
 logarithmic and inverse gamma correction, 5-4 to 5-6  
 predefined lookup tables, 5-3

lowpass filters  
 binary morphology  
 basic concepts, 9-24  
 effects (table), 9-24  
 example, 9-25  
 classes (table), 5-14  
 definition, 5-14  
 nonlinear  
 basic concepts, 5-30  
 mathematical concepts, 5-35

lowpass frequency (FFT) filters, 7-6 to 7-8  
 attenuation, 7-7  
 examples, 7-8  
 overview, 7-2  
 truncation, 7-7  
 LUTs. *See* lookup tables.

## M

M-skeleton function, 9-27  
 manual. *See* documentation.  
 mapping methods for 16-bit image  
 display, 2-3 to 2-4  
 mask FFT filters  
 overview, 7-2  
 purpose and use, 7-11  
 Mask operator (table), 6-3  
 masks. *See* image masks; structuring elements.  
 max chord length parameter, digital  
 particles, 10-4  
 max chord X and max chord Y parameter,  
 digital particles, 10-4  
 max intercept parameter, digital particles, 10-5  
 Max operator (table), 6-3  
 max X, max Y parameter, digital  
 particles, 10-4  
 mean chord X and mean chord Y parameters,  
 digital particles, 10-4  
 mean intercept perpendicular parameter,  
 digital particles, 10-5  
 Mean operator (table), 6-3  
 median filter  
 basic concepts, 5-30  
 mathematical concepts, 5-35  
 meter functions  
 algorithm limits, 15-2  
 purpose and use, 15-1  
 metric technique, in automatic thresholding  
 in-depth discussion, 8-8  
 overview, 8-5  
 Min operator (table), 6-3

min X, min Y parameter, digital particles, 10-4  
 Modulo operator (table), 6-2  
 moments of inertia  $I_{xx}$ ,  $I_{yy}$ ,  $I_{xy}$  parameter, 10-8  
 moments technique, in automatic thresholding  
 in-depth discussion, 8-9  
 overview, 8-5  
 morphology functions. *See* binary  
 morphology; grayscale morphology  
 functions.  
 multiple instances of patterns. *See* pattern  
 orientation and multiple instances.  
 Multiply operator (table), 6-2

## N

NAND operator (table), 6-3  
 National Instruments internal image file  
 format (AIPD), 1-6  
 neighbors (pixels), definition, 1-8  
 NI Developer Zone, B-1  
 noise. *See* blur and noise conditions.  
 nondestructive overlay, 2-11  
 basic concepts, 2-11  
 when to use, 2-11  
 nonlinear algorithm for calibration, 3-11  
 nonlinear filters, 5-27 to 5-31  
 classes (table), 5-14  
 differentiation filter  
 mathematical concepts, 5-34  
 overview, 5-29  
 gradient filter  
 mathematical concepts, 5-34  
 overview, 5-29  
 in-depth discussion, 5-33 to 5-36  
 lowpass filter  
 mathematical concepts, 5-35  
 overview, 5-30  
 median filter  
 mathematical concepts, 5-35  
 overview, 5-30

Nth order filter  
     effects (table), 5-31  
     mathematical concepts, 5-35 to 5-36  
     overview, 5-31

Prewitt filter  
     description, 5-27 to 5-29  
     example, 5-28  
     mathematical concepts, 5-33  
     predefined kernels, A-1

Roberts filter  
     mathematical concepts, 5-34  
     overview, 5-29

Sigma filter  
     mathematical concepts, 5-35  
     overview, 5-30

Sobel filter  
     description, 5-28  
     example, 5-28 to 5-29  
     mathematical concepts, 5-33  
     predefined kernels, A-2 to A-4

NOR operator (table), 6-3

Nth order filter  
     basic concepts, 5-31  
     examples, 5-31  
     mathematical concepts, 5-35 to 5-36

number of planes, 1-3

**0**

opening function  
     binary morphology  
         basic concepts, 9-13  
         examples, 9-14  
     grayscale morphology  
         description, 5-38  
         examples, 5-39 to 5-40

operators  
     arithmetic, 6-2  
     basic concepts, 6-1 to 6-2  
     logic and comparison, 6-2 to 6-7  
     when to use, 6-1

optical representation, FFT display, 7-5 to 7-6  
 OR operator (table), 6-3  
 outer gradient function, binary  
     morphology, 9-14 to 9-15

**P**

palettes, 2-4 to 2-9  
     basic concepts, 2-5  
     Binary palette, 2-8 to 2-9  
     definition, 2-4  
     Gradient palette, 2-7  
     Gray palette, 2-5 to 2-6  
     Rainbow palette, 2-7  
     Temperature palette, 2-6  
     when to use, 2-4

particle area parameter, 10-1

particle measurement. *See* digital particles.

particle orientation parameter, 10-5

particle perimeter parameter, 10-2

pattern matching, 12-1 to 12-11. *See also*  
     color pattern matching; edge detection.  
     binary shape matching, 12-10 to 12-11  
     coordinate system for dimensional  
         measurements, 13-8 to 13-10  
     cross correlation  
         in-depth discussion, 12-8 to 12-9  
         overview, 12-5  
     grayscale pattern matching  
         combining color location and  
         grayscale pattern  
         matching, 14-24 to 14-25  
         methods, 14-23 to 14-24  
     new techniques, 12-6 to 12-7  
     overview, 12-1  
     pyramidal matching, 12-5  
     scale-invariant matching, 12-5 to 12-6  
     shape matching, 12-10 to 12-11  
     traditional techniques, 12-5 to 12-6  
     what to expect, 12-3 to 12-4  
         ambient lighting conditions, 12-4

- blur and noise conditions, 12-4
- pattern orientation and multiple instances, 12-3 to 12-4
  - when to use, 12-1 to 12-2
- pattern orientation and multiple instances
  - color location tool, 14-14
  - color pattern matching, 14-21
  - pattern matching, 12-3 to 12-4
- periodic palette (figure), 2-9
- perspective, 3-5 to 3-6
  - camera angle relative to object (figure), 3-6
  - perspective and distortion errors (figure), 3-6
- perspective algorithm for calibration, 3-11
- picture element, 1-1
- pixel frame shape, 9-4 to 9-7
  - examples (figures), 9-4 to 9-6
  - hexagonal, 9-7
  - overview, 9-4
  - square frame, 9-6
- pixel resolution
  - determining, 3-3
  - relationship with field of view, 3-4
- pixels
  - gray-level value, 1-1
  - neighbors, 1-8
  - number of pixels
    - in area of particle, 10-1
    - in sensor, 3-5
  - spatial coordinates, 1-1
  - values for image border, 1-8 to 1-9
- planes, number in image, 1-3
- PNG (portable network graphics) file format, 1-5
- predefined kernels
  - Gaussian kernels, A-7
  - gradient kernels, A-1 to A-7
    - Prewitt filters, A-1
    - Sobel filters, A-2 to A-4
  - Laplacian kernels, A-5
- smoothing kernels, A-6
- predefined lookup tables, 5-3
- Prewitt filter
  - basic concepts, 5-27 to 5-28
  - example, 5-28 to 5-29
  - mathematical concepts, 5-33
  - predefined kernels, A-1
- primary binary morphology functions. *See* binary morphology.
- proper-closing function
  - binary morphology, 9-21
  - grayscale morphology
    - concept and mathematics, 5-42
    - overview, 5-40
- proper-opening function
  - binary morphology, 9-21
  - grayscale morphology
    - concept and mathematics, 5-42
    - overview, 5-40
- pyramidal matching, 12-5

## Q

- quality information for spatial calibration, 3-12 to 3-13
- quality score output of calibration function, 3-12

## R

- Rainbow palette, 2-7
- Rake function, 11-11
- ratio area parameter, 10-1
- rectangle big side parameter, 10-7
- rectangle ratio parameter, 10-7
- rectangle small side parameter, 10-7
- regions of interest, 2-9 to 2-10
  - calibration correction region, 3-15 to 3-16
  - calibration ROI, 3-12
  - defining, 2-10
  - functions, 2-1

- types of contours (table), 2-10
  - when to use, 2-9
  - resolution, 3-3 to 3-5
    - definition, 3-3
    - determining pixel resolution, 3-3
    - field of view, 3-4
    - sensor size and number of pixels in sensor, 3-5
  - RGB color space
    - basic concepts, 1-15 to 1-16
    - RGB cube (figure), 1-16
    - transforming color spaces
      - RGB and CIE L\*a\*b\*, 1-20 to 1-21
      - RGB and CMY, 1-21
      - RGB and HSL, 1-19
      - RGB and YIQ, 1-21
      - RGB to grayscale, 1-18
  - Roberts filter
    - definition, 5-29
    - mathematical concepts, 5-34
  - ROI. *See* regions of interest.
- 
- ## S
- saturation
    - definition, 1-17
    - detecting with histogram, 4-1 to 4-2
  - scale of histograms, 4-4 to 4-5
  - scale-invariant matching, 12-5
  - scaling mode, in calibration, 3-14 to 3-15
  - scanned area parameter, 10-1
  - segmentation function
    - basic concepts, 9-28 to 9-29
    - compared with skiz function, 9-29
  - sensation of colors, 1-14
  - sensor size
    - definition, 3-3
    - number of pixels in sensor, 3-5
  - separation function, binary morphology, 9-25 to 9-26
  - shape equivalence, digital particles, 10-6 to 10-7
    - ellipse major axis, 10-6
    - ellipse minor axis, 10-6
    - ellipse ratio, 10-7
    - equivalent ellipse minor axis, 10-6
    - rectangle big side, 10-7
    - rectangle ratio, 10-7
    - rectangle small side, 10-7
  - shape features, 10-8 to 10-9
    - compactness factor, 10-8
    - elongation factor, 10-8
    - Heywood circularity factor, 10-8
    - hydraulic radius, 10-8 to 10-9
    - moments of inertia  $I_{xx}$ ,  $I_{yy}$ ,  $I_{xy}$ , 10-8
    - Waddel disk diameter, 10-9
  - shape matching
    - basic concepts, 12-10 to 12-11
    - dimensional measurement, 13-13
    - example, 12-11
    - when to use, 12-10
  - Sigma filter
    - basic concepts, 5-30
    - mathematical concepts, 5-35
  - skeleton functions, 9-26 to 9-28
    - comparison between segmentation and skiz functions, 9-29
    - L-skeleton, 9-26 to 9-27
    - M-skeleton, 9-27
    - skiz, 9-27 to 9-28
  - skiz function
    - basic concepts, 9-27 to 9-28
    - compared with segmentation function, 9-29
  - smoothing filters, 5-24 to 5-25
    - example, 5-24
    - kernel definition, 5-24 to 5-25
    - predefined kernels, A-6
  - Sobel filter
    - basic concepts, 5-28 to 5-29
    - example, 5-28 to 5-29

- mathematical concepts, 5-33
- predefined kernels, A-2 to A-4
- spatial calibration, 3-7 to 3-18
  - algorithms, 3-11 to 3-12
  - coordinate system, 3-9 to 3-11
  - correction region, 3-15 to 3-16
  - definition, 3-7
  - image correction, 3-14
  - overview, 3-7
  - process of calibration, 3-8 to 3-9
  - quality information, 3-12 to 3-13
  - redefining coordinate systems, 3-17 to 3-18
  - scaling mode, 3-14 to 3-15
  - simple calibration, 3-16 to 3-17
  - when to use, 3-7 to 3-8
- spatial filters, 5-13 to 5-43
  - categories, 5-14
  - classification summary (table), 5-14
  - definition, 5-14
  - linear filters, 5-15 to 5-27
    - Gaussian filters, 5-26 to 5-27
    - gradient filter, 5-15 to 5-19
    - in-depth discussion, 5-32 to 5-33
    - Laplacian filters, 5-20 to 5-23
    - smoothing filters, 5-24 to 5-25
  - nonlinear filters, 5-27 to 5-31
    - differentiation filter, 5-29
    - gradient filter, 5-29
    - in-depth discussion, 5-33 to 5-36
    - lowpass filter, 5-30
    - median filter, 5-30
    - Nth order filter, 5-31
  - predefined kernels, A-1 to A-4
  - Prewitt filter, 5-27 to 5-29
  - Roberts filter, 5-29
  - Sigma filter, 5-30
  - Sobel, 5-28 to 5-29
- spatial frequencies, 7-1
- spatial resolution of images, 1-2
- Spoke function, 11-12
- square pixel frame, 9-6
- standard representation, FFT
  - display, 7-3 to 7-4
- structuring elements, 9-1 to 9-7
  - basic concepts, 9-2
  - dilation function effects (table), 9-13
  - erosion function effects (table), 9-12
  - pixel frame shape, 9-4 to 9-7
  - size, 9-2 to 9-3
  - values, 9-3
  - when to use, 9-1 to 9-2
- Subtract operator (table), 6-2
- Sum X parameter, particle measurement, 10-9
- Sum Y parameter, particle measurement, 10-9
- SumXX, SumYY, SumXY parameter, particle measurement, 10-9
- system integration, by National Instruments, B-1
- system setup, 3-1 to 3-7. *See also* spatial calibration.
  - acquiring quality images, 3-3 to 3-7
  - basic concepts, 3-2 to 3-3
  - contrast, 3-5
  - depth of field, 3-5
  - distortion, 3-5 to 3-6
  - fundamental parameters (figure), 3-2
  - perspective, 3-5 to 3-6
  - resolution, 3-3 to 3-5

## T

- tagged image file format (TIFF), 1-5
- technical support resources, B-1 to B-2
- Temperature palette, 2-6
- thickening function, binary morphology
  - basic concepts, 9-19 to 9-20
  - examples, 9-20
- thinning function, binary morphology, 9-17 to 9-19
  - basic concepts, 9-17 to 9-18
  - examples, 9-18 to 9-19

thresholding, 8-1 to 8-11  
    automatic, 8-3 to 8-6  
        clustering, 8-3 to 8-5, 8-7  
        entropy, 8-5, 8-7 to 8-8  
        in-depth discussion, 8-6 to 8-9  
        interclass variance, 8-6, 8-8  
        metric, 8-5, 8-8  
        moments, 8-5, 8-9  
        techniques, 8-6 to 8-7  
    color, 8-9 to 8-11  
    example, 8-2 to 8-3  
    intensity threshold, 8-2  
    overview, 8-1  
    when to use, 8-1  
TIFF (tagged image file format), 1-5  
total area parameter, 10-2  
transforming color spaces. *See* color spaces.  
tri-chromatic theory of color, 1-14  
truncation  
    highpass FFT filters, 7-10  
    lowpass FFT filters, 7-7

truth tables, 6-4 to 6-5  
two-dimensional edge-detection. *See* edge detection.

## W

Waddel disk diameter, 10-9  
Web support from National Instruments, B-1  
working distance, definition, 3-3  
Worldwide technical support, B-2

## X

XOR operator (table), 6-3

## Y

YIQ color space  
    description, 1-18  
    transforming RGB to YIQ, 1-21