

# IMAQ™

---

## IMAQ Vision User Manual

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

[www.natinst.com](http://www.natinst.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 (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, 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, Norway 32 27 73 00,  
Singapore 2265886, Spain (Madrid) 91 640 0085, Spain (Barcelona) 93 582 0251, Sweden 08 587 895 00,  
Switzerland 056 200 51 51, Taiwan 02 2377 1200, United Kingdom 01635 523545

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

© Copyright 1999 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 THEREOF 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™ is a trademark of National Instruments Corporation.

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

## WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

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. Applications of National Instruments products involving medical or clinical treatment can create a potential for death or bodily injury caused by product failure, or by errors on the part of the user or application designer. 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
-------------------	----

## Chapter 1 Images

Introduction to Digital Images .....	1-1
Properties of a Digitized Image .....	1-2
Image Resolution .....	1-2
Image Definition .....	1-2
Number of Planes .....	1-2
Image Types and Formats .....	1-3
Gray-Level Images .....	1-3
Color Images .....	1-3
Complex Images .....	1-3
Image Files .....	1-5
Processing Color Images .....	1-5
Image Pixel Frame .....	1-6
Rectangular Frame .....	1-7
Hexagonal Frame .....	1-8

## Chapter 2 Tools and Utilities

Palettes .....	2-1
Gray Palette .....	2-2
Temperature Palette .....	2-3
Rainbow Palette .....	2-3
Gradient Palette .....	2-4
Binary Palette .....	2-4
Image Histogram .....	2-5
Definition .....	2-5
Linear Histogram .....	2-6
Cumulative Histogram .....	2-6
Interpretation .....	2-7
Histogram of Color Images .....	2-7
Histogram Scale .....	2-7
Line Profile .....	2-8
3D View .....	2-9

## Chapter 3

### Look-up Table Transformations

About Look-up Table Transformations.....	3-1
Example .....	3-2
Predefined Look-Up Tables .....	3-3
Equalize.....	3-3
Example 1 .....	3-4
Example 2 .....	3-4
Logarithmic and Inverse Gamma Correction.....	3-5
Exponential and Gamma Correction.....	3-7

## Chapter 4

### Operators

Concepts and Mathematics.....	4-1
Arithmetic Operators .....	4-2
Logic Operators .....	4-2
Truth Tables .....	4-3
Example 1 .....	4-4
Example 2 .....	4-5

## Chapter 5

### Spatial Filtering

Concept and Mathematics .....	5-1
Spatial Filter Classification Summary .....	5-3
Linear Filters or Convolution Filters .....	5-3
Gradient Filter .....	5-5
Example .....	5-5
Kernel Definition .....	5-6
Filter Axis and Direction .....	5-6
Examples .....	5-6
Edge Extraction and Edge Highlighting .....	5-7
Edge Thickness .....	5-9
Predefined Gradient Kernels.....	5-10
Prewitt Filters .....	5-10
Sobel Filters.....	5-11
Laplacian Filters.....	5-12
Example .....	5-12
Kernel Definition .....	5-13
Contour Extraction and Highlighting .....	5-14
Examples .....	5-14

Contour Thickness .....	5-15
Predefined Laplacian Kernels .....	5-16
Smoothing Filter .....	5-17
Example .....	5-17
Kernel Definition .....	5-17
Examples .....	5-18
Predefined Smoothing Kernels .....	5-19
Gaussian Filters .....	5-19
Example .....	5-20
Kernel Definition .....	5-20
Predefined Gaussian Kernels .....	5-21
Nonlinear Filters .....	5-22
Nonlinear Prewitt Filter .....	5-22
Nonlinear Sobel Filter .....	5-22
Example .....	5-23
Nonlinear Gradient Filter .....	5-24
Roberts Filter .....	5-24
Differentiation Filter .....	5-24
Sigma Filter .....	5-25
Lowpass Filter .....	5-25
Median Filter .....	5-26
Nth Order Filter .....	5-26
Examples .....	5-27

## Chapter 6

### Frequency Filtering

Introduction to Frequency Filters .....	6-1
Lowpass FFT Filters .....	6-2
Highpass FFT Filters .....	6-2
Mask FFT Filters .....	6-2
Definition .....	6-3
FFT Display .....	6-4
Standard Representation .....	6-5
Optical Representation .....	6-6
Frequency Filters .....	6-8
Lowpass Frequency Filters .....	6-8
Lowpass Attenuation .....	6-8
Lowpass Truncation .....	6-9
Highpass Frequency Filters .....	6-10
Highpass Attenuation .....	6-11
Highpass Truncation .....	6-11

## Chapter 7

### Morphology Analysis

Thresholding.....	7-1
Example .....	7-2
Thresholding a Color Image .....	7-3
Automatic Threshold.....	7-3
Clustering.....	7-3
Example of Clustering.....	7-4
Entropy .....	7-5
Metric.....	7-5
Moments .....	7-5
Interclass Variance.....	7-6
Structuring Element.....	7-6
Primary Morphology Functions .....	7-7
Erosion Function .....	7-8
Concept and Mathematics .....	7-8
Dilation Function .....	7-8
Concept and Mathematics .....	7-9
Erosion and Dilation Examples.....	7-9
Opening Function.....	7-11
Closing Function .....	7-11
Opening and Closing Examples .....	7-12
Inner Gradient Function .....	7-12
Outer Gradient Function .....	7-12
External and Internal Edge Example .....	7-13
Hit-Miss Function .....	7-13
Concept and Mathematics .....	7-14
Example 1 .....	7-14
Example 2 .....	7-15
Thinning Function.....	7-16
Examples .....	7-17
Thickening Function .....	7-18
Examples .....	7-19
Proper-Opening Function.....	7-20
Proper-Closing Function .....	7-21
Auto-Median Function .....	7-21
Advanced Binary Morphology Functions .....	7-22
Border Function .....	7-22
Hole Filling Function .....	7-22
Labeling Function .....	7-22
Lowpass and Highpass Filters.....	7-23
Lowpass and Highpass Example.....	7-24

Separation Function.....	7-24
Skeleton Functions .....	7-25
L-Skeleton Function.....	7-25
M-Skeleton Function.....	7-26
Skiz Function .....	7-26
Segmentation Function.....	7-27
Comparisons Between Segmentation and Skiz Functions.....	7-28
Distance Function .....	7-28
Danielsson Function .....	7-29
Example .....	7-29
Circle Function .....	7-30
Example .....	7-30
Convex Function .....	7-31
Example .....	7-31
Gray-Level Morphology .....	7-32
Erosion Function .....	7-32
Concept and Mathematics.....	7-32
Dilation Function.....	7-33
Concept and Mathematics.....	7-33
Erosion and Dilation Examples.....	7-33
Opening Function .....	7-34
Closing Function .....	7-34
Opening and Closing Examples .....	7-35
Proper-Opening Function .....	7-36
Proper-Closing Function .....	7-36
Auto-Median Function .....	7-37

## Chapter 8

### Quantitative Analysis

Spatial Calibration .....	8-1
Intensity Calibration .....	8-2
Digital Particles.....	8-2
Intensity Threshold.....	8-2
Connectivity .....	8-3
Connectivity-8.....	8-3
Connectivity-4.....	8-3
Area Threshold .....	8-4
Particle Measurements .....	8-5
Areas.....	8-5
Number of Pixels .....	8-5
Particle Area.....	8-5
Scanned Area .....	8-5



Ratio.....	8-5
Number of Holes .....	8-6
Holes' Area.....	8-6
Total Area .....	8-6
Lengths.....	8-6
Particle Perimeter .....	8-7
Holes' Perimeter .....	8-7
Breadth.....	8-7
Height .....	8-7
Coordinates .....	8-7
Center of Mass X and Center of Mass Y .....	8-8
Min(X, Y) and Max(X, Y).....	8-8
Max Chord X and Max Chord Y .....	8-9
Chords and Axes .....	8-9
Max Chord Length.....	8-9
Mean Chord X .....	8-9
Mean Chord Y .....	8-10
Max Intercept.....	8-10
Mean Intercept Perpendicular.....	8-10
Particle Orientation.....	8-10
Shape Equivalence .....	8-11
Equivalent Ellipse Minor Axis .....	8-12
Ellipse Major Axis.....	8-12
Ellipse Minor Axis.....	8-13
Ellipse Ratio .....	8-13
Rectangle Big Side .....	8-13
Rectangle Small Side.....	8-13
Rectangle Ratio.....	8-14
Shape Features .....	8-14
Moments of Inertia $I_{xx}$ , $I_{yy}$ , $I_{xy}$ .....	8-14
Elongation Factor .....	8-14
Compactness Factor.....	8-15
Heywood Circularity Factor .....	8-15
Hydraulic Radius .....	8-15
Waddel Disk Diameter .....	8-16
Definitions of Primary Measurements .....	8-16
Derived Measurements.....	8-16
Densitometry .....	8-18
Diverse Measurements.....	8-19

## Chapter 9

### Pattern Matching

Introduction to Pattern Matching .....	9-1
Pattern Matching Applications .....	9-2
What to Expect from a Pattern Matching Tool .....	9-3
Pattern Orientation and Multiple Instances .....	9-3
Ambient Lighting Conditions .....	9-4
Blur and Noise Conditions .....	9-5
Traditional Pattern Matching Techniques .....	9-5
Cross Correlation .....	9-5
Pyramidal Matching .....	9-7
Scale-Invariant Matching .....	9-7
New Pattern Matching Techniques .....	9-8
Using Pattern Matching .....	9-9
Defining and Creating Good Template Images .....	9-10
Using the Reference Pattern to Train the Pattern Matching Tool .....	9-11
Defining a Search Area .....	9-12
Setting Matching Parameters and Tolerances .....	9-13
Test the Search Tool on Test Images .....	9-13
Using a Ranking Method to Verify Results .....	9-14
Other Searching Techniques in IMAQ Vision .....	9-15
Binary Shape Matching .....	9-15
Blob Analysis .....	9-16
Search Using Edge Detection and Alignment Tools .....	9-18

## Appendix A

### Technical Support Resources

### Glossary

### Index

## Figures

Figure 1-1.	Rectangular Frame .....	1-7
Figure 1-2.	Hexagonal Frame .....	1-8
Figure 2-1.	Sample of a Linear Histogram .....	2-6
Figure 2-2.	Sample of a Cumulative Histogram .....	2-7
Figure 2-3.	Linear and Logarithmic Histograms .....	2-8
Figure 6-1.	FFT of an Image in Standard Representation .....	6-6
Figure 6-2.	FFT of an Image in Optical Representation.....	6-7
Figure 7-1.	Rectangular Frame, Neighborhood $3 \times 3$ .....	7-7
Figure 7-2.	Hexagonal Frame, Neighborhood $5 \times 3$ .....	7-7
Figure 9-1.	Example of a Common Fiducial .....	9-2
Figure 9-2.	Pattern Orientation and Multiple Instances.....	9-4
Figure 9-3.	Examples of Lightning Conditions .....	9-4
Figure 9-4.	Examples of Blur and Noise .....	9-5
Figure 9-5.	The Correlation Procedure .....	9-6
Figure 9-6.	Good Pattern Matching Sampling Techniques .....	9-8
Figure 9-7.	Edge Detection and Pattern Matching Techniques .....	9-9
Figure 9-8.	Selecting a Search Area .....	9-12
Figure 9-9.	Optical Character Verification .....	9-14
Figure 9-10.	Using Shape Matching to Search for Windshield Wiper Parts.....	9-16
Figure 9-11.	List of Parameters Extracted from a Particle in a Binary Image .....	9-17
Figure 9-12.	Using Edge Detection and Alignment Tools .....	9-19

## Tables

Table 1-1.	Bytes Per Pixel .....	1-4
Table 1-2.	Pixel Neighborhood Representations.....	1-7
Table 3-1.	LUT Transfer Functions .....	3-3
Table 4-1.	Arithmetic Operators .....	4-2
Table 4-2.	Logical Operators.....	4-2
Table 4-3.	Using Logical Operators with Binary Image Masks.....	4-3
Table 5-1.	Spatial Filter Classifications .....	5-3
Table 5-2.	Prewitt Filters.....	5-10
Table 5-3.	Sobel Filters .....	5-11
Table 5-4.	Gradient $5 \times 5$ .....	5-12
Table 5-5.	Gradient $7 \times 7$ .....	5-12

Table 5-6.	Laplacian $3 \times 3$ .....	5-16
Table 5-7.	Laplacian $5 \times 5$ .....	5-16
Table 5-8.	Laplacian $7 \times 7$ .....	5-16
Table 5-9.	Smoothing $3 \times 3$ .....	5-19
Table 5-10.	Smoothing $5 \times 5$ .....	5-19
Table 5-11.	Smoothing $7 \times 7$ .....	5-19
Table 5-12.	Gaussian $3 \times 3$ .....	5-21
Table 5-13.	Gaussian $5 \times 5$ .....	5-21
Table 5-14.	Gaussian $7 \times 7$ .....	5-21
Table 7-1.	How the Structure Element Affects Erosion .....	7-10
Table 7-2.	How the Structure Element Affects Dilation .....	7-11
Table 7-3.	Using the Hit-Miss Function .....	7-16
Table 7-4.	Effect of Lowpass and Highpass Filtering .....	7-23
Table 7-5.	Erosion and Dilation Examples .....	7-34
Table 8-1.	Example of Thresholding by Area .....	8-4
Table 8-2.	Derived Measurements .....	8-16

# About This Manual

---

The *IMAQ Vision User Manual* contains information you need to get started with IMAQ Vision.

This manual presents the basics of computer-based vision applications. This manual does not show you how to solve every possible imaging problem. Use the *IMAQ Vision for G Reference Manual* and the online reference for further function-specific information.

## Conventions

---



The following conventions appear in this manual:

This icon denotes a note, which alerts you to important information.

### **bold**

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.

### *italic*

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.

---

# Images

This chapter describes the algorithms and principles of image files and data structures.

## Introduction to Digital Images

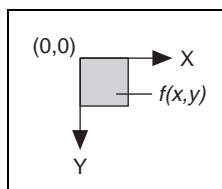
---

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 default the spatial reference of the pixel with the coordinates  $(0, 0)$  is located at the upper-left corner of the image. Notice in the following representation that the value of  $x$  increases moving from left to right, and  $y$  increases in the downward direction.



In *digital image processing*, an acquisition device converts an image into a discrete number of pixels. This device assigns a numeric location and *gray-level* value that specifies the brightness of pixels.

# Properties of a Digitized Image

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

## Image Resolution

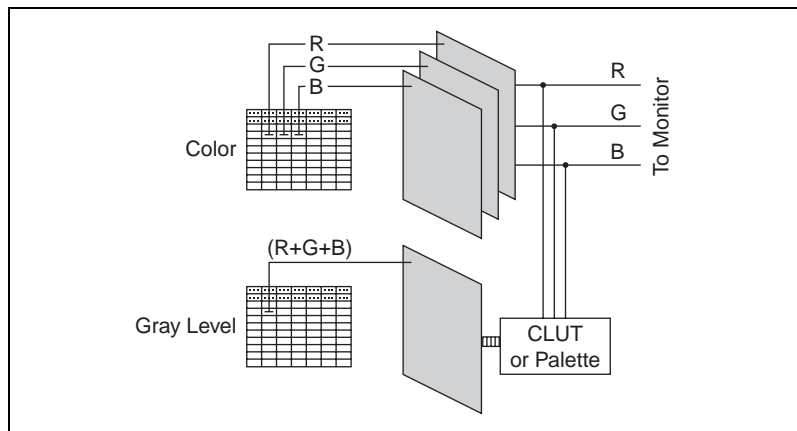
The *spatial resolution* of an image is its number of rows and columns of pixels. An image composed of  $m$  rows and  $n$  columns has a resolution of  $m \times n$ . This image has  $n$  pixels along its horizontal axis and  $m$  pixels along its vertical axis.

## Image Definition

The definition of an image, also called *pixel depth*, indicates the number of colors or shades that you can see in the image. Pixel depth is the number of bits used to code the intensity of a pixel. For a given definition of  $n$ , 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 from  $-32,768$  to 32,767.

## Number of Planes

The number of planes in an image is the number of arrays of pixels that compose the image. A gray-level or pseudo-color image is composed of one plane, while a true-color image is composed of three planes—one for the red component, one for the blue, and one for the green—as shown in the following figure.



In gray-level images, the red, green, and blue intensities (*RGB*) of a pixel combine to produce a single value. This single value is converted back to an RGB intensity when displayed on a monitor. This conversion is performed by a *color look-up table* (CLUT) transformation.

In three-plane or true color images, the red, green, and blue intensities of a pixel are coded into three different values. The image is the combination of three arrays of pixels corresponding to the red, green, and blue components.

## Image Types and Formats

---

The IMAQ Vision libraries can manipulate three types of images: *gray-level*, *color*, and *complex* images.

### Gray-Level Images

Gray-level images are composed of a single plane of pixels. Standard gray-level formats are 8-bit *BMP*, *TIFF*, *PNG*, *JPEG*, and *AIPD*. Standard 16-bit gray-level formats are PNG and AIPD. AIPD is an internal file format that offers the advantage of storing the spatial calibration of an image. Gray-level images that use other formats and have an 8-bit or 16-bit pixel depth can be imported into the IMAQ Vision libraries.

### Color Images

Color images are composed of pixels. These pixels are a composite of four values. In the RGB color model, each color is encoded with an 8-bit red, green, and blue value. In the HSL color model, each color is encoded with an 8-bit hue, saturation, and luminance value. In both color models, an additional 8-bit value goes unused. This representation is called both  $4 \times 8$  bit and 32 bit. Standard color file formats for RGB images are BMP, TIFF, PNG, JPEG, and AIPD.







### Complex Images

Complex images are composed of complex data in which pixel values have a real part and an imaginary part. Such images are derived from the Fast Fourier transform of gray-level images. Four representations of a complex image can be given: the real part, imaginary part, magnitude, and phase.

Table 1-1 shows how many bytes are used per pixel in gray-level, color, and complex images. For an identical spatial resolution, a color image occupies four times the memory space used by an 8-bit gray-level image, and a complex image occupies eight times this amount.



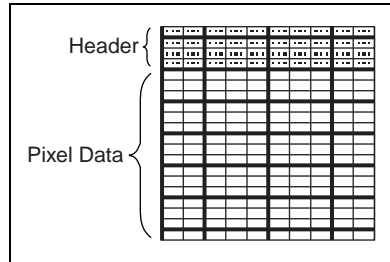
**Table 1-1.** Bytes Per Pixel

Image Type	Number of Bytes Per Pixel Data
<p><b>8-bit (Unsigned) Integer Gray-Level</b></p> <p>(1 byte or 8-bit)</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">8-bit for the gray-level intensity</p>
<p><b>16-bit (Signed) Integer Gray-Level</b></p> <p>(2 bytes or 16-bit)</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">16-bit for the gray-level intensity</p>
<p><b>32-bit Floating-Point Gray-Level</b></p> <p>(4 bytes or 32-bit)</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">32-bit floating for the gray-level intensity</p>
<p><b>RGB Color</b></p> <p>(4 bytes or 32-bit)</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">             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         </p>
<p><b>HSL Color</b></p> <p>(4 bytes or 32-bit)</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">             8 bits not used                  8-bit for the hue                  8-bit for the saturation                  8-bit for the luminance         </p>
<p><b>Complex</b></p> <p>(8 bytes or 64-bit)</p>	<div style="text-align: center;">  </div> <p style="text-align: center;">             32-bit floating for the real part                  32-bit floating for the imaginary part         </p>

# Image Files

---

An *image file* is composed of a header followed by pixel values. Depending on the file format, the header contains information such as the image horizontal and vertical resolution, its pixel definition, the physical calibration, and the original palette.



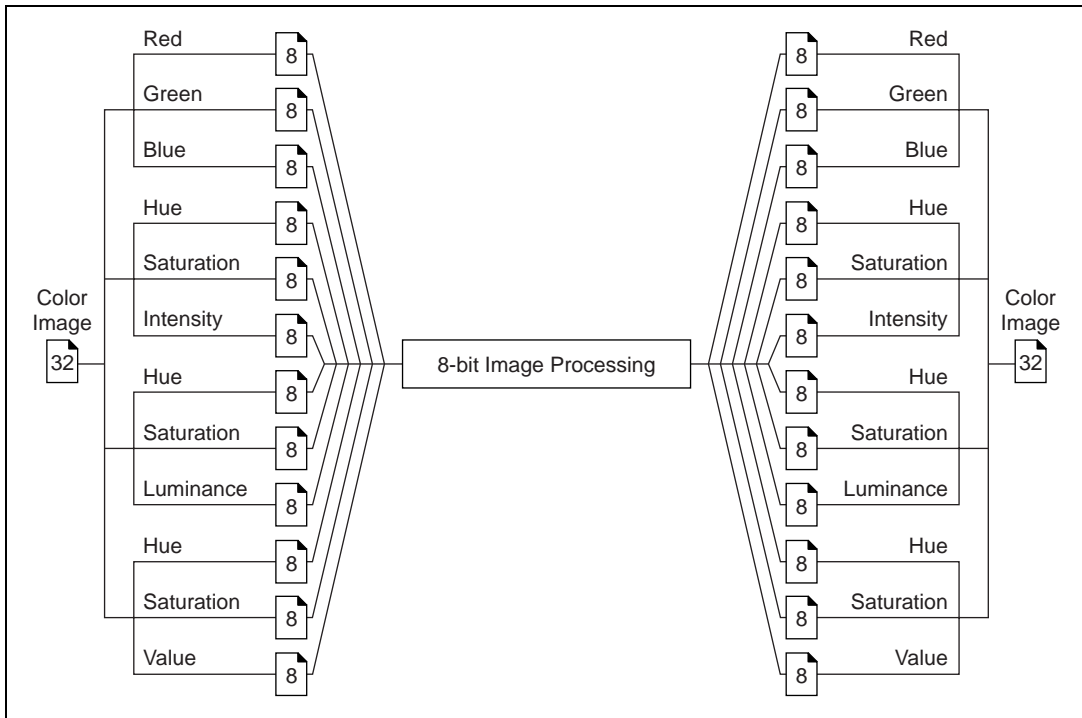
# Processing Color Images

---

Most image-processing and analysis functions apply to 8-bit images. However, you also can process color images by manipulating their color components individually.

You can break down a color image into various sets of primary components such as RGB (red, green, and blue), *HSI* (hue, saturation, and intensity), *HSL* (hue, saturation, and luminance), or *HSV* (hue, saturation, and value). Each component becomes an 8-bit image and can be processed as any gray-level image.

You can reassemble a color image later from a set of three 8-bit images taking the place of its RGB, HSL, or HSV components.




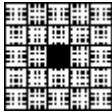
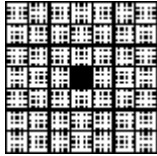
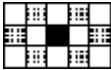
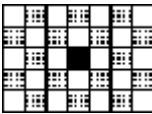
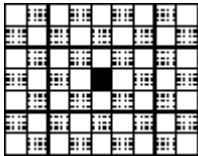
## Image Pixel Frame

The notion of pixel frame is important for a category of image processing functions called *neighborhood operations*. These functions alter the value of pixels depending on the intensity values of their neighbors. They include *spatial filters*, which alter the intensity of a pixel with respect to variations in intensities of neighboring pixels, and *morphological transformations*, which extract and alter the structure of objects in an image.

A digital image is a two-dimensional array of pixel values. From this definition, you might assume that pixels are arranged only in a regular rectangular frame. However, from an image processing point of view, you can consider another grid arrangement, such as a hexagonal pixel frame. A hexagonal pixel frame offers the advantage that the six neighbors of a pixel are equidistant.

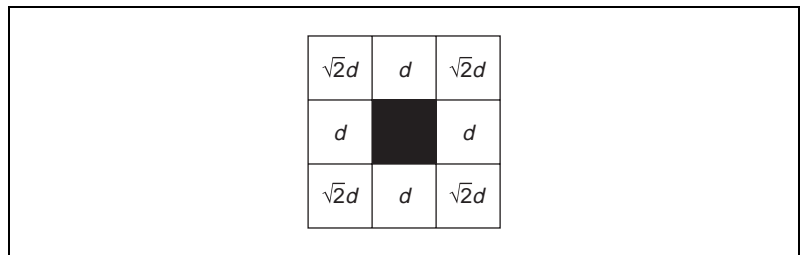
The pixels in an image are arranged in a rectangular grid. However, some image processing algorithms can reproduce a hexagonal neighborhood using the representations illustrated in Table 1-2. The pixels considered as neighbors of the given pixel (shown in solid) are indicated by the shaded pattern.

**Table 1-2.** Pixel Neighborhood Representations

Pixel Frame	Neighborhood Size		
Rectangular	$3 \times 3$ 	$5 \times 5$ 	$7 \times 7$ 
Hexagonal	$5 \times 3$ 	$7 \times 5$ 	$9 \times 7$ 

## Rectangular Frame

Figure 1-1 shows how each pixel in a rectangular frame is surrounded by eight neighbors.

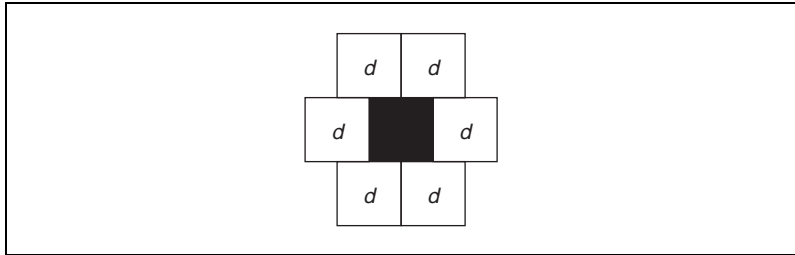


**Figure 1-1.** Rectangular Frame

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.

## Hexagonal Frame

Figure 1-2 shows how each pixel in a hexagonal frame is surrounded by six neighbors. Each neighbor is found at an equal distance  $d$  from the central pixel.



**Figure 1-2.** Hexagonal Frame

---

# Tools and Utilities

This chapter describes the tools and utilities used in IMAQ Vision.

## Palettes

---

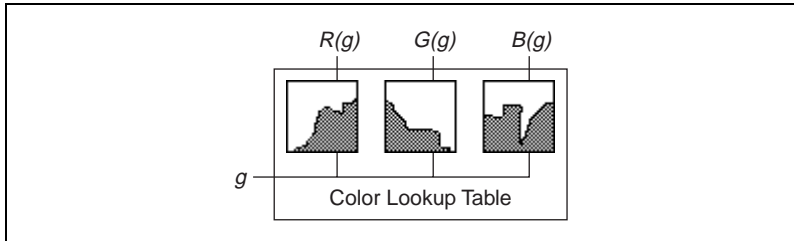
At the time an image is displayed on the screen, the value of each pixel is converted into a red, green, and blue intensity that produces a color. This conversion is defined in a table called a color look-up table (CLUT). For 8-bit images, the CLUT associates a color to each gray-level value and produces a gradation of colors, called a *palette*.

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

In the case of 8-bit resolution, pixels can take  $2^8$  or 256 values ranging from 0 to 255. A black and white palette associates different shades of gray to each value so as to produce a linear and continuous gradation of gray, from black to white. At this point, the palette can be set up 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.

The gray-level value of a pixel acts as an address that is indexed into three tables, with three values corresponding to a red, green, and blue (RGB) intensity. This set of three conversion tables defines a palette in which varying amounts of red, green, and blue are mixed to produce a color representation of the value range [0, 255].



Five color palettes are predefined in IMAQ Vision. Each palette emphasizes different shades of gray. However, they all use the following conventions:

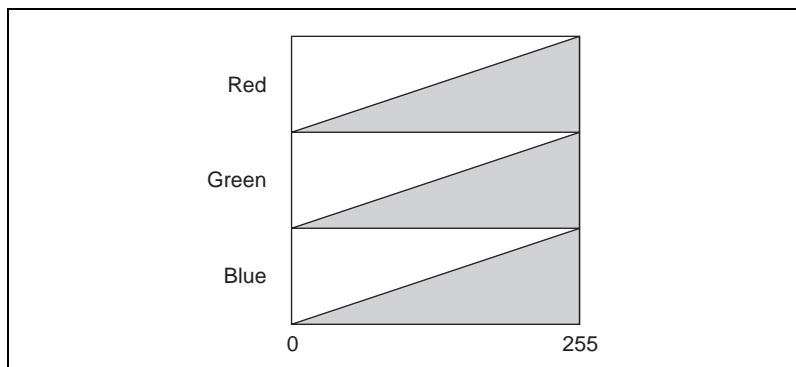
- Gray level 0 is assigned to black.
- Gray level 255 is assigned to white.

Because of these conventions, you can associate bright areas to the presence of pixels with high gray-level values, and dark areas to the presence of pixels with low gray-level values.

The following sections introduce the five predefined palettes. The graphs in each section represent the three RGB look-up tables used by each palette. The horizontal axes of the graphs represent the input gray-level range  $[0, 255]$ , while the vertical axes give the RGB intensities assigned to a given gray-level value.

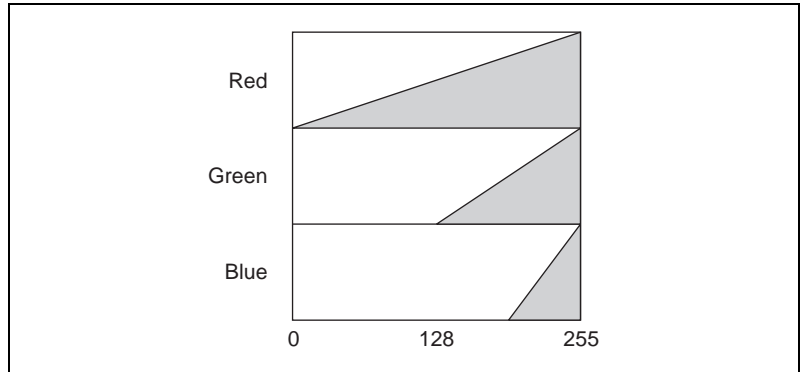
## Gray Palette

This palette has a gradual gray-level variation from black to white. Each value is assigned to an equal amount of the RGB intensities.



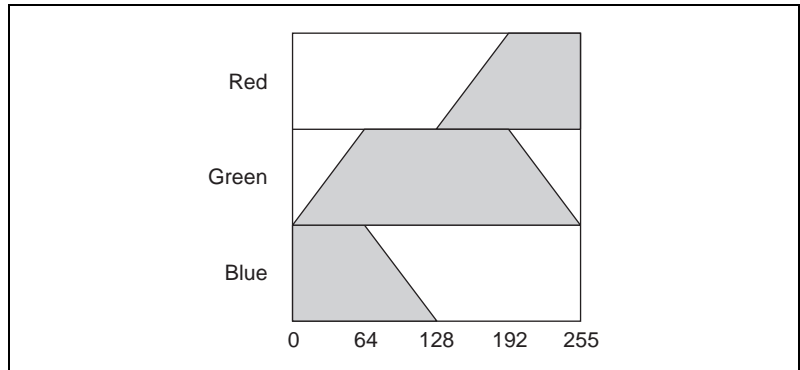
## Temperature Palette

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



## Rainbow Palette

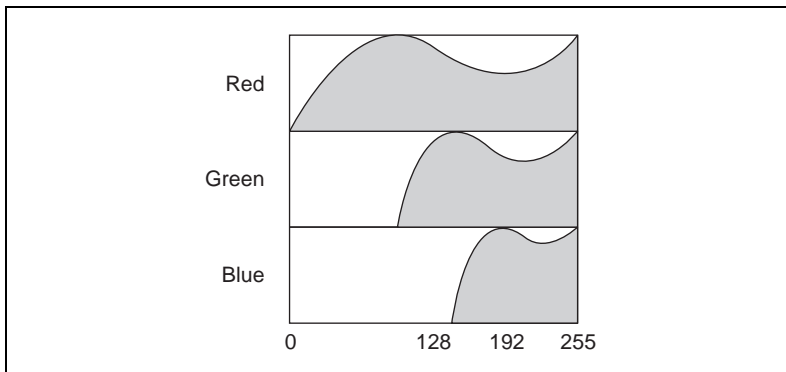
This palette has a gradation from blue to red with a prominent range of greens in the middle value range. 0 is black and 255 is white.





## Gradient Palette

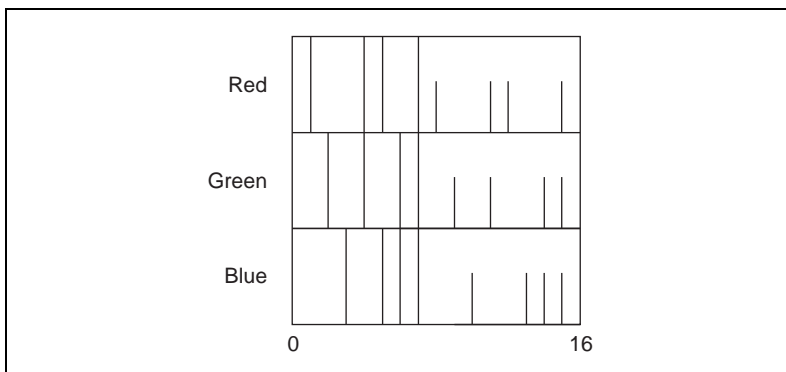
This palette has a gradation from red to white with a prominent range of light blue in the upper value range. 0 is black and 255 is white.



## Binary Palette

This palette has 16 cycles of 16 different colors, where  $g$  is the gray-level value and

$g = 0$  corresponds to  $R = 0, G = 0, B = 0$ , which appears black;  
 $g = 1$  corresponds to  $R = 1, G = 0, B = 0$ , which appears red;  
 $g = 2$  corresponds to  $R = 0, G = 1, B = 0$  which appears green;  
 and so forth.



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

# Image Histogram

The *histogram* of an image indicates the quantitative distribution of pixels per gray-level value. It provides a general description of the appearance of an image and helps identify various components such as the background, objects, and noise.

## Definition

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

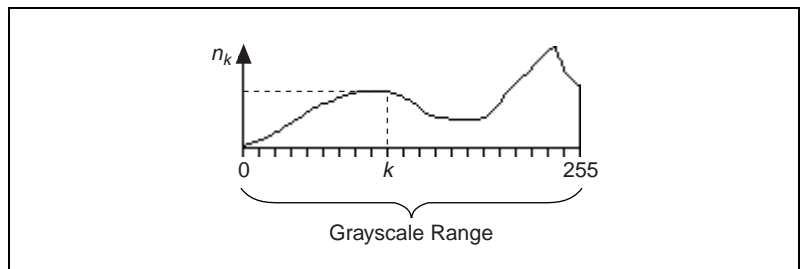
$$H(k) = n_k$$

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.



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

In both cases, the horizontal axis represents the gray-level range 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  $k$ .

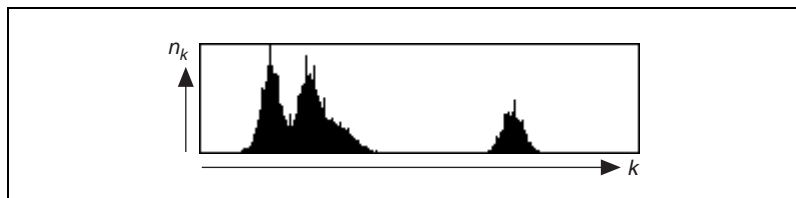


Figure 2-1. Sample of a Linear Histogram

## Cumulative Histogram

The distribution function is

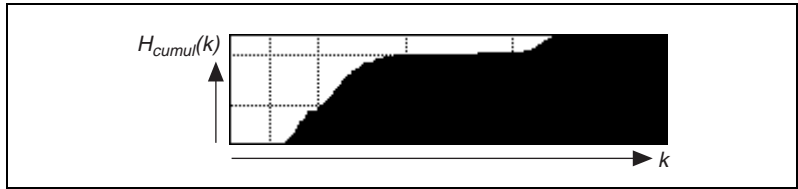
$$H_{Cumul}(k) = \sum_0^k n_k$$

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

The probability function is

$$P_{Cumul}(k) = \sum_0^k \frac{n_k}{n}$$

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



**Figure 2-2.** Sample of a Cumulative Histogram

## Interpretation

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

In Figure 2-1, the linear histogram reveals that the image is composed of three major elements. The cumulative histogram of the same image in Figure 2-2 shows that the two left-most peaks compose approximately 80% of the image, while the remaining 20% corresponds to the third peak.

## Histogram of Color Images

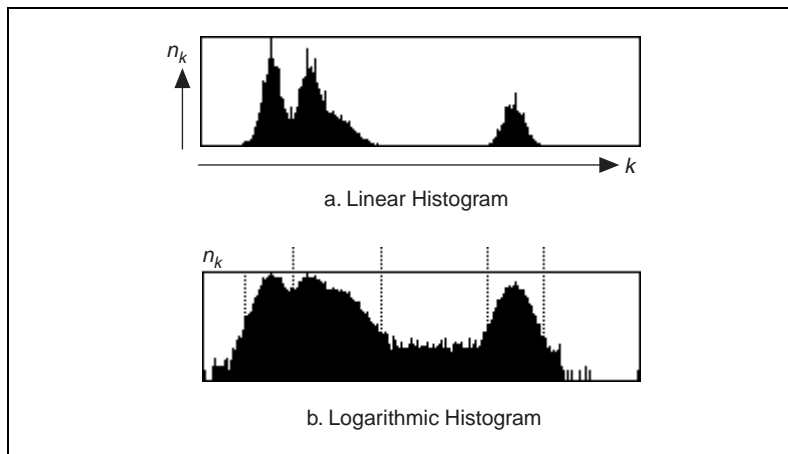
The histogram of a color image is expressed as a series of three tables corresponding to the histograms of the three primary components ( $R$ ,  $G$ , and  $B$ ;  $H$ ,  $S$ , and  $I$ ;  $H$ ,  $S$ , and  $L$ ; or  $H$ ,  $S$ , and  $V$ ).

## 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 the case of 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 is made more prominent at the expense of the dominant gray-level values.

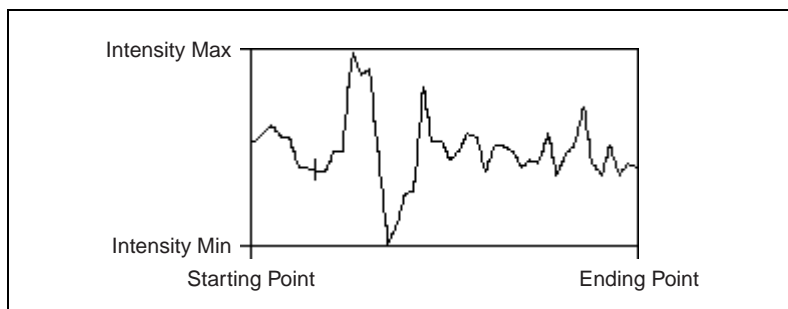
Figure 2-3 illustrates the difference between the display of the histogram of the same image in a linear and logarithmic scale. In this particular image, three pixels are equal to 0. This information is unobservable in the linear representation of the histogram but evident in the logarithmic representation.



**Figure 2-3.** Linear and Logarithmic Histograms

## Line Profile

A *line profile* plots the variations of intensity along a line. This utility is helpful for examining boundaries between components, quantifying the magnitude of intensity variations, and detecting the presence of repetitive patterns. The following figure illustrates a line profile.

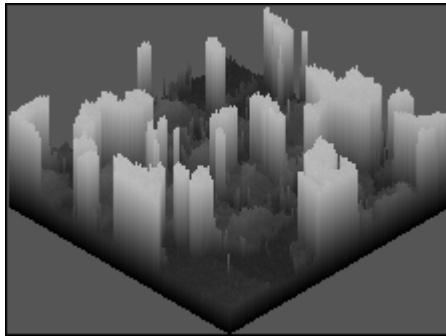


The peaks and valleys reveal increases and decreases of the light intensity along the line selected in the image. Their width and magnitude are proportional to the size and intensity of their related regions.

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.

## 3D View

The *3D view*, illustrated in the following graphic, displays a three-dimensional perspective of the light intensity in an image. It gives a relief map of the image in which high-intensity values are associated to summits and low-intensity values are associated to valleys.



---

# Look-up Table Transformations

This chapter provides an overview of look-up table transformations.

## About Look-up Table Transformations

---

The *look-up table* (LUT) transformations are basic image-processing functions that you can use to improve the contrast and brightness of an image by modifying the dynamic intensity of regions with poor contrast. LUT transformations can highlight details in areas containing significant information, at the expense of other areas. These functions include *histogram equalization*, *histogram inversion*, *gamma corrections*, *inverse gamma corrections*, *logarithmic corrections*, and *exponential corrections*.

A LUT transformation converts input gray-level values (those from the source image) into other gray-level values (in the transformed image). The transfer function has an intended effect on the brightness and contrast of the image.

Each input gray-level value is given a new value such that

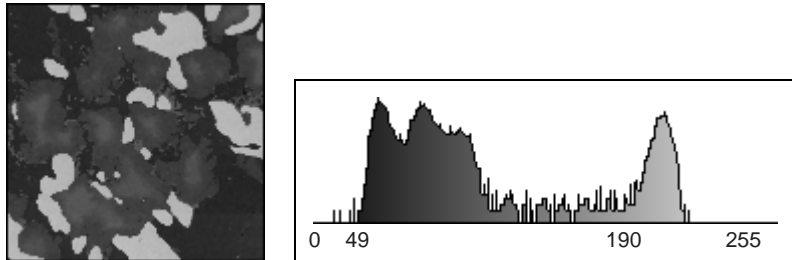
$$\text{output value} = F(\text{input value})$$

where  $F$  is a linear or nonlinear, continuous or discontinuous transfer function defined over the interval  $[0, \text{max}]$ .

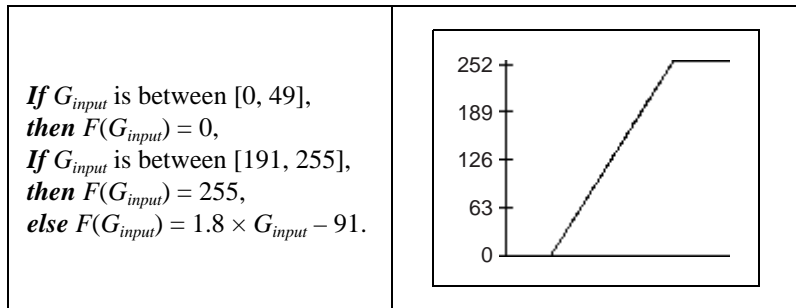
In the case of an 8-bit resolution, a LUT is a table of 256 elements. Each element of the array represents an input gray-level value. Its content indicates the output value.

## Example

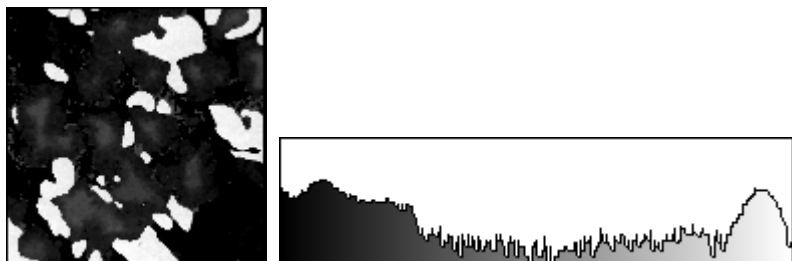
This example uses the following source image. In the linear histogram of the source image, the gray-level intervals [0, 49] and [191, 255] do not contain significant information.



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, 255], 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 Look-Up Tables

Seven predefined LUTs are available in IMAQ Vision: Equalize, Logarithmic, Power 1/Y, Square Root, Exponential, Power Y, and Square.

Table 3-1 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 B&W or Gray palette).

**Table 3-1.** LUT Transfer Functions

LUT	Transfer Function	Shading Correction
Equalize		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.

## Equalize

The *Equalize* function alters the gray-level value of pixels so they become distributed evenly 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 of images in which not all gray levels are used.

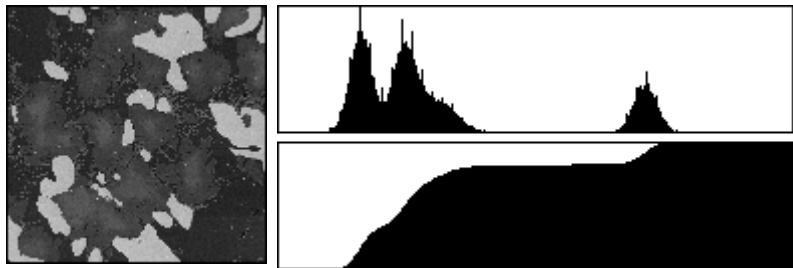
The equalization can be limited to a gray-level interval, also called the equalization range. In this case, the function evenly distributes the pixels belonging to the equalization range over the full interval (0 to 255 for an 8-bit image) and the other pixels are set to 0. The image produced reveals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

## Example 1

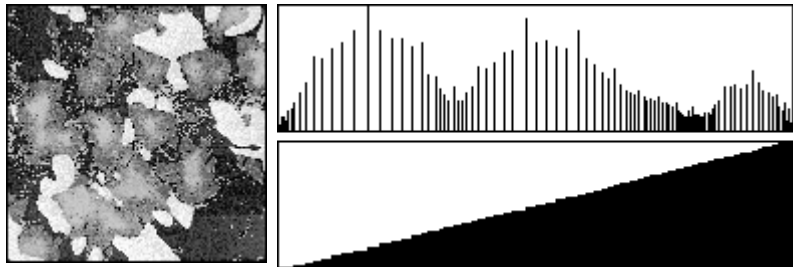
This example shows how an equalization of the interval  $[0, 255]$  can spread the information contained in the three original peaks over larger intervals. The transformed image reveals more details about each component in the original image. The following graphics show the original image and histograms.



**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.

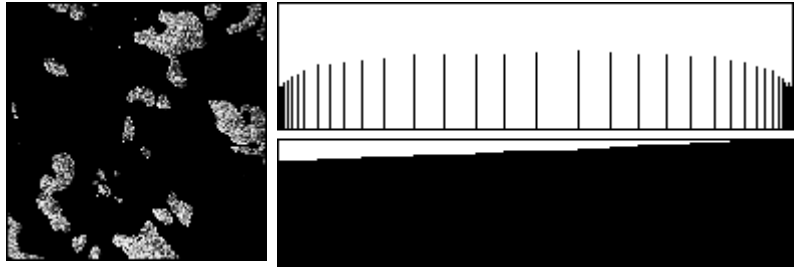


**Note** The cumulative histogram of an image after a histogram equalization always has a linear profile, as seen in the preceding example.

## 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

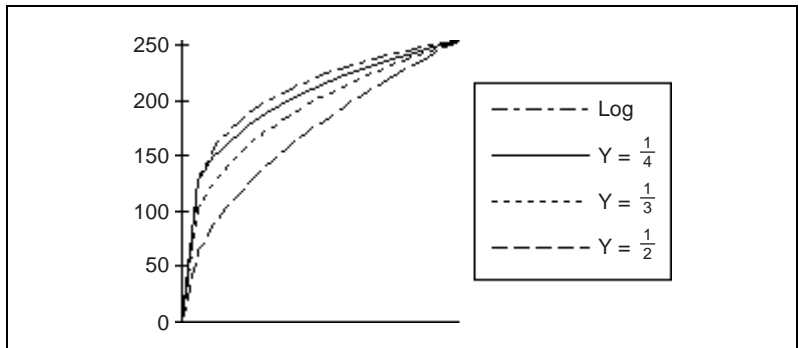
other components are set to black. An equalization from [166, 200] to [0, 255] produces the following image and histograms.



## 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.

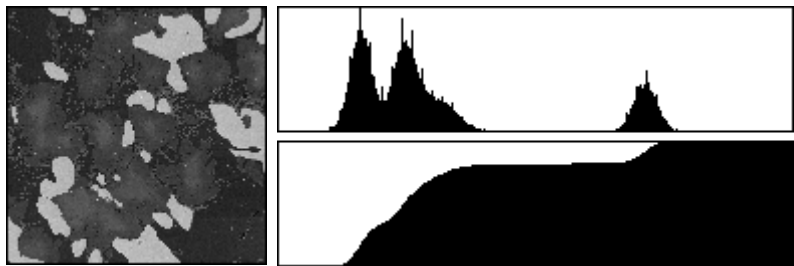
The higher the gamma coefficient  $Y$ , the stronger the intensity correction. The Logarithmic correction has a stronger effect than the Power  $1/Y$  function.

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.

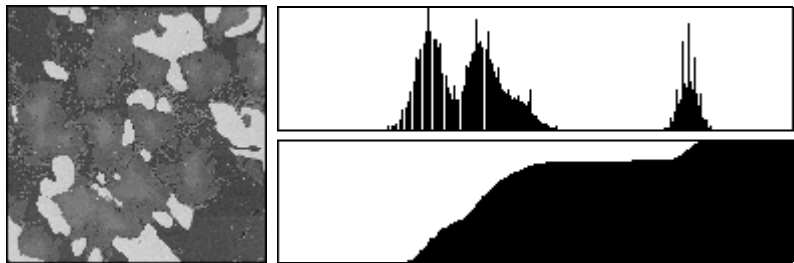


**Note** Graphics on the left represent the original image, graphics on the top right represent the linear histogram, and graphics on the bottom right represent the cumulative histogram.

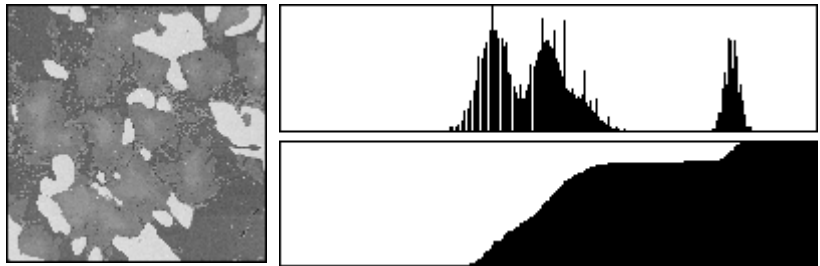
The following graphic shows the original image and histograms.



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.



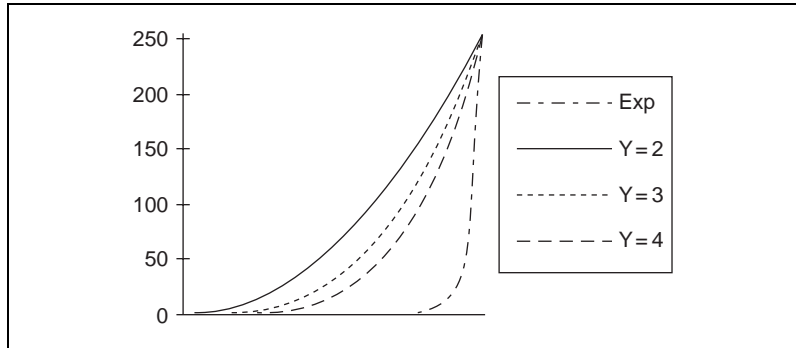
A Logarithm transformation produces the following image and histograms.



## 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.

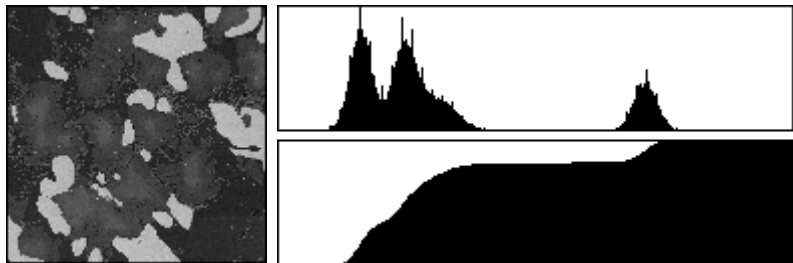
The higher the gamma coefficient  $Y$ , the stronger the intensity correction. The Exponential correction has a stronger effect than the Power  $Y$  function.

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 left, the darker the image.

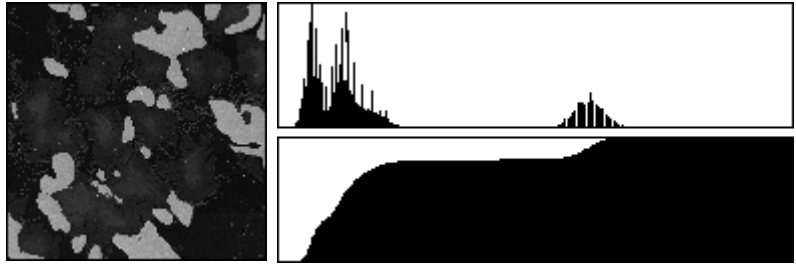


**Note** Graphics on the left represent the original image, graphics on the top right represent the linear histogram, and graphics on the bottom right represent the cumulative histogram.

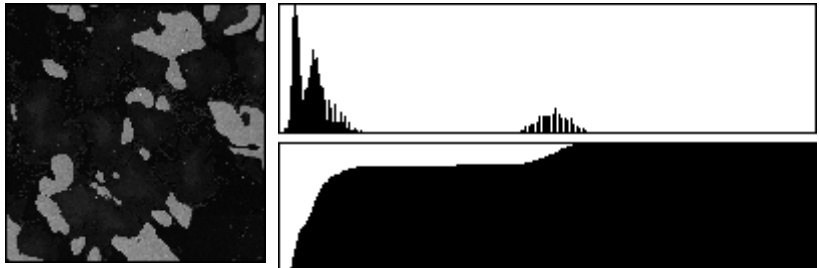
The following graphic shows the original image and histograms.



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.



# Operators

This chapter describes the arithmetic and logic operators used in IMAQ Vision.

## Concepts and Mathematics

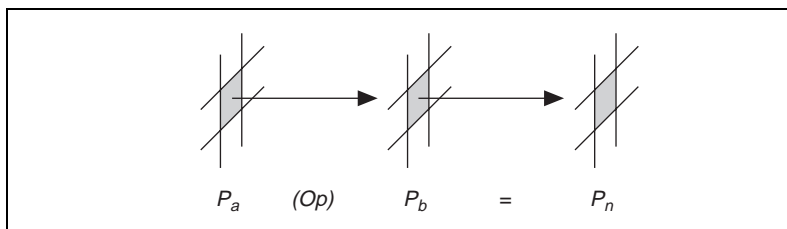
Arithmetic and logic *operators* mask, combine, and compare images. Common applications of these operators include time-lapse comparisons, identification of the union or intersection between images, and comparisons between several images and a model. Operators also can be used to *threshold* or *mask* images and to alter contrast and brightness.

An arithmetic or logic operation between images is a pixel-by-pixel transformation. It produces an image in which each pixel derives 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  $N$  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 4-1 describe the usage of *arithmetic operators* with 8-bit resolution images.

**Table 4-1.** 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	$p_n = \max(p_a - p_b, 0)$
Remainder	$p_n = p_a \bmod p_b$

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 Operators

*Logic operators* are bitwise operators. They manipulate gray-level values coded on one byte at the bit level. The equations in Table 4-2 describe the usage of logical operators. The truth tables for logic operators are presented in the *Truth Tables* section in this chapter.

**Table 4-2.** Logical Operators

Operator	Equation
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	$p_n = p_a \text{ XOR } p_b$
Difference	$p_n = p_a \text{ AND } (\text{NOT } p_b)$
Mask	<i>if</i> $p_b = 0$ , <i>then</i> $p_n = 0$ , <i>else</i> $p_n = p_a$

**Table 4-2.** Logical Operators (Continued)

Operator	Equation
Mean	$p_n = \text{mean}[p_a, p_b]$
Max	$p_n = \text{max}[p_a, p_b]$
Min	$p_n = \text{min}[p_a, p_b]$

In the case of images with 8-bit resolution, logic operators mainly are 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).

Table 4-3 illustrates how logic operators can be used to extract or remove information in an image.

**Table 4-3.** 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.

<b>AND</b>	
	$b = 0$ $b = 1$
$a = 0$	0    0
$a = 1$	0    1

<b>NAND</b>	
	$b = 0$ $b = 1$
$a = 0$	1    1
$a = 1$	1    0

<b>OR</b>	
-----------	--

	<b><math>b = 0</math> <math>b = 1</math></b>	
<b><math>a = 0</math></b>	0	1
<b><math>a = 1</math></b>	1	1

<b>NOR</b>	
------------	--

	<b><math>b = 0</math> <math>b = 1</math></b>	
<b><math>a = 0</math></b>	1	0
<b><math>a = 1</math></b>	0	0

<b>XOR</b>	
------------	--

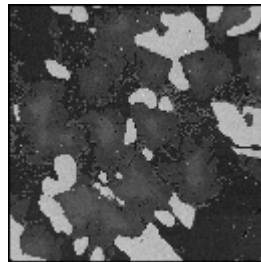
	<b><math>b = 0</math> <math>b = 1</math></b>	
<b><math>a = 0</math></b>	0	1
<b><math>a = 1</math></b>	1	0

<b>NOT</b>	
------------	--

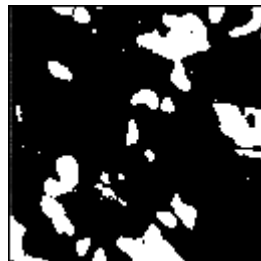
	<b>NOT <math>a</math></b>
<b><math>a = 0</math></b>	1
<b><math>a = 1</math></b>	0

## Example 1

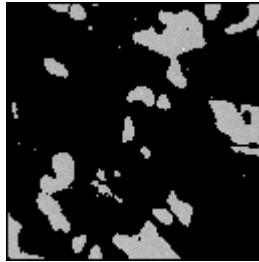
The following series of graphics illustrates images in which regions of interest have been isolated in a binary format, retouched with morphological manipulations, and finally multiplied by 255. The following gray-level source image is used for this example.



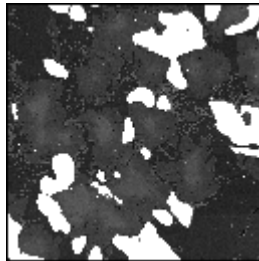
The following *mask image* results.



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



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

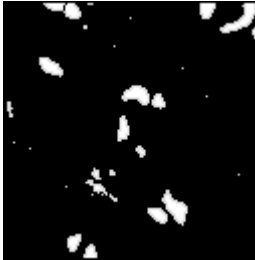


## Example 2

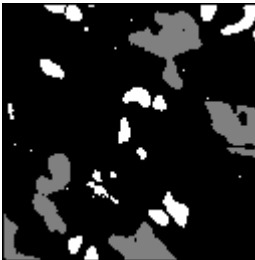
An image reveals two groups of objects that require different processing results in two binary images. Multiplying each binary image by a constant and applying an OR operation produces an image that shows their union, as illustrated in the following series of graphics. The following image illustrates *Object Group #1*  $\times$  128.



The following image illustrates *Object Group #2*  $\times$  255.



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



---

# Spatial Filtering

This chapter provides an overview of the linear and nonlinear spatial filters used in IMAQ Vision.

## Concept and Mathematics

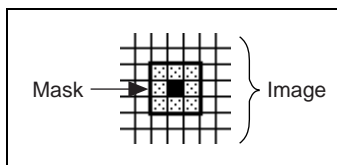
---

*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 can serve a variety of purposes, such as detecting edges along a specific direction, contouring patterns, reducing noise, and detail outlining or smoothing.

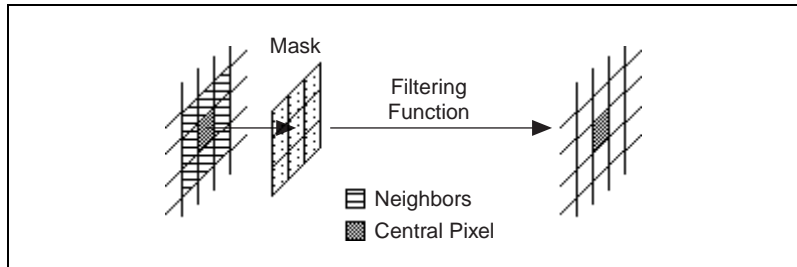
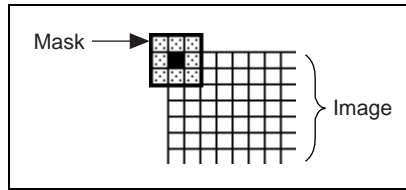
Spatial filters fall into two categories:

- *Highpass filters* emphasize significant variations of the light intensity usually found at the boundary of objects.
- *Lowpass filters* attenuate variations of the light intensity. They have the tendency to smooth images by eliminating details and blurring edges.

In the case of a  $3 \times 3$  matrix as illustrated in the following illustration, the value of the central pixel (shown in solid) derives from the values of its eight surrounding neighbors (shown in shaded pattern).



A  $5 \times 5$  matrix specifies 24 neighbors, a  $7 \times 7$  matrix specifies 48 neighbors, and so forth.



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:

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

## Spatial Filter Classification Summary

Table 5-1 describes the classification of spatial filters.

**Table 5-1.** Spatial Filter Classifications

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

## Linear Filters or Convolution Filters

A *convolution* is a mathematical function that 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*.

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.

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.

Finally, 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

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

then

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

If the convolution kernel is

$$\begin{array}{ccc} 0 & 1 & 0 \\ 1 & \mathbf{0} & 1 \\ 0 & 1 & 0 \end{array}$$

then

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

If the 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.

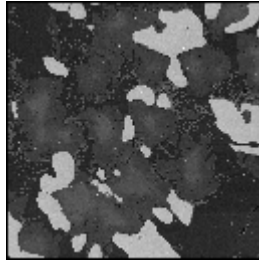
If all coefficients in the kernel are positive, the transfer function is equivalent to a weighted summation and produces a smoothing or lowpass filter. Typical lowpass filters include smoothing and Gaussian filters.

## Gradient Filter

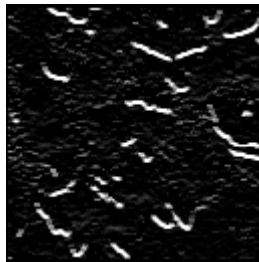
A *gradient filter* highlights the variations of light intensity along a specific direction, which has the effect of outlining edges and revealing texture.

### Example

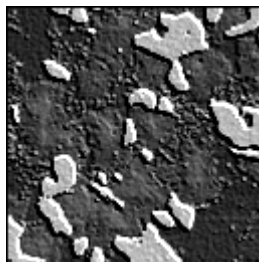
This example uses the following source image.



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:

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

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

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.

## Filter Axis and Direction

The *axis of symmetry* of the gradient kernel gives the orientation of the edges to outline. For example:

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

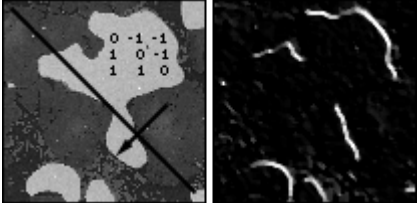
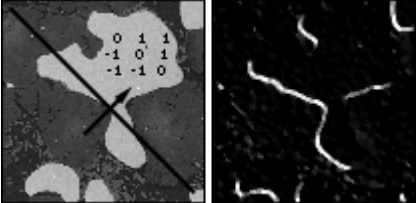
$$\begin{array}{ccc} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{array}$$

The axis of symmetry is 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.

## Examples

The following two kernels emphasize edges oriented at  $135^\circ$ .

Gradient #1	Gradient #2
$\begin{matrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{matrix}$ <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> 	$\begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix}$ <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> 

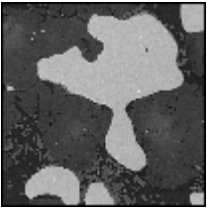
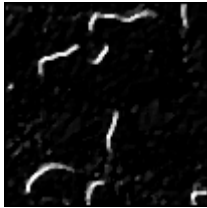


**Note** Applying Gradient #1 to an image gives the same results as applying Gradient #2 to its photometric negative, because reversing the look-up table of an image converts bright regions into dark regions and vice versa.

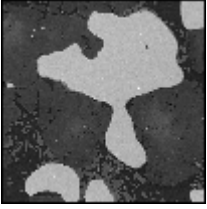
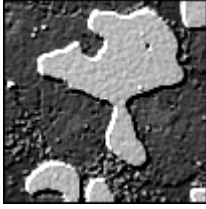
## Edge Extraction and Edge Highlighting

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

- 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. You can use this type of kernel for grain extraction and perception of texture.

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

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

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



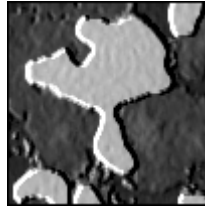
**Note** The convolution filter using the second kernel on the right side of the equation reproduces the source image. All neighboring pixels are multiplied by 0 and the central pixel remains equal to itself:  $(P_{(i,j)} = 1 \times P_{(i,j)})$ .

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

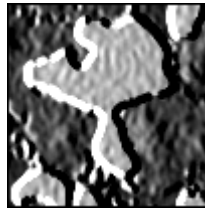
$$\text{Gradient \#2} = \text{Gradient \#1} + \text{Source Image}$$

## 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  $7 \times 7$ .



## 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 5-2.** Prewitt Filters

W/Edge	W/Image	SW/Edge	SW/Image
-1 0 1	-1 0 1	0 1 1	0 1 1
-1 <b>0</b> 1	-1 <b>1</b> 1	-1 <b>0</b> 1	-1 <b>1</b> 1
-1 0 1	-1 0 1	-1 -1 0	-1 -1 0
S/Edge	S/Image	SE/Edge	SE/Image
1 1 1	1 1 1	1 1 0	1 1 0
0 <b>0</b> 0	0 <b>1</b> 0	1 <b>0</b> -1	1 <b>1</b> -1
-1 -1 -1	-1 -1 -1	0 -1 -1	0 -1 -1
E/Edge	E/Image	NE/Edge	NE/Image
1 0 -1	1 0 -1	0 -1 -1	0 -1 -1
1 <b>0</b> -1	1 <b>1</b> -1	1 <b>0</b> -1	1 <b>1</b> -1
1 0 -1	1 0 -1	1 1 0	1 1 0
N/Edge	N/Image	NW/Edge	NW/Image
-1 -1 -1	-1 -1 -1	-1 -1 0	-1 -1 0
0 <b>0</b> 0	0 <b>1</b> 0	-1 <b>0</b> 1	-1 <b>1</b> 1
1 1 1	1 1 1	0 1 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 5-3.** Sobel Filters

<b>W/Edge</b>	<b>W/Image</b>	<b>SW/Edge</b>	<b>SW/Image</b>
-1 0 1	-1 0 1	0 1 2	0 1 2
-2 <b>0</b> 2	-2 <b>1</b> 2	-1 <b>0</b> 1	-1 <b>1</b> 1
-1 0 1	-1 0 1	-2 -1 0	-2 -1 0
<b>S/Edge</b>	<b>S/Image</b>	<b>SE/Edge</b>	<b>SE/Image</b>
1 2 1	1 2 1	2 1 0	2 1 0
0 <b>0</b> 0	0 <b>1</b> 0	1 <b>0</b> -1	1 <b>1</b> -1
-1 -2 -1	-1 -2 -1	0 -1 -2	0 -1 -2
<b>E/Edge</b>	<b>E/Image</b>	<b>NE/Edge</b>	<b>NE/Image</b>
1 0 -1	1 0 -1	0 -1 -2	0 -1 -2
2 <b>0</b> -2	2 <b>1</b> -2	1 <b>0</b> -1	1 <b>1</b> -1
1 0 -1	1 0 -1	2 1 0	2 1 0
<b>N/Edge</b>	<b>N/Image</b>	<b>NW/Edge</b>	<b>NW/Image</b>
-1 -2 -1	-1 -2 -1	-2 -1 0	-2 -1 0
0 <b>0</b> 0	0 <b>1</b> 0	-1 <b>0</b> 1	-1 <b>1</b> 1
1 2 1	1 2 1	0 1 2	0 1 2



The following tables list the predefined gradient  $5 \times 5$  and  $7 \times 7$  kernels.

**Table 5-4.** Gradient  $5 \times 5$

W/Edge	W/Image
0 -1 0 1 0	0 -1 0 1 0
-1 -2 0 2 1	-1 -2 0 2 1
-1 -2 <b>0</b> 2 1	-1 -2 <b>1</b> 2 1
-1 -2 0 2 1	-1 -2 0 2 1
0 -1 0 1 0	0 -1 0 1 0

**Table 5-5.** Gradient  $7 \times 7$

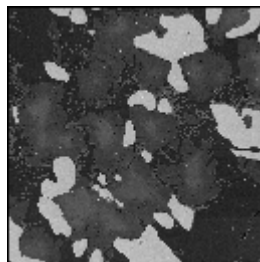
W/Edge	W/Image
0 -1 -1 0 1 1 0	0 -1 -1 0 1 1 0
-1 -2 -2 0 2 2 1	-1 -2 -2 0 2 2 1
-1 -2 -3 0 3 2 1	-1 -2 -3 0 3 2 1
-1 -2 -3 <b>0</b> 3 2 1	-1 -2 -3 <b>1</b> 3 2 1
-1 -2 -3 0 3 2 1	-1 -2 -3 0 3 2 1
-1 -2 -3 0 3 2 1	-1 -2 -3 0 3 2 1
0 -1 -1 0 1 1 0	0 -1 -1 0 1 1 0

## Laplacian Filters

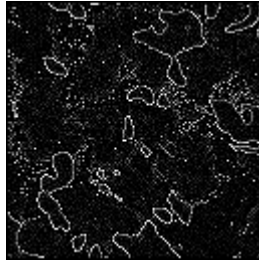
A *Laplacian filter* highlights the variation of the light intensity surrounding a pixel. The filter extracts the contour of objects and outlines details. Unlike the gradient filter, it is omnidirectional.

### Example

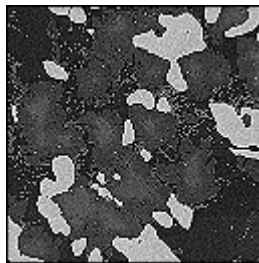
This example uses the following source image.



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:

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

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:

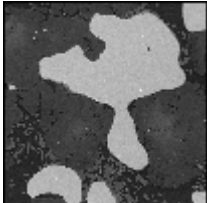

$$x \geq 2(|a| + |b| + |c| + |d|)$$

## Contour Extraction and Highlighting

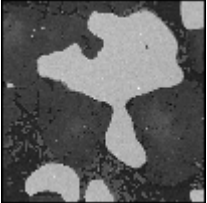

If the central coefficient is equal to this sum ( $x = 2(|a| + |b| + |c| + |d|)$ ), the Laplacian filter extracts the pixels where significant variations of light intensity are found. The presence of sharp edges, boundaries between objects, modification in the texture of a background, noise, and other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

### Examples

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

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

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
	$\begin{matrix} -1 & -1 & -1 \\ -1 & \mathbf{9} & -1 \\ -1 & -1 & -1 \end{matrix}$	

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

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



**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:  $(P_{(i,j)} = 1 \times P_{(i,j)})$ .

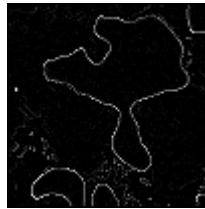
This equation indicates that the Laplacian #2 kernel adds the contours extracted by the Laplacian #1 kernel to the source image.

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

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

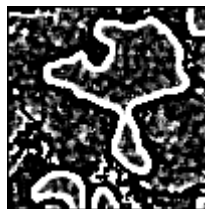
Larger kernels correspond to thicker contours. The following image is a Laplacian  $3 \times 3$ .



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



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



## Predefined Laplacian Kernels

The following tables list the predefined Laplacian kernels.

**Table 5-6.** Laplacian 3 × 3

<b>Contour 4</b>	<b>+ Image × 1</b>	<b>+ Image × 2</b>
0 -1 0	0 -1 0	0 -1 0
-1 <b>4</b> -1	-1 <b>5</b> -1	-1 <b>6</b> -1
0 -1 0	0 -1 0	0 -1 0
<b>Contour 8</b>	<b>+ Image × 1</b>	<b>+ Image × 2</b>
-1 -1 -1	-1 -1 -1	-1 -1 -1
-1 <b>8</b> -1	-1 <b>9</b> -1	-1 <b>10</b> -1
-1 -1 -1	-1 -1 -1	-1 -1 -1
<b>Contour 12</b>	<b>+ Image × 1</b>	
-1 -2 -1	-1 -2 -1	
-2 <b>12</b> -2	-2 <b>13</b> -2	
-1 -2 -1	-1 -2 -1	

**Table 5-7.** Laplacian 5 × 5

<b>Contour 24</b>	<b>+ Image × 1</b>
-1 -1 -1 -1 -1	-1 -1 -1 -1 -1
-1 -1 -1 -1 -1	-1 -1 -1 -1 -1
-1 -1 <b>24</b> -1 -1	-1 -1 <b>25</b> -1 -1
-1 -1 -1 -1 -1	-1 -1 -1 -1 -1
-1 -1 -1 -1 -1	-1 -1 -1 -1 -1

**Table 5-8.** Laplacian 7 × 7

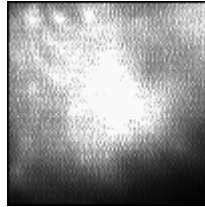
<b>Contour 48</b>	<b>+ Image × 1</b>
-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 <b>48</b> -1 -1 -1	-1 -1 -1 <b>49</b> -1 -1 -1
-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1

## Smoothing Filter

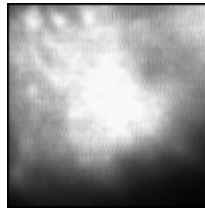
A *smoothing filter* attenuates the variations of light intensity in the neighborhood of a pixel. It smooths the overall shape of objects, blurs edges, and removes details.

### Example

This example uses the following source image.



A smoothing filter produces the following image.



### Kernel Definition

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

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

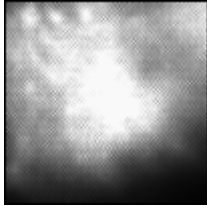
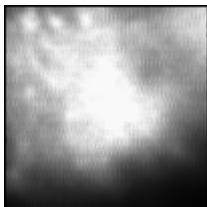
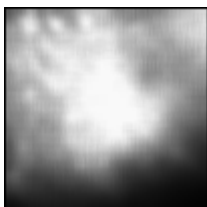
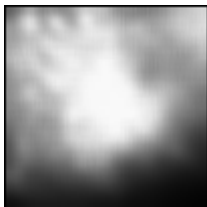
where  $a$ ,  $b$ ,  $c$ , and  $d$  are integers, and  $x = 0$  or  $1$ .

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 ( $x = 1$ ).

### Examples

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

<p style="text-align: center;"><b>Kernel #1</b></p> <pre style="text-align: center;"> 0  1  0 1  0  1 0  1  0                     </pre>	<p style="text-align: center;"><b>Filtered Image</b></p> 
<p style="text-align: center;"><b>Kernel #2</b></p> <pre style="text-align: center;"> 2  2  2 2  1  2 2  2  2                     </pre>	<p style="text-align: center;"><b>Filtered Image</b></p> 
<p style="text-align: center;"><b>Kernel #3</b></p> <pre style="text-align: center;"> 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                     </pre>	<p style="text-align: center;"><b>Filtered Image</b></p> 
<p style="text-align: center;"><b>Kernel #4</b></p> <pre style="text-align: center;"> 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  1  1  1  1  1  1                     </pre>	<p style="text-align: center;"><b>Filtered Image</b></p> 

## Predefined Smoothing Kernels

The following tables list the predefined smoothing kernels.

**Table 5-9.** 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 5-10.** 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 5-11.** 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
1 1 1 <b>0</b> 1 1 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 1 1

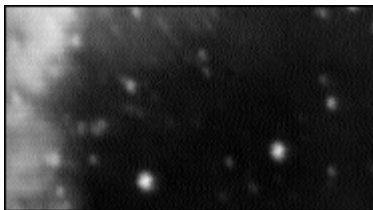
## Gaussian Filters

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.



## Example

This example uses 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:

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

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

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.

## Predefined Gaussian Kernels

The following tables list the predefined Gaussian kernels.

**Table 5-12.** 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 5-13.** 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 5-14.** 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

## Nonlinear Filters

A *nonlinear filter* replaces each pixel value with a nonlinear function of its surrounding pixels. Like the convolution filters, the nonlinear filters operate on a neighborhood. The following notations describe the behavior of the nonlinear spatial filters.

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):

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

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

### 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**

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

**Kernel #2**

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

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

### Nonlinear Sobel Filter

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**

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
**Kernel #2**

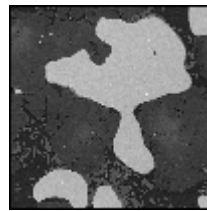
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

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

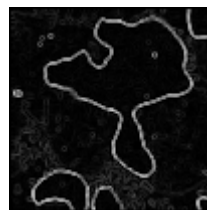
$$P_{(i,j)} = \max[|P_{(i+1,j-1)} - P_{(i-1,j-1)} + 2P_{(i+1,j)} - 2P_{(i-1,j)} + P_{(i+1,j+1)} - P_{(i-1,j+1)}|, |P_{(i-1,j+1)} - P_{(i-1,j-1)} + 2P_{(i,j+1)} - 2P_{(i,j-1)} + P_{(i+1,j+1)} - P_{(i+1,j-1)}|]$$

## Example

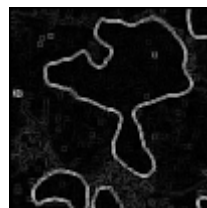
This example uses the following source image.



A nonlinear Prewitt filter produces the following image.



A nonlinear Sobel filter produces the following image.



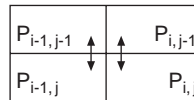
Both filters outline the contours of the objects. Because of the different convolution kernels they combine, the nonlinear Prewitt has the tendency to outline curved contours while the nonlinear Sobel extracts square contours. This difference is noticeable when observing the outlines of isolated pixels.

## Nonlinear Gradient Filter

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

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.

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

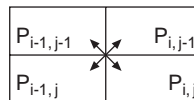


## Roberts Filter

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

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.

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

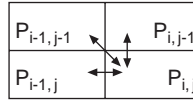


## Differentiation Filter

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

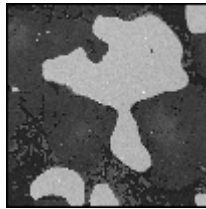
The new value of a pixel becomes the absolute value of its maximum deviation from its upper-left neighbors.

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

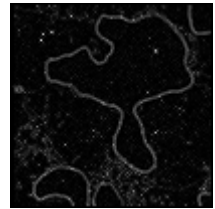


## Sigma Filter

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.



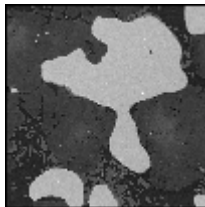
If  $P_{(i,j)} - M > S$ ,  
 then  $P_{(i,j)} = P_{(i,j)}$ ,  
 else  $P_{(i,j)} = M$ .



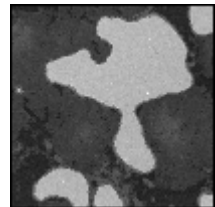
Given  $M$ , the mean value of  $P_{(i,j)}$  and its neighbors and  $S$ , their standard deviation, each pixel  $P_{(i,j)}$  is set to the mean value  $M$  if it falls inside the range  $[M - S, M + S]$ .

## Lowpass Filter

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.



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

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

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

You can implement the median filter by performing an  $N$ th order filter and setting the order to  $(f^2 - 1) / 2$ .

## 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.

Each pixel is assigned the  $N$ th value of its neighborhood,  $N$  being specified by the user.

$$P_{(i,j)} = N\text{th value in the series } [P_{(n,m)}]$$

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

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

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

=

13	10	9
12	4	8
5	5	6

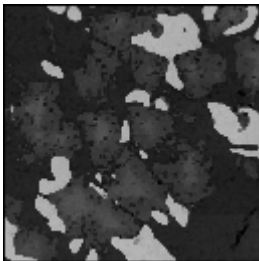
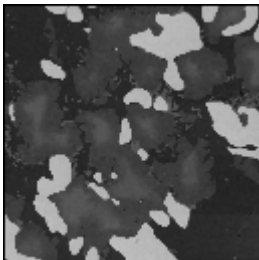
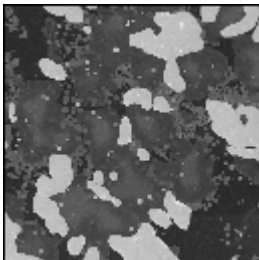
The following table shows the new output value of the central pixel for each  $N$ th order value:

Nth 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 ( $3^2 - 1$ ).

## Examples

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)	



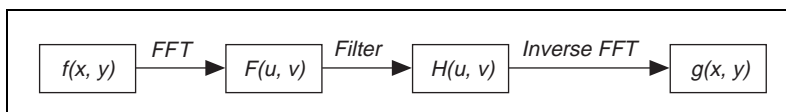
# Frequency Filtering

This chapter describes the frequency filters used in IMAQ Vision.

## Introduction to Frequency Filters

*Frequency filters* alter pixel values with respect to the periodicity and spatial distribution of the variations in light intensity in the image. Highpass frequency filters help isolate abruptly varying patterns that correspond to sharp edges, details, and noise. Lowpass frequency filters help emphasize gradually varying patterns such as objects and the background. Frequency filters do not apply directly to a spatial image, but to its frequency representation. Frequency representation 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.

The spatial frequencies seen in an FFT image can be filtered and the Inverse FFT then restores a spatial representation of the filtered FFT image.

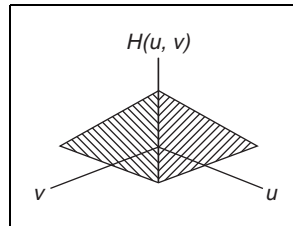


In an image, details and sharp edges are associated with high spatial frequencies. They are associated because details and sharp edges introduce significant gray-level variations over short distances. Gradually varying patterns are associated with low spatial frequencies.

For example, an image can have extraneous noise such as periodic stripes introduced during the digitization process. 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, yet the overall features remain.

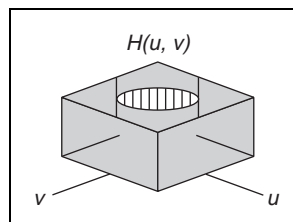
## Lowpass FFT Filters

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



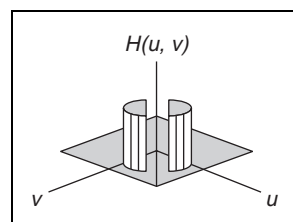
## Highpass FFT Filters

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.



## 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.



## Definition

---

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  $f(x, y)$  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). This algorithm requires that the resolution of the image be  $2^n \times 2^m$ . Notice that the values  $n$  and  $m$  can be different, which indicates that the image does not have to be square.

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

where  $NM$  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.

The relation between the sampling increments in the spatial domain  $(\Delta x, \Delta y)$  and the frequency domain  $(\Delta u, \Delta v)$  is

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

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

## 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

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

where  $|F(u, v)|$  is the magnitude and  $\phi(u, v)$  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  $\phi(u, v)$  is also called the phase angle and is equal to

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

Given an image with a resolution  $NM$  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:

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

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 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

$$[0, N\Delta u] \times [0, M\Delta v]$$

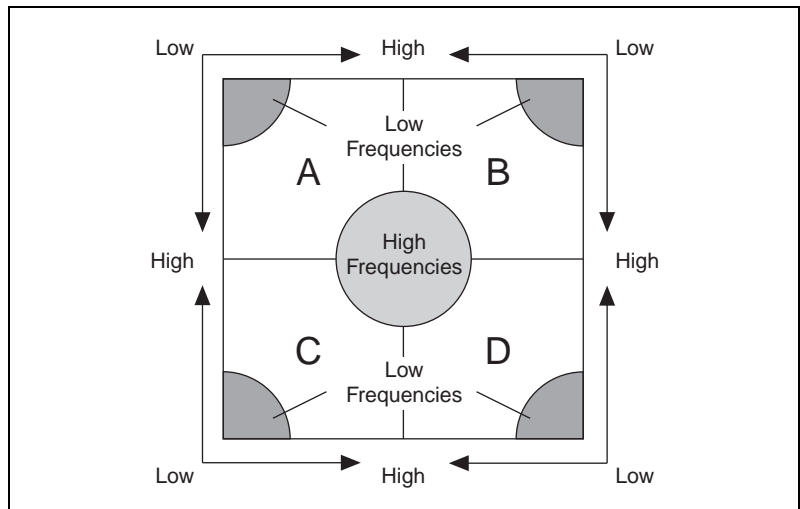
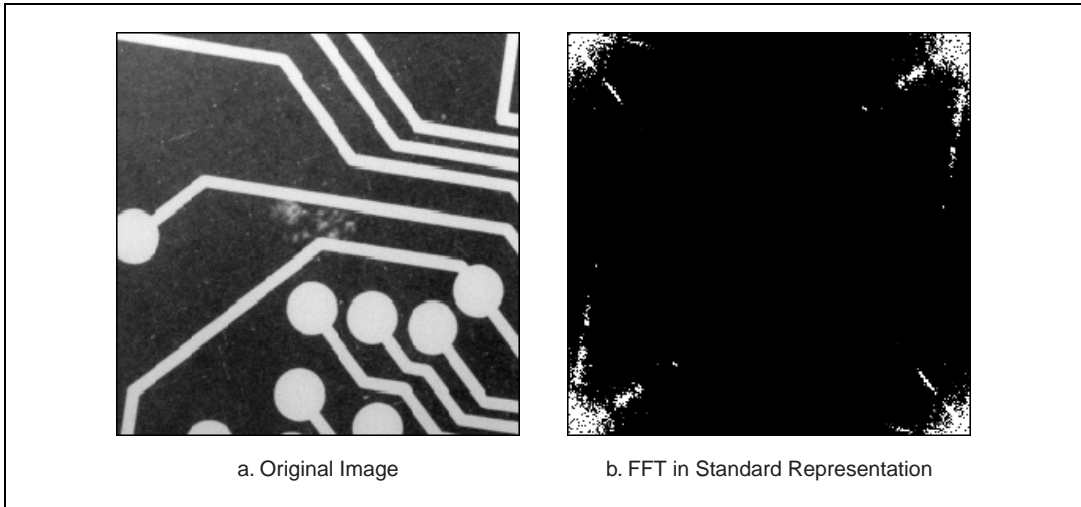


Figure 6-1a shows an image. Figure 6-1b shows the FFT of the same image in standard representation.

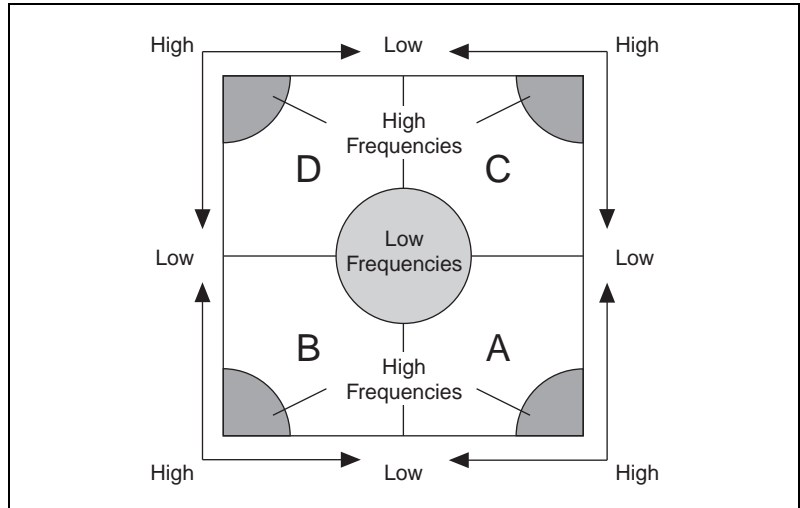


**Figure 6-1.** FFT of an Image in Standard Representation

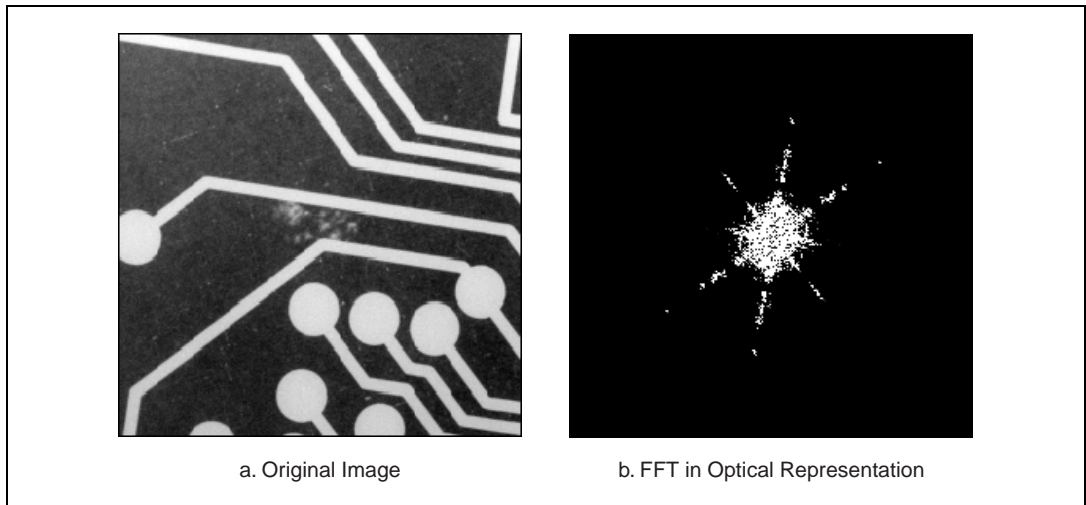
## Optical Representation

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

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



You can switch from standard representation to optical representation by permuting the *A*, *B*, *C*, and *D* quarters. Figure 6-2a shows the same original image as shown in Figure 6-1a. Figure 6-2b shows the FFT of the image in optical representation.



**Figure 6-2.** FFT of an Image in Optical Representation

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

# Frequency Filters

---

This section describes the frequency filters and includes information about lowpass and highpass attenuation and truncation.

## Lowpass Frequency 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.

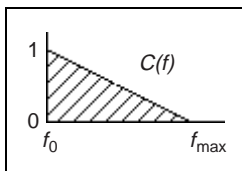
A lowpass frequency filter removes or attenuates spatial frequencies located outside a frequency range centered on the fundamental (or null) frequency.

## Lowpass Attenuation

*Lowpass attenuation* applies a linear attenuation to the full frequency range, decreasing 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  $C(f_{\max}) = 0$

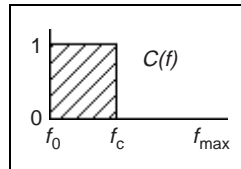




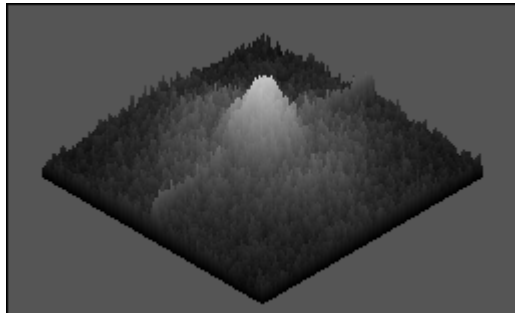
## Lowpass Truncation

*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  $f_c$ .

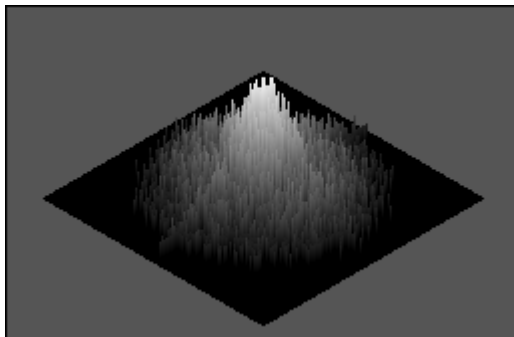
If  $f > f_c$   
 then  $C(f) = 0$   
 else  $C(f) = 1$



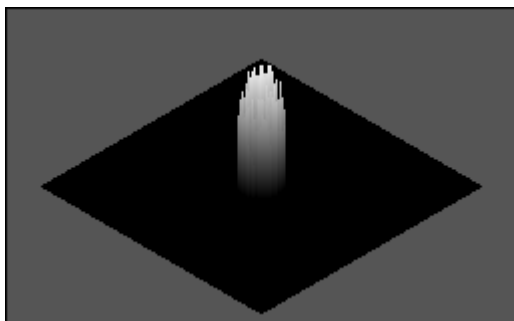
The following series of graphics illustrates the behavior of both types of lowpass filters. They give the 3D-view profile of the magnitude of the FFT. This example uses the following original FFT.



After lowpass attenuation, the magnitude of the central peak has been attenuated, 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 Frequency Filters

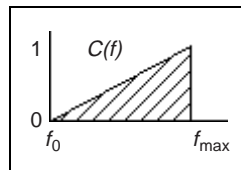
A *highpass frequency filter* attenuates or removes low frequencies present in the FFT plane. It has the effect of suppressing information related to slow variations of light intensities in the spatial image. In this case, the inverse FFT produces an image in which overall patterns are attenuated and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

## Highpass Attenuation

*Highpass attenuation* applies a linear attenuation to the full frequency range, decreasing 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.

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

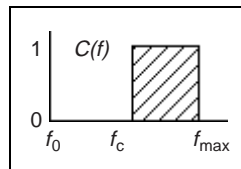
where  $C(f_0) = 1$  and  $C(f_{\max}) = 0$ .



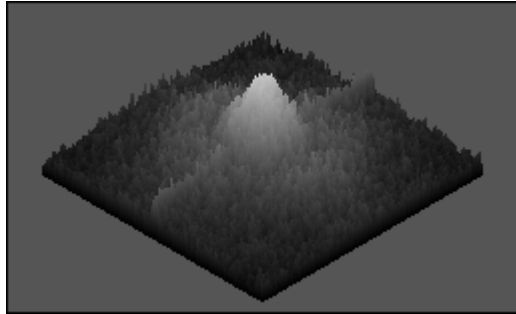
## Highpass Truncation

*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  $f_c$ .

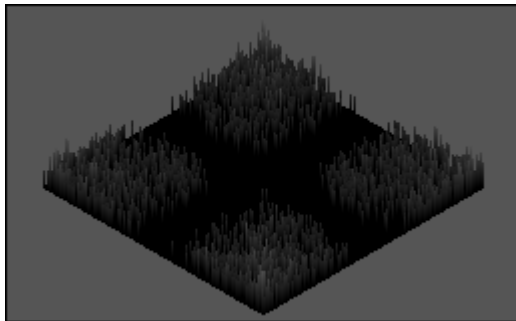
If  $f < f_c$   
 then  $C(f) = 0$   
 else  $C(f) = 1$



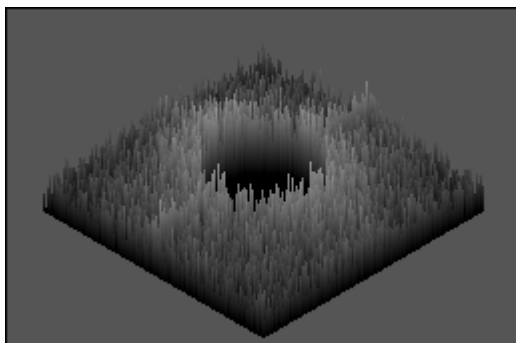
The following series of graphics illustrates the behavior of both types of highpass filters. They give the 3D-view profile of the magnitude of the FFT. This example uses the following original FFT image.



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.



---

# Morphology Analysis

This chapter provides an overview of morphology image analysis.

*Morphological transformations* extract and alter the structure of particles in an image. 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.

You can use the morphological transformations for expanding or reducing particles, filling holes, closing inclusions, smoothing borders, removing dendrites, and more. They fall into two categories:

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

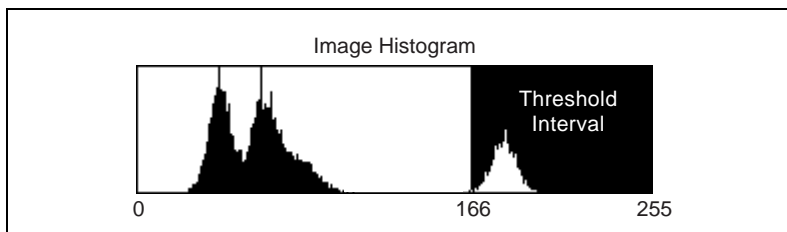
A *binary image* is an image that has been segmented into a particle region (pixels equal to 1) and a background region (pixels equal to 0). The *thresholding* process typically generates such an image.

---

## Thresholding

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.

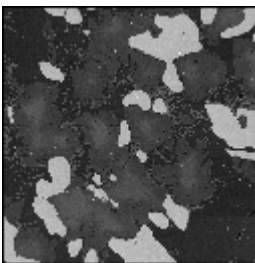
You can use this operation to extract areas that correspond to significant structures in an image and to focus the analysis on these areas.



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.

## 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 a particle and a background region occurs when the boundaries are not sharply demarcated. In such a case, the choice of a correct threshold becomes subjective. Therefore, it is highly recommended that images be enhanced before thresholding to outline where the correct borders lie. Observing the intensity profile of a line crossing a boundary area also can be 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.

## Thresholding a Color 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.

## 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 the user 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 gray-scale image before applying an automatic threshold function (for example, 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

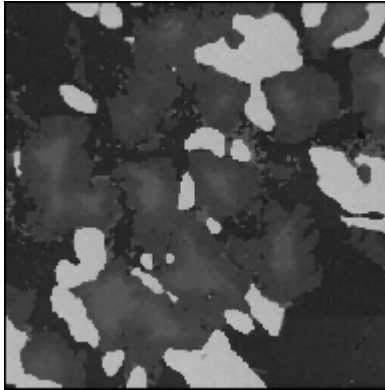
This technique sorts the histogram of the image within a discrete number of classes corresponding to the number of phases perceived in an image. The gray values are determined, and a *barycenter* is determined for each class. This process repeats until it obtains a value that represents the center of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

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

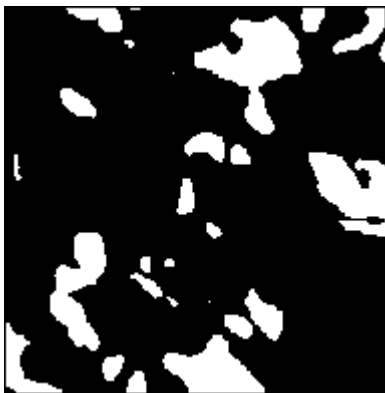
## Example of Clustering

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.

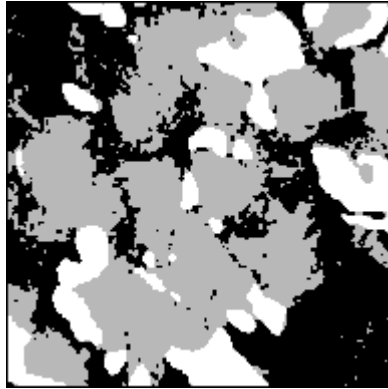


Clustering in two phases produces the following image.





Clustering in three phases produces the following image.



## Entropy

Based on a classical image analysis technique, this method is best suited for detecting particles that are present in minuscule proportions on the image. For example, this function would be suitable for defect 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 (an overexposed image is better processed than an underexposed image). 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 (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.

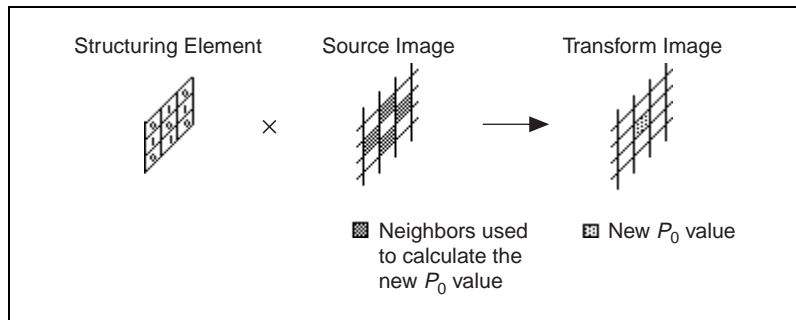
## Interclass Variance

Interclass variance is a classical statistic 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 has the tendency to underestimate the class of the smallest standard deviation if the two classes have a significant variation.

## Structuring Element

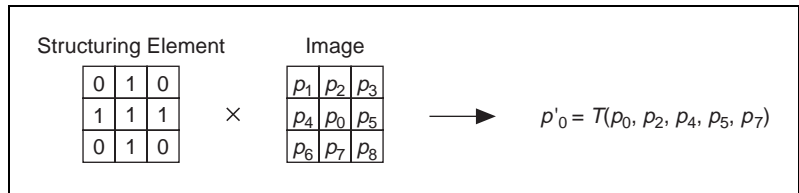
A *structuring element* is a binary mask used by most morphological transformations. You can use a structuring element to weigh the effect of these functions on the shape and the boundary of particles.

A morphological transformation using a structuring element alters a pixel  $P_0$  so that it becomes a function of its neighboring pixels. These neighboring pixels are masked by 1 when the structuring element is centered on  $P_0$ . Neighbors masked by 0 are discarded by the function.



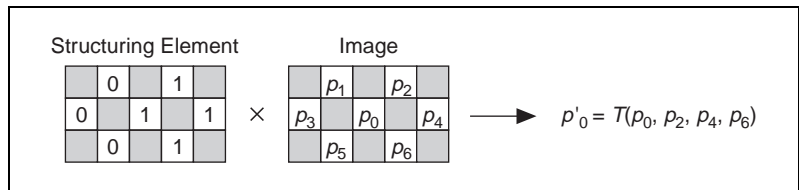
The structuring element is a binary mask (composed of 1 and 0 values). It is used to determine which neighbors of a pixel contribute to its new value. A structuring element can be defined in the case of a rectangular or hexagonal pixel frame, as shown in the following examples.

Figure 7-1 illustrates a morphological transformation using a structuring element. This example uses a  $3 \times 3$  image that has a rectangular frame.



**Figure 7-1.** Rectangular Frame, Neighborhood  $3 \times 3$

Figure 7-2 illustrates a morphological transformation using a structuring element for an image that has a hexagonal frame. This example uses a  $5 \times 3$  image.



**Figure 7-2.** Hexagonal Frame, Neighborhood  $5 \times 3$

The default configuration of the structuring element is a  $3 \times 3$  matrix with each coefficient set to 1:

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

## Primary Morphology Functions

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 derive from 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 Function

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

### Concept and Mathematics

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  $P_0 = 0$ .

## Dilation Function

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

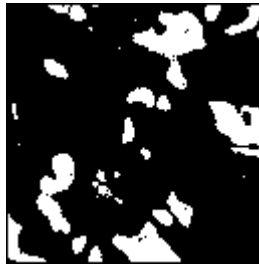
## Concept and Mathematics

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  $P_0 = 0$ .*

## Erosion and Dilation Examples

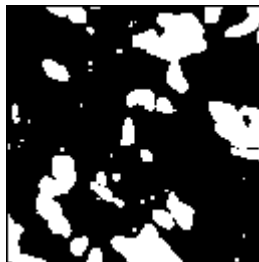
This example uses the following binary source image.



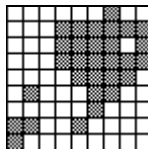
The erosion function produces the following image.



The dilation function produces the following image.


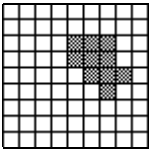

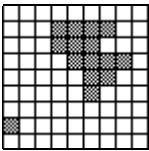


The next example uses the following source image. Gray cells indicate pixels equal to 1.


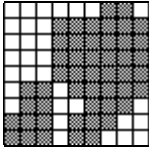

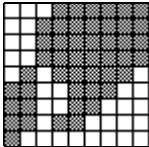


Tables 7-1 and show how the structuring element can be used to control the effect of an erosion or a dilation, respectively. The larger the structuring element, the more templates can be edited and the more selective the effect.

**Table 7-1.** How the Structure Element Affects Erosion

Structuring Element	After Erosion	Description
		<p>A pixel is cleared if it is equal to 1 and does not have its three upper-left neighbors equal to 1. The erosion truncates the upper-left borders of the particles.</p>
		<p>A pixel is cleared if it is equal to 1 and does not have its lower and right neighbors equal to 1. The erosion truncates the bottom and right borders of the particles but retains the corners.</p>

**Table 7-2.** 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 it has one of its three upper-left neighbors equal to 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 it has its lower or right neighbor equal to 1. The dilation expands the upper and left borders of the particles.

## Opening Function

The *opening function* is an erosion followed by a dilation. This function removes small particles and smooths boundaries. If  $I$  is an image,

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

This operation does not significantly alter the area and shape of particles because erosion and dilation are *dual transformations*. Borders removed by the erosion are restored by the dilation. However, small particles that vanish during the erosion do not reappear after the dilation.

## Closing Function

The *closing function* is a dilation followed by an erosion. It fills tiny holes and smooths boundaries. If  $I$  is an image,

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

This operation does not significantly alter the area and shape of particles because dilation and erosion are *morphological complements*. Borders expanded by the dilation function are reduced by the erosion function. However, tiny holes filled during the dilation do not reappear after the erosion.

## Opening and Closing Examples

The following series of graphics illustrate examples of openings and closings.



Original Image

```

1 1 1
1 1 1
1 1 1
    
```

Structuring Element



After Opening



After Closing

```

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
    
```

Structuring Element



After Opening

```

0 0 1 0 0
0 1 1 1 0
1 1 1 1 1
0 1 1 1 0
0 0 1 0 0
    
```

Structuring Element



After Closing

## 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. If  $I$  is an image,

$$internal\ edge(I) = I - erosion(I) = XOR(I, dilation(I))$$

## Outer Gradient Function

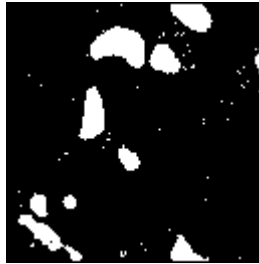
The *external edge* subtracts the source image from the dilated image of the source image. The remaining pixels correspond to the pixels added by the dilation. If  $I$  is an image,

$$external\ edge(I) = dilation(I) - I = XOR(I, dilation(I))$$

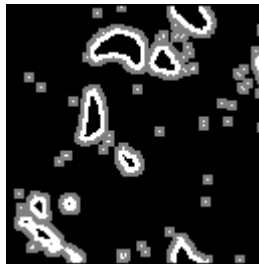


## External and Internal Edge Example

This example uses the following binary source image.



Extraction using a  $5 \times 5$  structuring element produces the following image. The superimposition of the internal edge is in white, and the external edge is in gray.



The thickness of the extracted contours depends on the size of the structuring element.

## Hit-Miss Function

You can use the *hit-miss function* to locate particular configurations of pixels. It extracts each pixel of an image that is placed in a neighborhood matching exactly 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 7-3, for strategies on using the hit-miss function.

## Concept and Mathematics

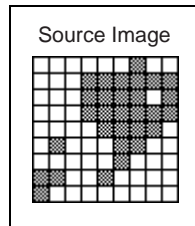
For a given pixel  $P_0$ , the structuring element is centered on  $P_0$ . The pixels masked by the structuring element are then referred to as  $P_i$ .

- 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  $P_0 = 0$ .

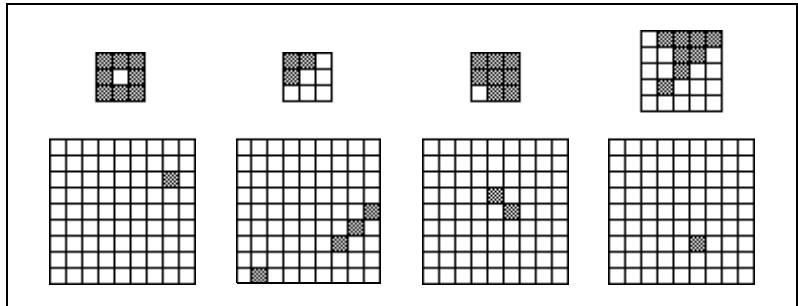
A hit-miss function using a structuring element with a central coefficient equal to 0 changes all pixels set to 1 in the source image to the value 0.

### Example 1

This example uses the following source image.



The following series of graphics shows the results of three hit-miss functions applied to the same source image. Each hit-miss function uses a different structuring element (specified above each transformed image). Gray cells indicate pixels equal to 1.

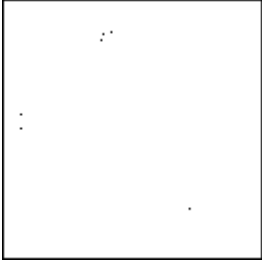
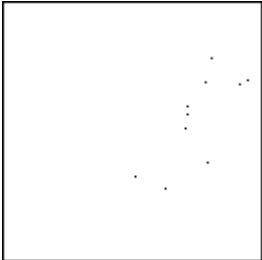
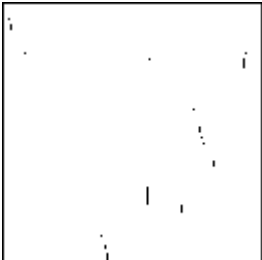


## Example 2

This example uses the following binary source image. Given this binary image, the hit-miss function can locate pixels surrounded by various patterns specified through the structuring element.



**Table 7-3.** Using the Hit-Miss Function

Strategy	Structuring Element	Resulting Image
<p>Use the hit-miss function to locate pixels isolated in a background.</p> <p>The structuring element presented on the right extracts all pixels equal to 1 that are surrounded by at least two layers of pixels equal to 0.</p>	<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>	
<p>Use the hit-miss function to locate single pixel holes in particles.</p> <p>The structuring element presented on the right extracts all pixels equal to 0 that are surrounded by at least one layer of pixels equal to 1.</p>	<pre> 1 1 1 1 0 1 1 1 1                     </pre>	
<p>Use the hit-miss function to locate pixels along a vertical left edge.</p> <p>The structuring element presented on the right extracts pixels surrounded by at least one layer of pixels equal to 1 to the left and pixels equal to 0 to the right.</p>	<pre> 1 1 0 1 1 0 1 1 0                     </pre>	

## Thinning Function

The *thinning function* eliminates pixels that are located in a neighborhood that matches a template specified by the structuring element. Depending on the configuration of the structuring element, thinning can be used 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.

If  $I$  is an image,

$$\text{thinning}(I) = I - \text{hit-miss}(I) = \text{XOR}(I, \text{hit-miss}(I))$$

This operation is not appropriate 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.

## Examples

This example uses the following binary source image.



This example uses the thinning function and the following structuring element:

```

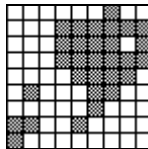
0 0 0
0 1 0
0 0 0

```

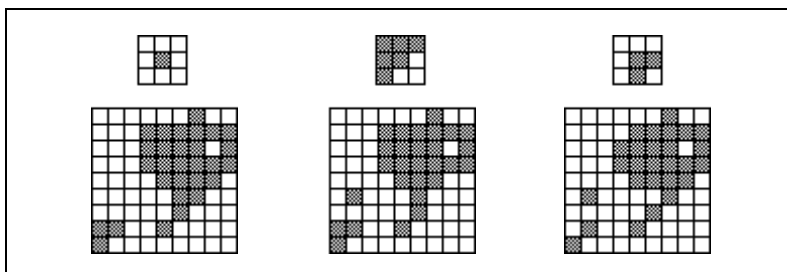
Thinning produces the following image. Single pixels in the background of this image have been removed.



The next example uses the following source image.



The following series of graphics shows the results of three thinnings applied to the source image. Each thinning uses a different structuring element (specified above each transformed image). Gray cells indicate pixels equal to 1.



## 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, thickening can be used to fill holes, smooth right angles along the edges of particles, and so forth. The larger the size of the structuring element, the more specific the template can be.

The thickening function extracts the union between a source image and its transformed image after a hit-miss function that uses the structuring element specified for the thickening. In binary terms, the operation adds a hit-miss transformation to a source image. If  $I$  is an image,

$$\text{thickening}(I) = I + \text{hit-miss}(I) = \text{OR}(I, \text{hit-miss}(I))$$

This operation is not appropriate when the central coefficient of the structuring element is equal to 1. In such case, 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.

## Examples

This example uses the following binary source image.



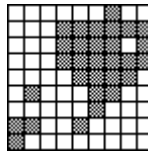
Thickening using the structuring element

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{array}$$

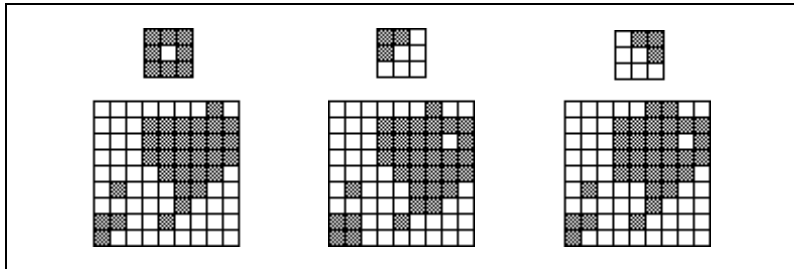
produces the following image. Single pixel holes are filled.



The next example uses the following source image.



The following series of graphics shows the results of three thickenings applied to the source image. Each thickening uses a different structuring element (specified on top of each transformed image). Gray cells indicate pixels equal to 1.



## 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 with respect 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 followed by another opening.

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

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 with respect 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 followed by another closing.

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

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  $I$ .

## Auto-Median Function

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

---

The advanced morphology functions are conditional combinations of fundamental transformations such as the binary erosion and dilation. They apply to binary images in which a threshold of 1 has been applied to particles and 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 might have been truncated during the digitization of the image, and their elimination might be useful 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 can identify particles using connectivity-4 or connectivity-8 criteria.

## Lowpass and Highpass Filters

The *lowpass filter* removes small particles with respect to their widths (specified by a parameter called *filter size*). For a given filter size  $N$ , the lowpass filter eliminates particles with widths less than or equal to  $(N - 1)$  pixels. These particles are those that would disappear after  $(N - 1)/2$  erosions.

The *highpass filter* removes large particles with respect to their widths (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 are those that would not disappear after  $(N/2 + 1)$  erosions.

Both the highpass and lowpass morphological filters use erosions to determine if a particle is to be removed. Therefore, they cannot discriminate particles with widths of  $2k$  pixels from particles with widths of  $2k - 1$  pixels. That is, a single erosion eliminates both particles that are 2 pixels wide and 1 pixel wide.

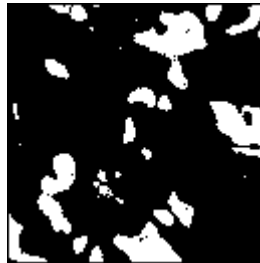
Table 7-4 shows the effect of lowpass and highpass filtering.

**Table 7-4.** Effect of Lowpass and Highpass Filtering

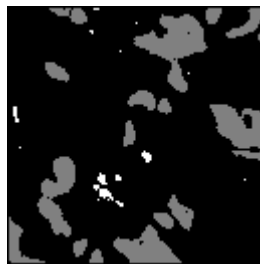
Filter Size ( $N$ )	Highpass Filter	Lowpass Filter
<b><math>N</math> is an even number</b> ( $N = 2k$ )	<ul style="list-style-type: none"> <li>Removes particles with a width greater than or equal to <math>2k</math></li> <li>Uses <math>k - 1</math> erosions</li> </ul>	<ul style="list-style-type: none"> <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</b> ( $N = 2k + 1$ )	<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>

## Lowpass and Highpass Example

This example uses the following binary source image.



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.



## Separation Function

The *separation function* breaks narrow isthmuses and separates particles that touch each other with respect to a user-specified filter size.

For example, after thresholding an image, two gray-level particles overlapping one another might appear as a single binary particle. A narrowing can be observed 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 at the same time to all particles that contain a narrowing shorter than  $N$  pixels.

For a given filter size  $N$ , the separation function segments particles having a narrowing shorter than or equal to  $(N - 1)$  pixels. These particles are those that are divided into two parts after  $(N - 1)/2$  erosions.

This operation uses erosions, labeling, and conditional dilations.

The above definition is true when  $N$  is an odd number but needs to be modified slightly when  $N$  is an even number. This modification is due to the use of erosions to determine if a narrowing has to 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. For example, one erosion breaks both a narrowing that is 2 pixels wide and a narrowing that is 1 pixel wide.

The precision of the separation is then limited to the elimination of constrictions having a width lesser than an even number of pixels:

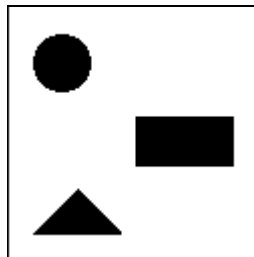
- If  $N$  is an even number ( $2k$ ), the separation breaks a narrowing with a width smaller than or equal to  $(2k - 2)$  pixels. It uses  $(k - 1)$  erosions.
- If  $N$  is an odd number ( $2k + 1$ ), the separation breaks a narrowing with a width smaller than or equal to  $2k$ . It uses  $k$  erosions.

## Skeleton Functions

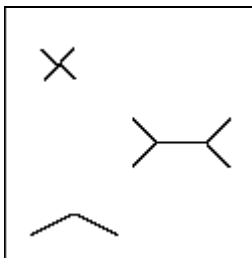
A *skeleton function* applies a succession of thinnings until the width of each particle becomes equal to 1 pixel. The skeleton functions are both time- and memory-consuming. They are based on conditional applications of thinnings and openings using various configurations of structuring elements.

### L-Skeleton Function

The *L-skeleton function* indicates the L-shaped structuring element skeleton function. For example, notice the following original image.

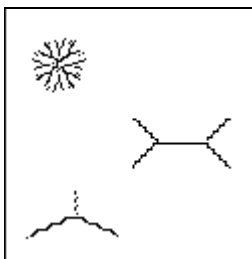


The L-skeleton function produces the following rectangle pixel frame image.



### M-Skeleton Function

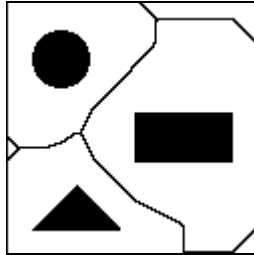
The *M-skeleton* (M-shaped structuring element) function extracts a skeleton with more dendrites or branches. Using the same original image as in the previous example, the M-skeleton function produces the following image.



### 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 same source image as in the previous example, the skiz function produces the following image (shown superimposed on the source image).



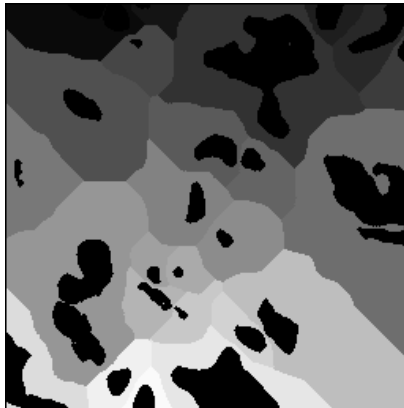
## Segmentation Function

The *segmentation function* is applied only to labeled images. It partitions an image into segments, each centered on a particle, such that they do not overlap each other or leave empty zones. This result is obtained by dilating particles until they touch one another.



**Note** The segmentation function is time-consuming. It is recommended that you reduce the image to its minimum significant size before selecting this function.

In the following image, binary particles (shown in black) are superimposed on top of the segments (shown in gray shades).



When applied to an image with binary particles, the transformed image turns entirely red because it is entirely composed of pixels set to 1.

## Comparisons Between Segmentation and Skiz Functions

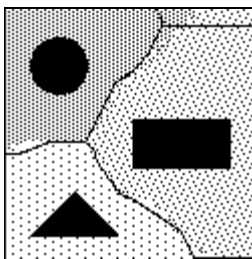
The segmentation function extracts segments that each contain one particle and represent the area in which this particle can be moved without intercepting another particle (assuming that all particles move at the same speed).

The edges of these segments give a representation of the external skeletons of the particles. As opposed to the skiz function, segmentation does not involve median distances.

Segments are obtained by 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. On the other hand, if you consider the inside of closed skiz lines as segments, you might produce more segments than particles originally present in the image. Notice the upper-right region in the following example.

The following image shows:

- Original particles in black
- Segments in dotted patterns
- Skiz lines



## Distance Function

The *distance function* assigns to each pixel a gray-level value 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.

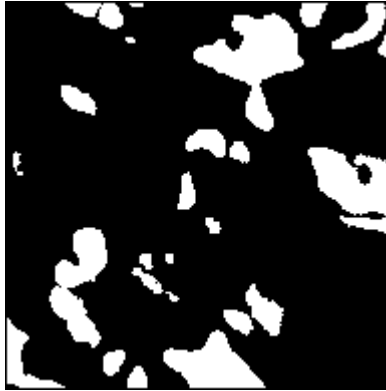


## Danielsson Function

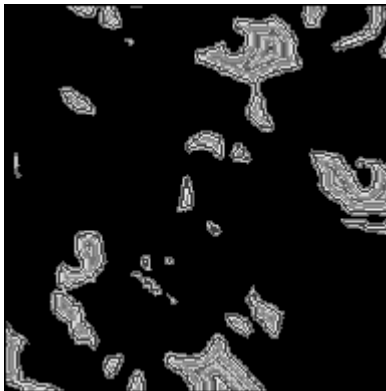
The *Danielsson function* also creates a distance map but is a more accurate algorithm than the classical distance function. Use the Danielsson function instead of the distance function when possible.

### Example

This example uses the following source threshold image.



The image is sequentially processed with a lowpass filter, hole filling, and the Danielsson function. The Danielsson function produces the following distance map image.



It is useful to view this final image with a binary palette. In this case, each level corresponds to a different color. The user easily can determine the relation of a set of pixels to the border of a particle. The first layer (the layer

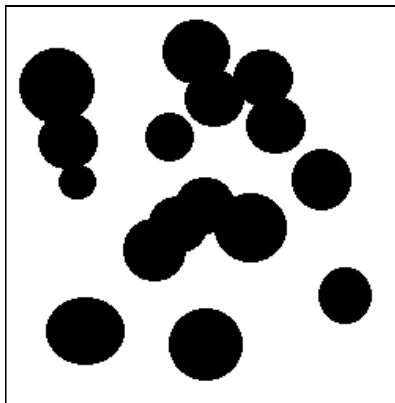
that forms the border) is red. The second layer (the layer closest to the border) is green, the third layer is blue, and so forth.

## Circle Function

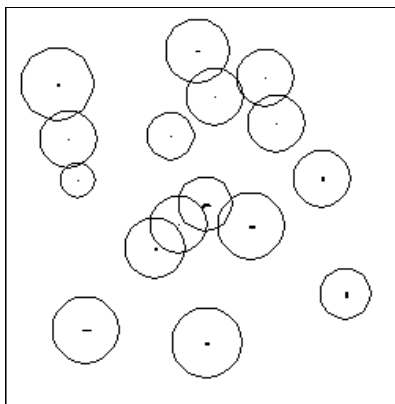
The *circle* function enables the user to separate overlapping circular particles. The circle function uses the Danielsson coefficient to reconstitute the form of an particle, provided that the particles are essentially circular. The particles are treated as a set of overlapping discs that is then separated into separate discs. Therefore, it is possible to trace circles corresponding to each particle.

### Example

This example uses the following source image.



The circle function produces the following processed image.



## 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 needed in cases in which the sample particle is cut because of the acquisition process.

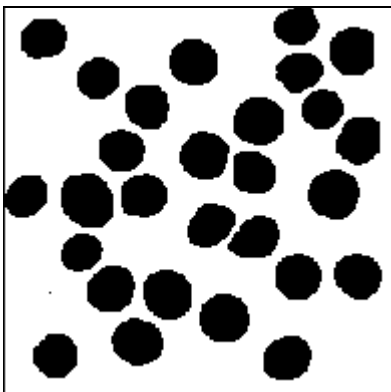
The convex function calculates a convex envelope around the perimeter of each particle, effectively closing the particle. The image to be treated must be both binary and labeled.

### Example

This example uses the following original binary labeled image.



The convex function produces the following image.



# Gray-Level Morphology

---

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 template.

### Concept and Mathematics

Each pixel  $P_0$  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 Chapter 5, *Spatial Filtering*, for more information.

## Dilation Function

The *gray-level dilation* has the same effect as the gray-level erosion because dilating bright regions is equivalent to eroding dark regions. 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.

### Concept and Mathematics

Each pixel  $P_0$  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 Chapter 5, *Spatial Filtering*, for more information.

## Erosion and Dilation Examples

This example uses the following source image.

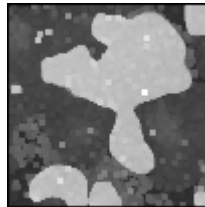
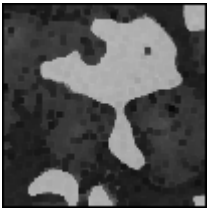
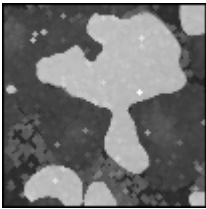
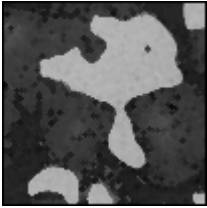
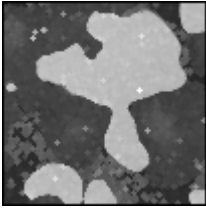


Table 7-5 provides example structuring elements and the corresponding eroded and dilated images.

**Table 7-5.** Erosion and Dilation Examples

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

$$opening(I) = dilation(erosion(I))$$

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

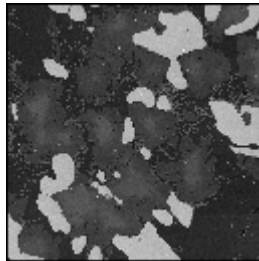
The gray-level *closing function* consists of a gray-level dilation followed by a gray-level erosion. It removes dark spots isolated in bright regions and smooths boundaries. The effects of the function are moderated by the configuration of the structuring element.

$$closing(I) = erosion(dilation(I))$$

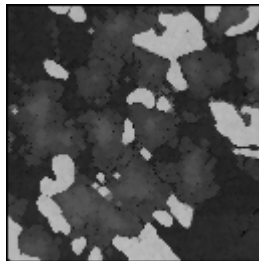
This operation does not significantly alter the area and shape of particles because dilation and erosion are morphological opposites. Bright borders expanded by the dilation are reduced by the erosion. However, small dark particles that vanish during the dilation do not reappear after the erosion.

## Opening and Closing Examples

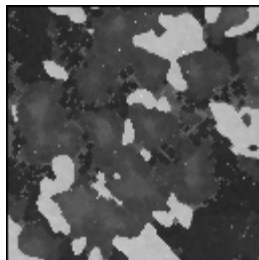
This example uses the following source image.



The opening function produces the following image.



Consecutive applications of an opening or closing command always give the same results. A closing function produces the following image.



## 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.

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 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.

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 Function

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

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  $I$ .

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

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  $I$ .

---

# Quantitative Analysis

This chapter provides an overview of quantitative image analysis. The *quantitative analysis* of an image consists of extracting information related to pixel intensities and particle measurements. Before starting this analysis, you must calibrate the image spatial dimensions and intensity scale to obtain measurements expressed in real units.

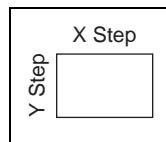
## Spatial Calibration

---

*Spatial calibration* consists of correlating the dimensions of a pixel with physical dimensions. The latter can be defined by three parameters:

- **X Step**—the horizontal length of a pixel
- **Y Step**—the vertical length of a pixel
- **Unit**—the selected unit of distance

The area of a pixel is then equal to  $(X\ Step \times Y\ Step)Unit^2$



If a pixel represents a square area, then

$$X\ Step = Y\ Step = Sampling\ Step$$

The spatial calibration of an image can be performed using two methods:

- *Pixel calibration*, or editing the dimensions of a single pixel
- *Distance calibration*, or editing the length of a line selected in the image

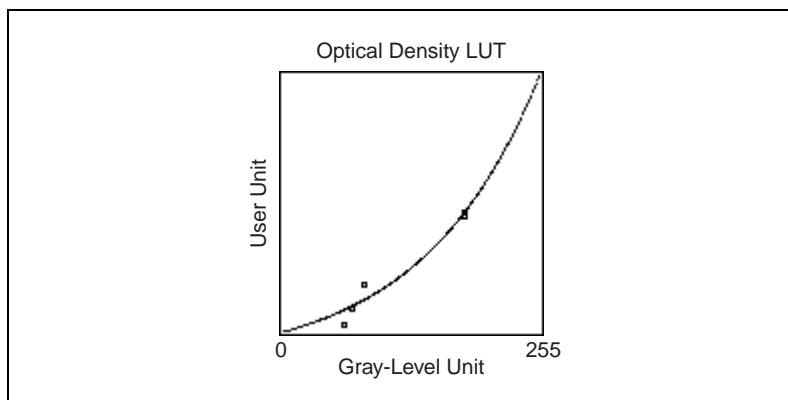
## Intensity Calibration

*Intensity calibration* consists of correlating the gray-scale values to user-defined quantities such as optical densities or concentrations.

The intensity calibration of an image is performed in two steps:

- Selection of sample points in an image and calibration of their gray-level value
- Selection of a curve-fitting algorithm to calibrate the entire gray scale

The following example uses an 8-bit image, or 256 gray levels.



## Digital Particles

In digital images, particles can be defined by three criteria: *intensity threshold*, *connectivity*, and *area threshold*.

### 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). Then other pixels are considered part of the background.

The *threshold interval* is defined by the two parameters **[Lower Threshold, Upper Threshold]**. In the case of binary particles the Threshold Interval is [1, 1].

## Connectivity

After the pixels belonging to a specified intensity threshold are identified, they are grouped into particles. This process introduces the notion of adjacent pixels or connectivity.

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  $P_8$ .

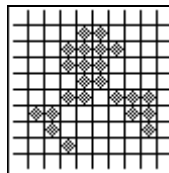
$$\begin{bmatrix} P_8 & P_1 & P_2 \\ P_7 & P_0 & P_3 \\ P_6 & P_5 & P_4 \end{bmatrix}$$

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-8

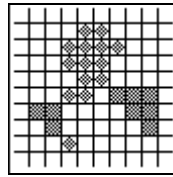
A pixel belongs to a particle if it is at a distance  $D$  or  $\sqrt{2}D$  from another pixel in the particle. In other words, two pixels are considered as part of the same particle if they are horizontally, vertically, or diagonally adjacent. In the following image, the particle count equals 1.



### Connectivity-4

A pixel belongs to a particle if it is at a distance  $D$  from another pixel in the particle. In other words, two pixels are considered as 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 the following image, the particle count equals 4.



## Area Threshold

A size criteria can be specified to detect only particles falling in a given area range.

The area threshold is defined by the two parameters, **Minimum Area** and **Maximum Area**.

### Examples

Table 8-1 shows an example of thresholding by area. In the following example, 1 pixel = 1 square inch.

**Table 8-1.** Example of Thresholding by Area

Particles to Detect	Lower Threshold	Upper Threshold	Minimum Area	Maximum Area
Black particles (gray level 0) at least 1 sq-in.	0	0	1	65536
White particles (gray level 255) bigger than 500 sq-in.	255	255	501	65536
Labeled particles placed in a black background and ranging from 200 to 1000 sq-in.	1	255	200	1000
Light-gray particles belonging to the gray-level range [190, 200] and smaller than 3000 sq-in.	190	200	1	3000



**Note** The most straightforward way to isolate particles is to use the threshold function and convert them to binary particles. This method offers the advantage of clearly showing the particles while the threshold interval remains constant and equal to [1, 1].

# Particle Measurements

---

A digital particle can be characterized by a set of morphological and intensity parameters described in the *Areas*, *Lengths*, *Coordinates*, *Chords and Axes*, *Shape Equivalence*, *Shape Features*, *Densitometry*, and *Diverse Measurements* sections.

## Areas

This section describes the following area parameters:

- **Number of pixels**—Area in number of pixels
- **Particle area**—Area expressed in real units (based on image spatial calibration)
- **Scanned area**—Area of the entire image expressed in real units
- **Ratio**—Ratio of the particle area to the entire image area
- **Number of holes**—Number of holes within the particle
- **Holes' area**—Total area of the holes
- **Total area**—Area of the particle including its holes' area (equals **Particle area** + **Holes' area**)

## Number of Pixels

Number of pixels in a particle. This value gives the area of a particle, without holes, in pixel units.

## Particle Area

Area of a particle expressed in real units. 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

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

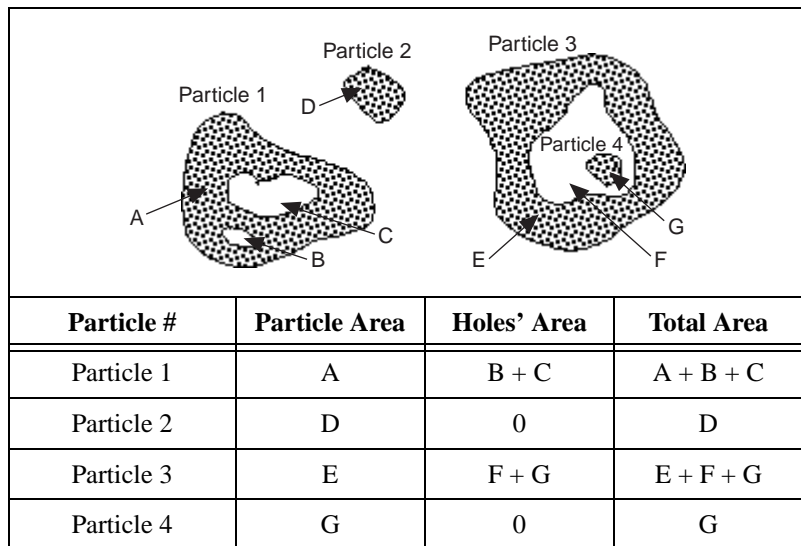
Total area of the holes within a particle.

## 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.



## Lengths

This section describes the following length parameters:

- **Particle perimeter**—Length of the outer contour
- **Holes' perimeter**—Sum of the perimeters of the holes within the particle

- **Breadth**—Distance between the left-most and right-most pixels in the particle
- **Height**—Distance between the upper-most and lower-most pixels in the particle

## Particle Perimeter

Length of the outer contour of a particle.

## Holes' Perimeter

Sum of the perimeters of the holes within a particle.

Holes' measurements can turn into 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  $\max(X_i) - \min(X_i)$ . It 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
- **Min X, Min Y**—Upper-left corner of the smallest horizontal rectangle containing the particle
- **Max X, Max Y**—Lower-right corner of the smallest horizontal rectangle containing the particle
- **Max chord X and Y**—Left-most point along the longest horizontal chord



## Center of Mass X and Center of Mass 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 horizontal segments in a particle.



**Note**  $G$  can be located outside a particle if the particle has a non-convex shape.

## Min(X, Y) and Max(X, 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  $(\min X, \min Y)$  and  $(\max X, \max Y)$  such that

$$\min X = \min(X_i)$$

$$\min Y = \min(Y_i)$$

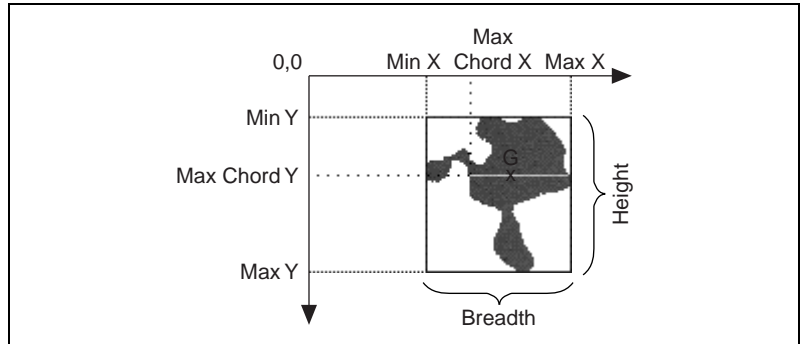
$$\max X = \max(X_i)$$

$$\max Y = \max(Y_i)$$

where  $X_i$  and  $Y_i$  are the coordinates of the pixels  $P_i$  in a particle.

## Max Chord X and Max Chord 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
- **Mean chord X**—Mean length of horizontal segments
- **Mean chord Y**—Mean length of vertical segments
- **Max intercept**—Length of the longest segment (in all possible directions)
- **Mean intercept perpendicular**—Mean length of the segments perpendicular to the max intercept
- **Particle orientation**—Orientation in degrees with respect to the horizontal axis

### Max Chord Length

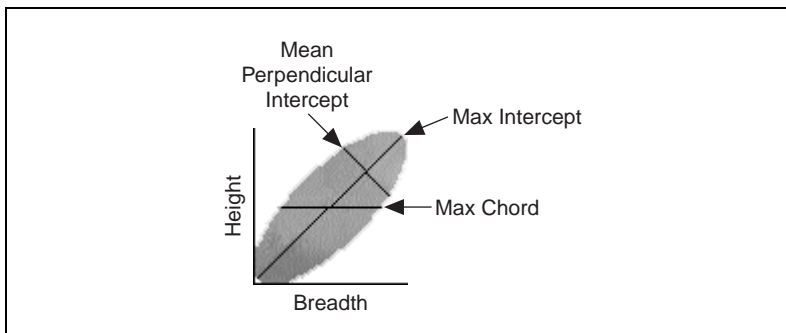
Length of the longest horizontal chord in a particle.

### 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 a particle (in all possible directions of projection).

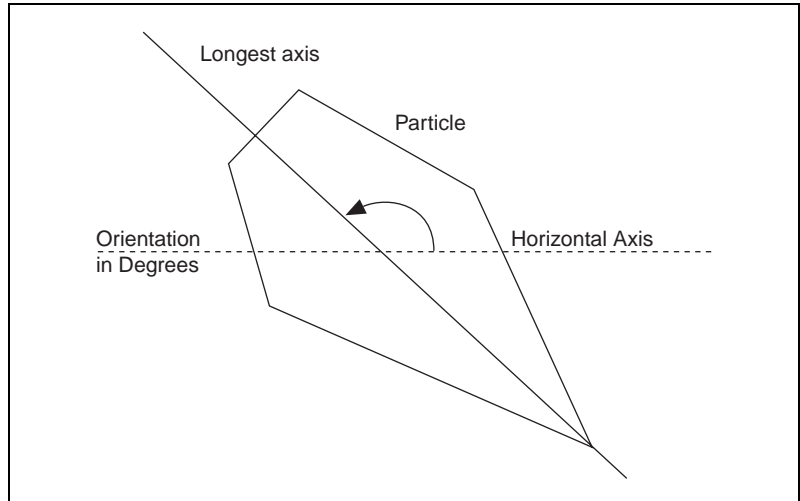
## Mean Intercept Perpendicular

Mean length of the segments in a particle perpendicular to the max intercept.

$$\text{Mean intercept perpendicular} = \frac{\text{particle area}}{\text{max intercept}}$$

## Particle Orientation

The angle of the longest axis with respect to the horizontal axis. The value can be between  $0^\circ$  and  $180^\circ$ . Notice that this value does not give information regarding the symmetry of the particle. Therefore, an angle of  $190^\circ$  is considered the same as  $10^\circ$ .



## Shape Equivalence

This section describes the following shape-equivalence parameters:

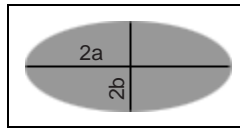
- **Equivalent ellipse minor axis**—Minor axis of the ellipse that has the same area as the particle and a major axis equal to half its max intercept
- **Ellipse major axis**—Major axis of the ellipse that has the same area and same perimeter as the particle
- **Ellipse minor axis**—Minor axis of the ellipse that has the same area and same perimeter as the particle
- **Ellipse Ratio**—Ratio of the major axis of the equivalent ellipse to its minor axis
- **Rectangle big side**—Big side of the rectangle that has the same area and same perimeter as the particle
- **Rectangle small side**—Small side of the rectangle that has the same area and same perimeter as the particle
- **Rectangle ratio**—Ratio of the big side of the equivalent rectangle to its small side

## Equivalent Ellipse Minor Axis

The *equivalent ellipse minor axis* is the 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:

$$\begin{aligned} \text{particle area} &= \pi ab \text{ and} \\ \text{max intercept} &= 2a \end{aligned}$$



The equivalent ellipse minor axis is defined as

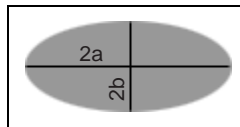
$$2b = \frac{4 \times \text{particle area}}{\pi \times \text{max intercept}}$$

## Ellipse Major Axis

The *ellipse major axis* is the 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:

$$\begin{aligned} \text{Area} &= \pi ab \\ \text{Perimeter} &= \pi \sqrt{2(a^2 + b^2)} \end{aligned}$$



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

The *ellipse minor axis* is the 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

The *ellipse ratio* is the 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.

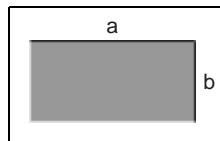
## Rectangle Big Side

*Rectangle big side* is the 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

*Rectangle small side* is the 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

*Rectangle ratio* is the 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**—Moments of inertia  $I_{xx}$ ,  $I_{yy}$ ,  $I_{xy}$  with respect to the center of gravity
- **Elongation factor**—Ratio of the longest segment within the particle to the mean length of the perpendicular segments
- **Compactness factor**—Ratio of the particle area to the area of the smallest rectangle containing the particle
- **Heywood circularity factor**—Ratio of the particle perimeter to the perimeter of the circle with the same area
- **Hydraulic radius**—Ratio of the particle area to its perimeter
- **Waddel disk diameter**—Diameter of the disk with the same area as the particle

## Moments of Inertia $I_{xx}$ , $I_{yy}$ , $I_{xy}$

The *moments of inertia* give a representation of the distribution of the pixels in a particle with respect to its center of gravity.

## Elongation Factor

The *elongation factor* is the 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

The *compactness factor* is the 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

The *Heywood circularity factor* is the 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

The *hydraulic radius* is the 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.

### 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 8-2 describes derived measurements in IMAQ Vision.

**Table 8-2.** 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>	$(\Sigma y^2) - A \times M_y^2$

Table 8-2. Derived Measurements (Continued)

Symbol	Derived Measurement	Primary Measurement
$I_{xy}$	<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>	$A/P_x$
$S_{\max}$	<b>Max Intercept</b>	$(C_{\max}/l)^2 \times \max(h, l) + d(1 - (C_{\max}/l)^2)$
$C$	<b>Mean Perpendicular Intercept</b>	$A/S_{\max}$
$A_{2b}$	<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>,  else <math>d^\circ =</math>  <math display="block">\frac{90}{\text{atan}(2 \times I_{xy}/(I_{xx} - I_{yy}))}</math></p> <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>
$E_{2a}$	<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>	$1/4 (p + t')$ where $t' = \sqrt{p^2 - 16A}$
$r_c$	<b>Rectangle Small Side</b>	$1/4 (p - t')$ where $t' = \sqrt{p^2 - 16A}$
$R_{Rr}$	<b>Rectangle Ratio</b>	$R_c/r_c$
$F_e$	<b>Elongation Factor</b>	$S_{\max}/C\pi$
$F_c$	<b>Compactness Factor</b>	$A/(h \times l)$

**Table 8-2.** Derived Measurements (Continued)

Symbol	Derived Measurement	Primary Measurement
$F_H$	<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>	$2\sqrt{\frac{A}{\pi}}$

## 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
- **Sum Gray Value**—Sum of the intensities in the particle expressed 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
- **Minimum User Value**—Minimum intensity value in user units
- **Maximum User Value**—Maximum intensity value in user units
- **Sum User Value**—Sum of the intensities in the particle expressed in user units
- **Mean User Value**—Mean intensity value in the particle expressed in user units
- **Standard Deviation (Unit)**—Standard deviation of the intensity values in user units

## 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

---

# Pattern Matching

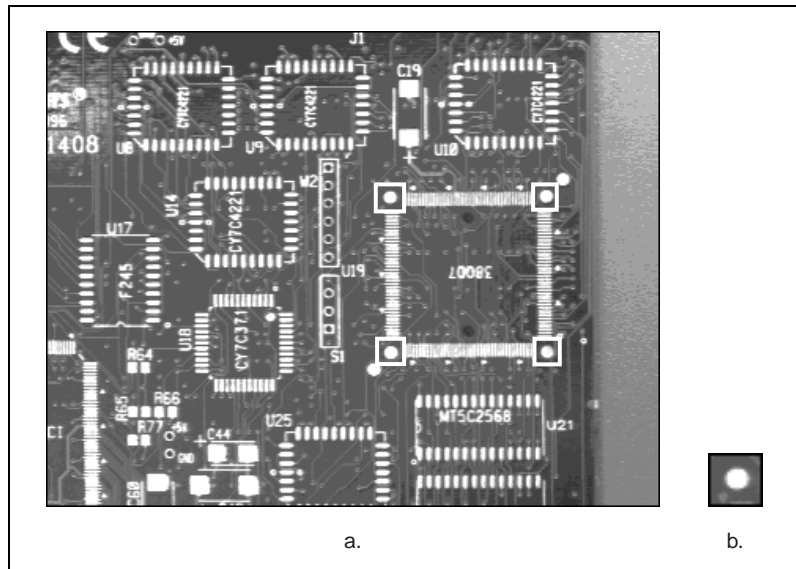
This chapter describes pattern matching and the pattern matching tools in IMAQ Vision.

## Introduction to Pattern Matching

---

You can use *pattern matching* to locate quickly 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 close the model matches the pattern found.

Pattern matching is the key to many applications. Pattern matching can provide 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 9-1a shows part of a circuit board. Figure 9-1b shows a common fiducial used in Printed Circuit Board (PCB) inspections or chip pick-and-place applications.



**Figure 9-1.** 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 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. In real-time applications, search speed is critically important.

Whether inspecting the leads on a quad pack or inspecting an antilock-brake sensor, the key to success for a machine vision application is finding and locating features. Simply put, a software strategy for finding fiducials is the starting point to many applications.

## Pattern Matching Applications

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. You 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 then the component is rejected. You can use gauging inline with the manufacturing process or offline. A sample of components determines the quality of a batch of manufactured components.
- 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, and the feature that may be present in an image. A feature is defined as a specific pattern of pixels in an image.

## 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 being inspected can change due to factors such as orientation of the part, scale changes, and lighting changes. The pattern matching tool should maintain its ability to locate the reference patterns despite these changes. The following are commonly occurring situations under which the pattern matching tool should give 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 and 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
- The size ratio of the pattern to the template image
- Multiple instances of the pattern in the image (if applicable)

Figure 9-2a shows a template image, or pattern. Figure 9-2b shows the template shifted in the image. Figure 9-2c shows the template rotated in the image. Figure 9-2d shows the template scaled in the image. Figures 9-2b to 9-2d illustrate multiple occurrences of the template.

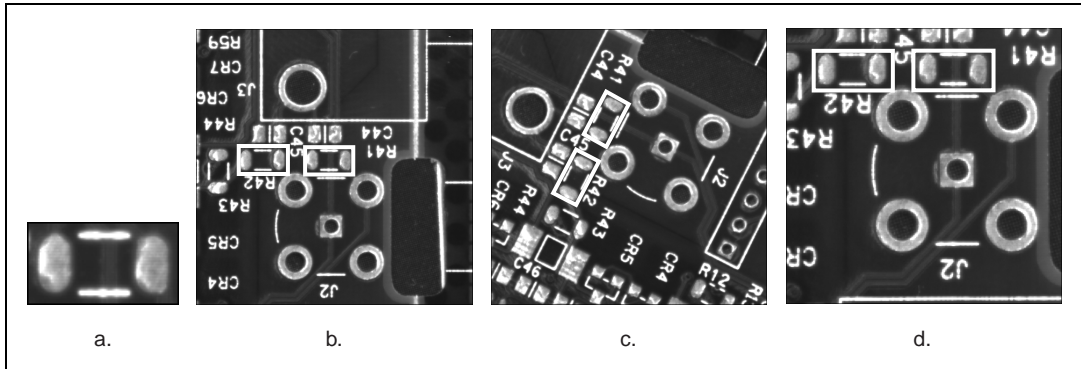


Figure 9-2. Pattern Orientation and Multiple Instances

## 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 9-3 illustrates typical conditions under which pattern matching works correctly. Figure 9-3a shows the original template image. Figure 9-3b shows the same pattern under bright light. Figure 9-3c shows the pattern under poor lighting.

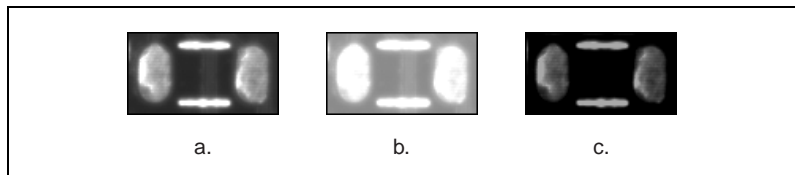


Figure 9-3. Examples of Lightning Conditions



## 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 9-4 illustrates typical blurring and noise conditions under which pattern matching works correctly. Figure 9-4a shows the original template image. Figure 9-4b shows changes on the image caused by blurring. Figure 9-4c shows changes on the image caused by noise.

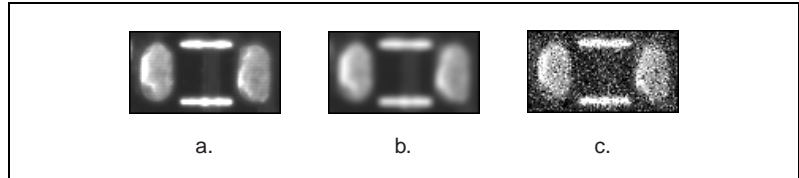


Figure 9-4. Examples of Blur and Noise

## Traditional Pattern Matching Techniques

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. 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 9-5 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  $f$ . At the borders of the image, correlation values are not accurate.

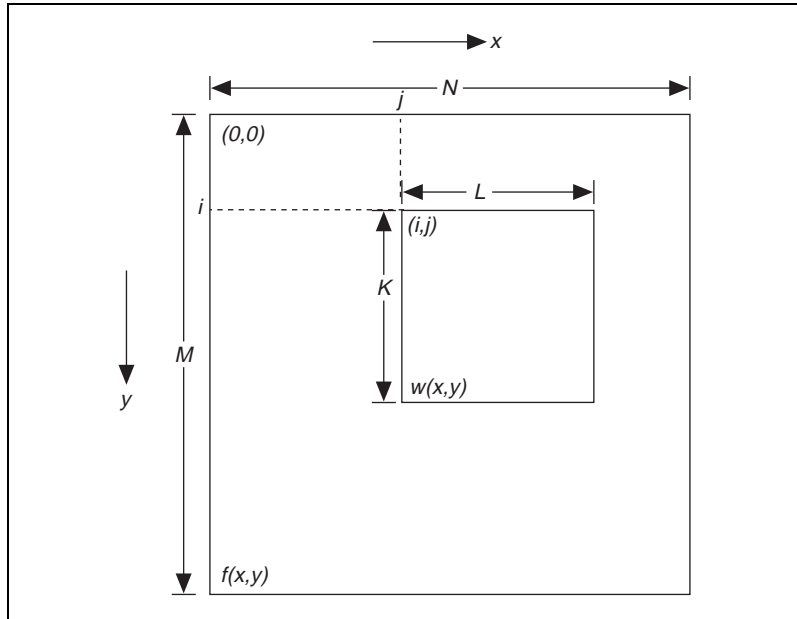


Figure 9-5. The Correlation Procedure

Basic correlation is very sensitive to amplitude changes in the image (i.e., changes in 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  $w$ .

Because the underlying mechanism for correlation is based on a series of multiplication operations, the correlation process is time consuming. With new technologies such as MMX, multiplications can be done in parallel, and the overall computation time can be reduced considerably. 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 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 as long as 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  $5^\circ$  to  $10^\circ$ . 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 will require 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

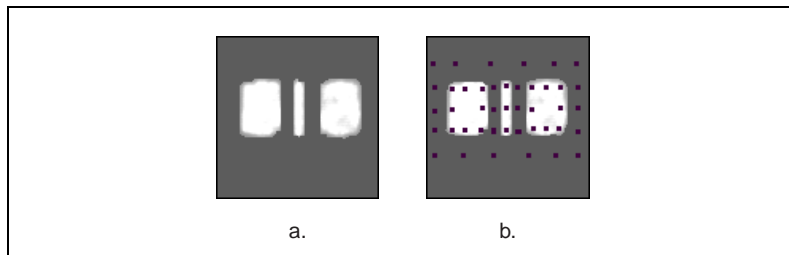
---

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, and hence speed up the searching process. Also, by extracting useful information from a template and removing the redundant and noisy information, the search is more accurate.

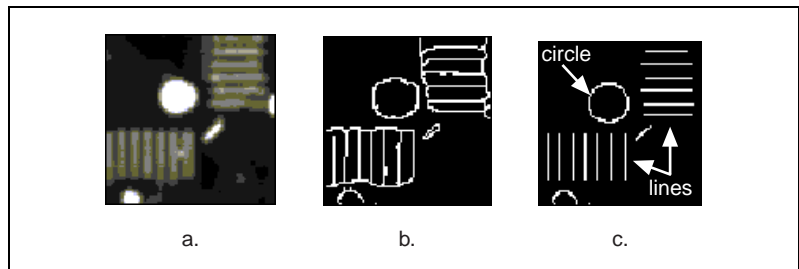
One new pattern matching technique uses non-uniform sampling. Most images contain redundant information. Using all the information in the image to match patterns is time-intensive and inaccurate. By sampling the image and extracting a few number of points that represent overall content of the image, the speed and accuracy of the pattern matching tool can be improved. Figure 9-6 shows an example of good sampling techniques for pattern matching. Figure 9-6a shows the original template, or reference image. The black dots on Figure 9-6b represent the points on the image that are used to represent the template.



**Figure 9-6.** 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.

The edge image can be further processed to extract higher level geometric information about the image, such as the number of straight lines or circles present in the image. Pattern matching then reduces to the matching of edge and/or geometric information between the template and the image. Figure 9-7 illustrates the importance of edges and geometric modeling. Figure 9-7a shows the reference pattern, Figure 9-7b shows edge information in the pattern, and Figure 9-7c shows the higher-level geometric interpretation of edges in the form of geometric objects, such as circles and lines.



**Figure 9-7.** 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. For 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 locate accurately objects in conditions where they vary in size ( $\pm 5\%$ ) and orientation (between  $0^\circ$  and  $360^\circ$ ) and when their appearance is degraded.

## Using Pattern Matching

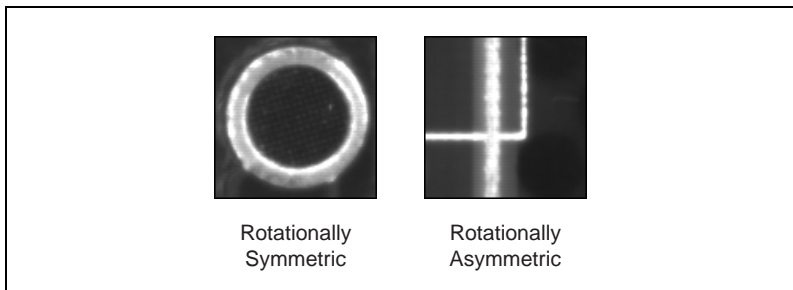
Follow these generalized steps to find features in an image using pattern matching:

1. Define a reference or fiducial pattern in the form of a template image.
2. Use the reference pattern to train the pattern matching tool.
3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.
4. Set the tolerances and parameters to specify how the tool operates at run time.
5. Test the search tool on test images.
6. Use a ranking method to verify results.

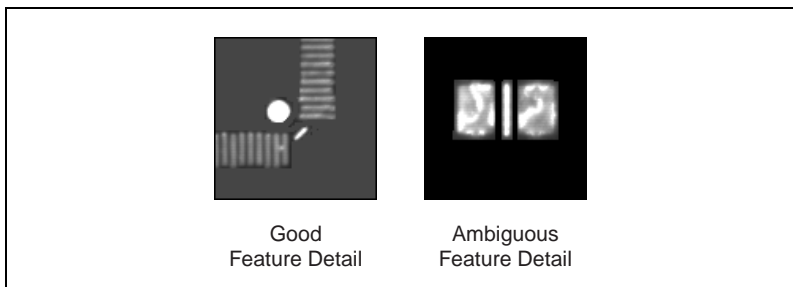
## Defining and Creating Good Template Images

The selection of a good template image plays a critical part in obtaining good results with the pattern matching tool. Because the template image represents the pattern that you want to find, make sure that all the important and unique characteristics of the pattern are well defined in the image. Several factors are critical in creating a template image. These critical factors include the following:

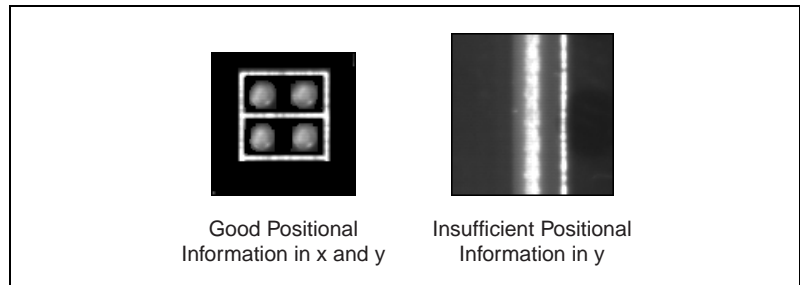
- **Symmetry**—A rotationally symmetric template is less sensitive to changes in rotation than one that is rotationally asymmetric.



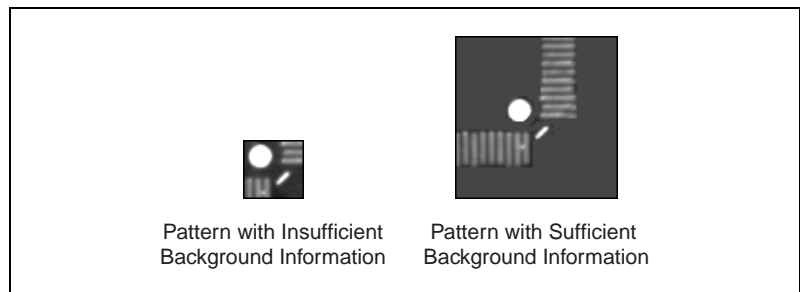
- **Feature detail**—A template with relatively coarse features is less sensitive to variations in size and rotation than a model with fine features. However, the model must contain enough detail to identify it.



- **Positional information**—A template with strong edges in both the x and y directions is easier to locate.



- **Background information**—Unique background information in a template improves search performance and accuracy.



## Using the Reference Pattern to Train the Pattern Matching Tool

After you have created a good template image, the pattern matching tool has to learn the important features of the template. The learning process will depend on the type of matching that you expect to perform. If you do not expect the instance of the template in the image to rotate or change its size, then the pattern matching tool has to learn only those features from the template that are necessary for shift-invariant matching. However, if you want to match the template at any orientation, the learning process must consider the possibility of arbitrary orientations.

Typically, the training or learning process is time intensive because the algorithm attempts to find the optimum features of the template for the particular matching process. However, you can train the pattern matching tool offline.

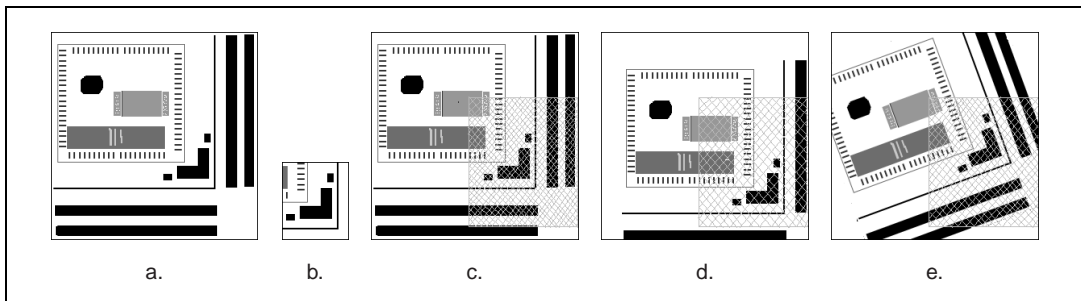
## Defining a Search Area

Two equally important factors define the success of a pattern matching tool: accuracy and speed. You can define a search area to reduce ambiguity in the search process. For example, if your image has multiple instances of a pattern and only one of them is required for the inspection task, the presence of additional instances of the pattern can produce incorrect results. To avoid this, you can reduce the search area so that only the desired pattern lies within the search area.

The time required to locate a pattern in an image depends on both the template size and the search area. The search time is directly proportional to the search area. By reducing the search area, you can reduce the required search time.

In many inspection applications, you have general information about the location of the fiducial. You should use this information to define a search area. For example, in a typical semiconductor inspection application, each wafer being tested may not be placed in the same location with the same orientation. The location of the wafer in various images can move and rotate within a known range of values, as illustrated in Figure 9-8.

Figure 9-8a shows a schematic of an image containing a semiconductor wafer that needs inspection. Figure 9-8b shows the template used to locate the wafer in the image. If you know before the matching process begins that the wafer can shift or rotate in the image within a fixed range, then you can limit the search for the template to a small region of the image. This small region is specified by the shaded portion in Figures 9-8c, 9-8d, and 9-8e.



**Figure 9-8.** Selecting a Search Area



## Setting Matching Parameters and Tolerances

Every pattern matching algorithm makes assumptions about the images and pattern matching parameters used in machine vision applications. These assumptions work for a high percentage of the applications. However, there may be cases where these assumptions do not hold, and the default values of the parameters used in the algorithm are not optimal. In such cases you need the flexibility to modify these pattern matching parameters. Knowing your particular application and the images you want to acquire is useful in selecting the pattern matching parameters. You have the flexibility to adjust the pattern matching parameters for IMAQ Vision.

The following are two parameters in the IMAQ Vision pattern matching tool and how they influence pattern matching:

- **Minimum Contrast**—You can use the minimum contrast parameter to potentially increase the pattern matching tool's speed. The pattern matching tool ignores all image regions where contrast values fall beneath a set minimum contrast value. If the image you are searching has high contrast but contains some low contrast regions, you can set a high minimum contrast value. By using a high minimum contrast value, you exclude all areas in the image with low contrast, significantly reducing the region in which the pattern matching tool must search. If the image you are searching has low contrast throughout, set a low minimum contrast parameter to ensure that the pattern matching tool looks for the template in all regions of the image.
- **Rotation Angle Ranges**—For matching objects that may vary in orientation (in cases where the template pattern can be at any orientation in the image), the pattern matching algorithm by default assumes that the pattern can take any orientation between 0° to 360°. If in your application you know that the pattern rotation is restricted to a certain range (for example, between -15° to 15°), you can provide this information to the pattern matching algorithm. This will speed up your search time because the pattern matching tool will not have to look for the pattern at all angles.

## Test the Search Tool on Test Images

To determine if your selected template or reference pattern is appropriate for your machine vision application, test the template on a few test images. The test images that you select should contain most variations you expect to see in images during a regular run of your machine vision application. If the pattern matching tool locates the reference pattern in all the test

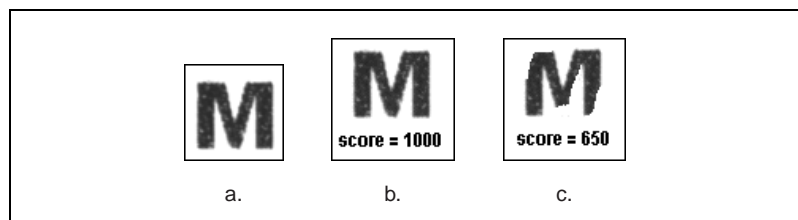
images, you have selected a good template. Otherwise, refine the current template, or select a better template, and repeat the training and testing steps.

## Using a Ranking Method to Verify Results

Interpreting results returned by the pattern matching tool will depend on your application. For typical alignment applications, such as finding a fiducial on a wafer, the most important information is the position and location of the best match returned by the tool.

In inspection applications, such as optical character verification (OCV), the score of the best match is more useful. The score of a match returned by the pattern matching tool is an indicator of the closeness between the original pattern and the match found in the image. A high score indicates a very close match, while a low score indicates a poor match. The score can be used as a gauge to determine whether a printed character is acceptable.

Figure 9-9 shows how you can use the pattern matching score for an OCV application. You can use OCV applications to verify the quality of printed characters. Pharmaceutical applications use the OCV technique to verify the quality of lot and date code printing on packaging. Figure 9-9a shows the reference image of the letter to inspect. Figure 9-9b shows an image containing a good representation of the letter M. The score returned by the pattern matching function in this case is 1000, indicating a perfect match. Figure 9-9c shows a corrupted version of the letter M. The score of 650 indicates that the matched pattern in this image is not an exact match to the template. Based on the scores, you would accept the image in Figure 9-9b and reject the image in Figure 9-9c for an optical character verification application.



**Figure 9-9.** Optical Character Verification

## Other Searching Techniques in IMAQ Vision

---

Pattern matching is the first choice for a search tool. However, IMAQ Vision provides other search techniques that may be more useful in specific scenarios—binary shape matching, blob analysis, and edge detection and alignment.

### Binary Shape Matching

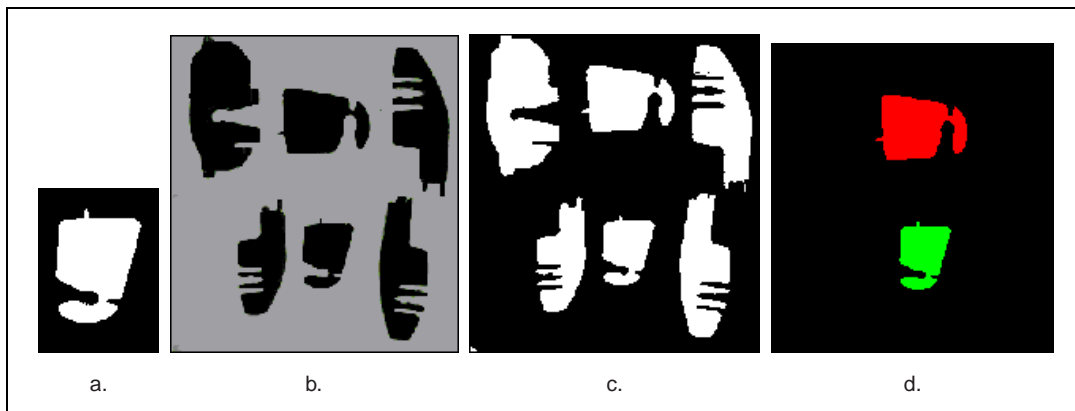
If you are searching for a feature that has a known shape but unknown size and orientation, and the image can be thresholded, consider using binary shape matching.

Thresholding is the process of converting a grayscale image into a black and white (binary) image. Binary shape matching is performed by extracting parameters from a template object that represent the object's shape 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. Binary shape matching is useful in robot guidance applications where, for example, a robot arm has to sort parts into groups of similar shapes. A shape matching function, such as the IMAQ ShapeMatch Tool in IMAQ Vision, 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 9-10 shows how binary shape matching is used to sort windshield wiper parts into different shapes. Figure 9-10a shows the shape template. Figure 9-10b shows the original grayscale image of different windshield parts. Figure 9-10c shows the binary (or thresholded) version of the original image. Figure 9-10d 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 9-10.** Using Shape Matching to Search for Windshield Wiper Parts

## Blob Analysis

If you have a binary image, but shape matching is not a viable option, consider using blob analysis. Using blob analysis, you can detect and analyze any two-dimensional shape in an image.

A typical blob analysis process scans through the entire image and detects all the particles, or blobs (**binary large objects**), 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.

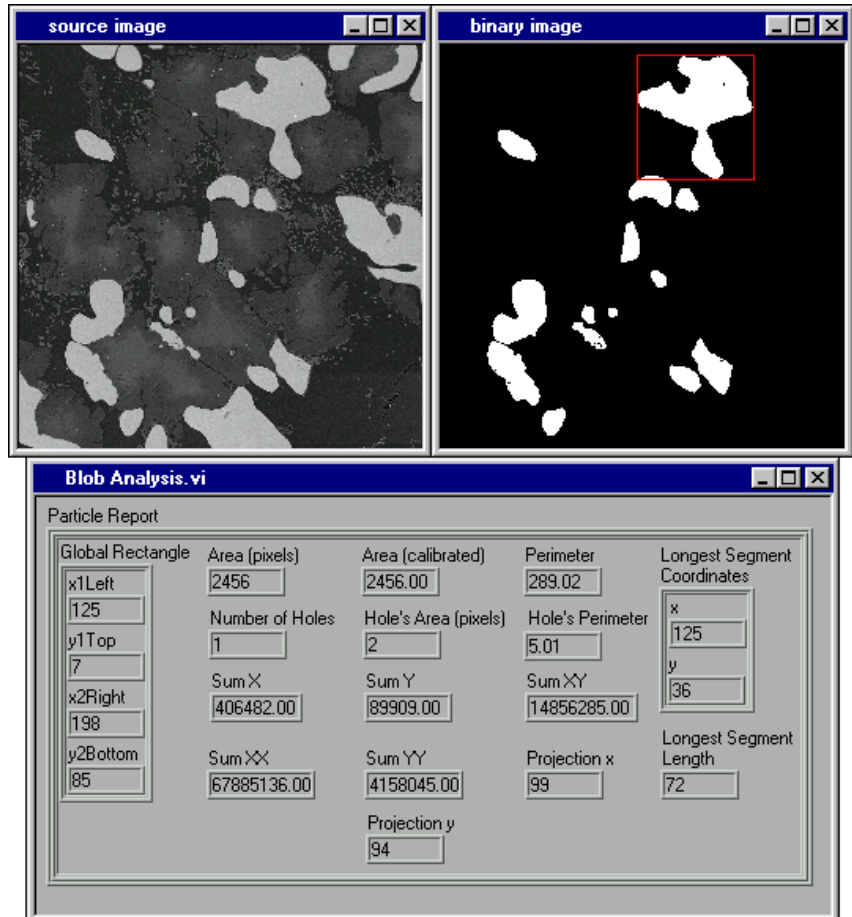
Multiple parameters such as perimeter, angle, area, and center of mass can be used to identify and classify these blobs. Using multiple parameters can be faster and more effective than pattern matching in many applications.

Also, by using different sets of parameters you can uniquely identify the feature in the image. For example, you could use the size of the template blob as a criterion to remove all blobs that do not match it within some tolerance. Then you can do 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 (ratio of the area of the blob to the area of the smallest rectangle that encloses the blob).

For many applications, blob analysis provides you with a powerful and flexible way to define your own searching or matching parameters.

Figure 9-11 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.



**Figure 9-11.** List of Parameters Extracted from a Particle in a Binary Image

To use shape matching or blob analysis, you must be able to threshold the image and clearly identify the object in the binary image. Clearly identifying an object in a binary image is not practical in many searching applications because of factors like low image contrast and highly reflective parts.

## Search Using Edge Detection and Alignment Tools

If performance or speed is the key issue, consider using a combination of line profiles and edge detection tools to locate your feature. Use line profiles to identify the outer boundaries or edges of an object under inspection. This technique makes assumptions about the orientation and shape of the object under inspection to improve performance.

For example, in a rectangular object, you can use two line profiles along the x-axis of the component and one line profile along the y-axis to identify quickly the orientation and location of the object under inspection. Three points on the object are necessary to calculate correctly the orientation for a square or rectangular part.

Once you know the orientation, you can create a local coordinate system for the part. You can use the local coordinate system to offset to any location to inspect for missing features. For performance gains, offset and search for a feature in a small region of interest using grayscale pattern matching or blob analysis.

Figure 9-12 shows a disk inspection application where this technique is used to determine if the label “HD” has been printed on the disk correctly. First, set up a mechanism that locates and determines the orientation of the disk and then locates the label on the disk, as follows:

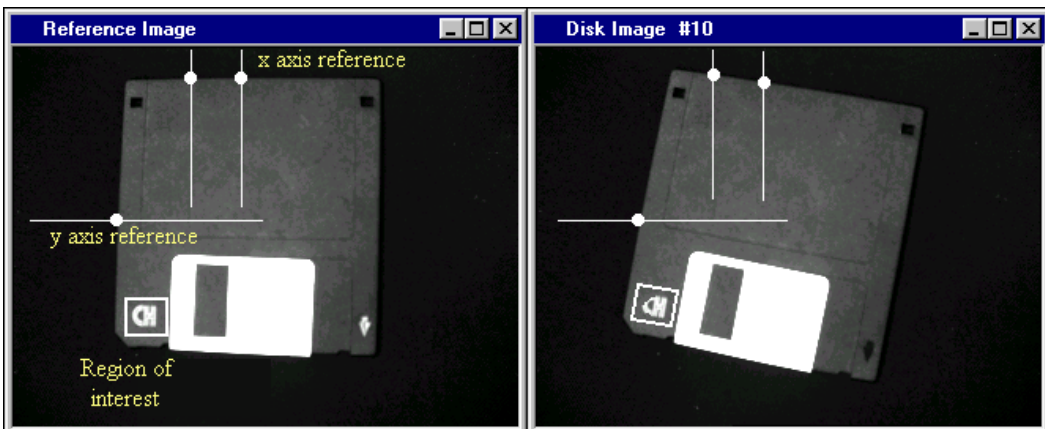
1. Build a coordinate system for the disk relative to the coordinate system of the image.
  - a. On a reference image, such as the one in Figure 9-12, obtain the intensity profiles along three lines: two vertical lines along the top edge of the disk and one horizontal line along the left edge of the disk.
  - b. Detect edge points along these line profiles. These edge locations provide you with two points along the x-axis and one along the y-axis of the disk’s coordinate system.
  - c. Using these three points, determine the origin and the orientation of disk’s coordinate system. Use this information as the initial reference point for the disk.
2. Define a region of interest (ROI) on the image that encloses the label to be inspected. This ROI is defined with respect to the disk’s coordinate system.

For every new image, the application determines the location and orientation of the disk in the image by finding the edge locations along the three line profiles. The application computes the edge positions of the disk

along the line profiles and calculates the rotation and translation of the disk in the image. The application uses the edge information to find the new origin and orientation of the disk.

The difference between the initial reference point and the new reference is the amount the disk has moved (translated and rotated) from the reference image to the image. The application uses the amount the disk has moved to move the ROI to its correct position (over the label) in the new image. In IMAQ Vision, this entire process uses only three functions: FindEdge, BuildCoordinateRef, and TransformROI.

After the application locates the label in each image, shape matching inspects the label. The application thresholds the area under the ROI to obtain the binary shape of the label and compares it to a template shape. If the label matches the template shape within some tolerance, the disk passes the inspection.



**Figure 9-12.** Using Edge Detection and Alignment Tools



---

# Technical Support Resources

This appendix describes the comprehensive resources available to you in the Technical Support section of the National Instruments Web site and provides technical support telephone numbers for you to use if you have trouble connecting to our Web site or if you do not have internet access.

## NI Web Support

---

To provide you with immediate answers and solutions 24 hours a day, 365 days a year, National Instruments maintains extensive online technical support resources. They are available to you at no cost, are updated daily, and can be found in the Technical Support section of our Web site at [www.natinst.com/support](http://www.natinst.com/support).

### Online Problem-Solving and Diagnostic Resources

- **KnowledgeBase**—A searchable database containing thousands of frequently asked questions (FAQs) and their corresponding answers or solutions, including special sections devoted to our newest products. The database is updated daily in response to new customer experiences and feedback.
- **Troubleshooting Wizards**—Step-by-step guides lead you through common problems and answer questions about our entire product line. Wizards include screen shots that illustrate the steps being described and provide detailed information ranging from simple getting started instructions to advanced topics.
- **Product Manuals**—A comprehensive, searchable library of the latest editions of National Instruments hardware and software product manuals.
- **Hardware Reference Database**—A searchable database containing brief hardware descriptions, mechanical drawings, and helpful images of jumper settings and connector pinouts.
- **Application Notes**—A library with more than 100 short papers addressing specific topics such as creating and calling DLLs, developing your own instrument driver software, and porting applications between platforms and operating systems.



## Software-Related Resources

- **Instrument Driver Network**—A library with hundreds of instrument drivers for control of standalone instruments via GPIB, VXI, or serial interfaces. You also can submit a request for a particular instrument driver if it does not already appear in the library.
- **Example Programs Database**—A database with numerous, non-shipping example programs for National Instruments programming environments. You can use them to complement the example programs that are already included with National Instruments products.
- **Software Library**—A library with updates and patches to application software, links to the latest versions of driver software for National Instruments hardware products, and utility routines.

## Worldwide Support

---

National Instruments has offices located around the globe. Many branch offices maintain a Web site to provide information on local services. You can access these Web sites from [www.natinst.com/worldwide](http://www.natinst.com/worldwide).

If you have trouble connecting to our Web site, please contact your local National Instruments office or the source from which you purchased your National Instruments product(s) to obtain support.

For telephone support in the United States, dial 512 795 8248. For telephone support outside the United States, contact your local branch office:

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20,  
Brazil 011 284 5011, 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, 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,  
Norway 32 27 73 00, Singapore 2265886, Spain (Madrid) 91 640 0085,  
Spain (Barcelona) 93 582 0251, Sweden 08 587 895 00,  
Switzerland 056 200 51 51, Taiwan 02 2377 1200,  
United Kingdom 01635 523545

# Glossary

---

Prefix	Meanings	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

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 The National Instruments internal image file format used for saving calibration information associated with an image and for saving complex images.

alignment The process by which a machine vision application determines the location, orientation, and scale of a part being inspected.

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.
asynchronous	Property of a function or operation that begins an operation and returns control to the program before the completion or termination of the operation.
auto-median function	A function that uses dual combinations of opening and closing operations to smooth the boundaries of objects.

## **B**

barycenter	The barycenter of a range of an image's grayscale values is the grayscale value representing the centroid of that range in the image histogram.
binary image	An image containing objects usually represented with a pixel intensity of 1 (or 255) and the background of 0.
binary morphology	Functions that perform morphological operations on a binary image.
blob	Binary large object. A particle, or object, present in a binary 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.
border function	Removes objects (or particles) in a binary image that touch the image border.

## **C**

caliper	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.
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.

CLUT	Color look-up 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.
complex images	Save 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 as neighbors.
convex function	Computes the convex regions of objects in a binary image.
convolution	<i>See</i> linear filter.
convolution kernel	Simple $3 \times 3$ , $5 \times 5$ , or $7 \times 7$ 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.

## D

Danielsson function	Similar to the distance functions, but with more accurate results.
densitometry	Determination of optical or photographic density.
density function	For each gray level in a linear histogram, it gives the number of pixels in the image that have the same gray level.
device	Plug-in data acquisition board that can contain multiple channels and conversion devices.
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.

## 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 hysteresis	The difference in threshold levels between a rising and a falling edge.
edge steepness	The number of pixels that correspond to the slope or transition area of an edge.
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	<i>See</i> histogram equalization.
erosion	Reduces the size of an object along its boundary and eliminates isolated points in the image.
exponential and gamma corrections	Expand the high gray-level information in an image while suppressing low gray-level information.
exponential function	Decreases the brightness and increases the contrast in bright regions of an image and decreases contrast in dark regions.

## 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.

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.

## G

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 point (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.
gray-level images	Images with monochrome information.
gray-level morphology	Functions that perform morphological operations on a gray-level image.

## H

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 truncations.
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.
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.
<b>I</b>	
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 file	A file containing image information and data.
image processing	Encompasses various processes and analysis functions that you can apply to an 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.
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 functions	Detects specific features in an image, including edges, peaks, and rotational shifts.
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 the 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 between known values when resampling an image or array of pixels.

## J

JPEG	Joint Photographic Experts Group. Image file format for storing 8-bit and color images with lossy compression.
------	--

## 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.
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 and vice versa. 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, 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.
L-skeleton function	Uses an L-shaped structuring element in the Skeleton function.
LUT	Look-up 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 look-up table.

**M**

machine vision application	An inspection or measurement application that uses images acquired from a 2D sensor (typically a CCD camera) to help with inspection or measurement.
mask	Isolates parts of an image for further processing.
mask FFT filter	Removes frequencies contained in a mask (range) specified by the user.
mask image	An image containing a value of 1 and values of 0. Pixels in the source image with a corresponding mask image value of 1 are processed, while the others are left unchanged.
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.
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.
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 mainly are used to delineate objects and prepare them for quantitative inspection analysis.
M-skeleton	Uses an M-shaped structuring element in the skeleton function.

**N**

neighbor	A pixel whose value affects nearby pixels' values when an image is processed. The neighbors of a pixel are usually defined by a kernel.
neighborhood operations	Operations on a point in an image that take into consideration the values of the pixels neighboring that point.

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 that favors horizontal and vertical edges in an image.
nonlinear Sobel filter	A highpass edge-extraction filter that favors horizontal and vertical edges in an image.
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 <i>N</i> th pixel value, where <i>N</i> is the order of the filter.

## O

opening	An erosion followed by a dilation. An opening removes small objects and smoothes 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 look-up table.
pattern matching	The technique used to locate quickly known reference patterns or fiducials in an image.
picture element	An element of a digital image.
pixel	Picture element.

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.
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 $3 \times 3$ 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.
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

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. An area of the image that is graphically selected from a window displaying the image. This area can be used focus further processing.
rotation-invariant matching	A pattern matching technique in which the reference pattern can be at any orientation in the test image.
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.

## S

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 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	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 $3 \times 3$ 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.
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	Used to find the location of the edge coordinates in terms of fractions of a pixel.
synchronous	Property or operation that begins an operation and returns control to the program only when the operation is complete.
syntax	Set of rules to which statements must conform in a particular programming language.

## T

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.

*Glossary*

**TIFF** Tagged Image File Format. Image format commonly used for encoding 8-bit and color images.

**truth table** A table associated with a logic operator that describes the rules used for that operation.

# Index

---

## Numbers

3D view, 2-9

## A

Add operator (table), 4-2

advanced binary morphology analysis functions.

*See* binary morphology analysis functions.

alignment, in pattern matching, 9-2

alignment tools, in pattern matching,  
9-18 to 9-19

ambient lighting conditions, in pattern  
matching, 9-4

AND operator (table), 4-2

area of digital particles, 8-5 to 8-7

holes' area, 8-6

number of holes, 8-6

number of pixels, 8-5

particle area, 8-5

ratio, 8-5

scanned area, 8-5

total area, 8-6

area threshold, 8-4

arithmetic operators (table), 4-2

attenuation

highpass FFT filters, 6-11

lowpass FFT filters, 6-8

auto-median function

binary morphology analysis, 7-21

gray-level morphology analysis, 7-38

axes. *See* chords and axes.

axis of symmetry, gradient filter, 5-6

## B

binary morphology analysis functions

advanced, 7-22 to 7-31

border, 7-22

circle, 7-30

comparisons between segmentation and  
skiz functions, 7-28

convex, 7-31

Danielsson, 7-29 to 7-30

distance, 7-28

highpass filters, 7-23 to 7-24

hole filling, 7-22

labeling, 7-22 to 7-23

lowpass and highpass filter  
example, 7-24

lowpass filters, 7-23 to 7-24

segmentation, 7-27 to 7-28

separation, 7-24 to 7-25

skeleton, 7-25 to 7-27

primary, 7-7 to 7-21

auto-median, 7-21

closing, 7-11

dilation, 7-8 to 7-9

erosion, 7-8

erosion and dilation examples,  
7-9 to 7-11

external and internal edge  
examples, 7-13

external edge, 7-12

hit-miss, 7-13 to 7-16

internal edge, 7-12

opening, 7-11

opening and closing examples, 7-12

proper-closing, 7-21

proper-opening, 7-20

thickening, 7-18 to 7-20

thinning, 7-16 to 7-18



binary palette, 2-4  
binary shape matching, 9-15 to 9-16  
blob analysis, in pattern matching,  
9-16 to 9-17  
blur, in pattern matching, 9-5  
border function, binary morphology  
analysis, 7-22  
breadth parameter, 8-7

## C

calibration  
intensity, 8-2  
spatial, 8-1  
center of mass X and center of mass Y, 8-8  
chords and axes, 8-9 to 8-11  
max chord length, 8-9  
max intercept, 8-10  
mean chord X, 8-9  
mean chord Y, 8-10  
mean intercept perpendicular, 8-10  
particle orientation, 8-10 to 8-11  
circle function, binary morphology  
analysis, 7-30  
closing function  
binary morphology analysis  
description, 7-11  
examples, 7-12  
gray-level morphology analysis  
description, 7-34 to 7-35  
examples, 7-35  
clustering technique, in automatic  
thresholding, 7-3 to 7-5  
color images  
components, 1-3  
histogram of color image, 2-7  
processing, 1-5 to 1-6  
thresholding, 7-3  
color lookup table transformation, 1-3  
compactness factor parameter, 8-15  
complex images

definition, 1-3  
number of bytes per pixel (table), 1-4  
connectivity of digital particles, 8-3 to 8-4  
connectivity-4, 8-3 to 8-4  
connectivity-8, 8-3  
contour extraction and highlighting,  
5-14 to 5-15  
contour thickness, 5-15  
conventions used in manual, xv  
convex function, 7-31  
convolution, definition, 5-3  
convolution filters. *See* linear or convolution  
filters.  
convolution kernel, 5-3  
coordinates of objects, 8-7 to 8-9  
center of mass X and center of  
mass Y, 8-8  
max chord X and max chord Y, 8-9  
min(X, Y) and max(X, Y), 8-8  
cross correlation, in pattern matching,  
9-5 to 9-7  
cumulative histogram, 2-6 to 2-7

## D

Danielsson function, 7-29 to 7-30  
densitometry parameters, 8-18  
diagnostic resources, online, A-1  
Difference operator (table), 4-2  
differentiation filter, 5-24 to 5-25  
digital image processing, definition, 1-1  
digital images  
image definition, 1-2  
image resolution, 1-2  
number of planes, 1-2 to 1-3  
overview, 1-1  
digital particles  
criteria for defining, 8-2 to 8-4  
area threshold, 8-4  
connectivity, 8-3 to 8-4  
intensity threshold, 8-2

- measurements, 8-5 to 8-19
  - areas, 8-5 to 8-7
  - chords and axes, 8-9 to 8-11
  - coordinates, 8-7 to 8-9
  - densitometry, 8-18
  - device measurements, 8-19
  - lengths, 8-8 to 8-7
  - shape equivalence, 8-11 to 8-14
  - shape features, 8-14 to 8-18
- dilation function
  - binary morphology analysis
    - concept and mathematics, 7-9
    - examples, 7-9 to 7-11
  - gray-level morphology analysis
    - concept and mathematics, 7-33
    - examples, 7-33 to 7-34
- direction, gradient filter, 5-6 to 5-7
- distance calibration, 8-1
- distance function, binary morphology analysis, 7-28
- diverse measurements, 8-19
- Divide operator (table), 4-2

**E**

- edge detection, in pattern matching, 9-18 to 9-19
- edge extraction and edge highlighting, 5-7 to 5-8
- edge thickness, 5-9
- ellipse major axis parameter, 8-12
- ellipse minor axis parameter, 8-13
- ellipse ratio parameter, 8-13
- elongation factor parameter, 8-14
- entropy technique, in automatic thresholding, 7-5
- Equalize function
  - description, 3-3
  - examples, 3-4 to 3-5
  - summary (table), 3-3
- equivalent ellipse minor axis parameter, 8-12

- erosion function
  - binary morphology analysis
    - concept and mathematics, 7-8
    - examples, 7-9 to 7-11
  - gray-level morphology analysis
    - concept and mathematics, 7-32
    - examples, 7-33 to 7-34
- exponential and gamma correction
  - description, 3-7 to 3-8
  - examples, 3-8 to 3-9
  - summary (table), 3-3
- external edge function, binary morphology analysis
  - description, 7-12
  - examples, 7-13

## F

- Fast Fourier Transform (FFT), 6-1, 6-3 to 6-4
- FFT display, 6-4 to 6-7
  - optical representation, 6-6 to 6-7
  - standard representation, 6-5 to 6-6
- fiducials
  - definition, 9-1
  - example of common fiducial (figure), 9-2
- filters. *See* frequency filters; spatial filters.
- frame. *See* image pixel frame.
- frequency filters, 6-1 to 6-12
  - definition, 6-3 to 6-4
  - FFT display, 6-4 to 6-7
    - optical representation, 6-6 to 6-7
    - standard representation, 6-5 to 6-6
  - highpass FFT filters, 6-10 to 6-12
    - attenuation, 6-11
    - examples, 6-12
    - overview, 6-2
    - truncation, 6-11
  - lowpass FFT filters, 6-8 to 6-10
    - attenuation, 6-8
    - examples, 6-9 to 6-10
    - overview, 6-2

- truncation, 6-9
- mask FFT filters, 6-2
- overview, 6-1 to 6-2

## G

- gauging, in pattern matching, 9-3
- Gaussian filters, 5-19 to 5-21
  - example, 5-20
  - kernel definition, 5-20
  - predefined kernels, 5-21
- gradient filters
  - linear, 5-4 to 5-12
    - definition, 5-5
    - edge extraction and edge highlighting, 5-7 to 5-8
    - edge thickness, 5-9
    - example, 5-5
    - filter axis and direction, 5-6 to 5-7
    - gradient  $5 \times 5$  kernel, 5-12
    - gradient  $7 \times 7$  kernel, 5-12
    - kernel definition, 5-6
    - predefined gradient kernels, 5-10 to 5-12
    - Prewitt filters, 5-10
    - Sobel filters, 5-11
  - nonlinear, 5-24
- gradient palette, 2-4
- gray palette, 2-2
- gray-level images, 1-3
- gray-level morphology analysis functions, 7-32 to 7-37
  - auto-median, 7-38
  - closing, 7-34 to 7-35
  - dilation, 7-33
  - erosion, 7-32
  - erosion and dilation examples, 7-33 to 7-34
  - opening, 7-34
  - opening and closing examples, 7-35
  - proper-closing, 7-36 to 7-37

- proper-opening, 7-36
- gray-level value of pixels, 1-1

## H

- height parameter, 8-7
- hexagonal pixel frame, 1-8
- Heywood circularity factor, 8-15
- highpass filters
  - binary morphology analysis
    - description, 7-23
    - example, 7-24
  - classes (table), 5-3
  - definition, 5-1
- highpass frequency (FFT) filters, 6-10 to 6-12
  - attenuation, 6-11
  - examples, 6-12
  - overview, 6-2
  - truncation, 6-11
- histogram of image. *See* image histogram.
- hit-miss function, binary morphology analysis, 7-13 to 7-16
  - concept and mathematics, 7-14
  - examples, 7-14 to 7-15
  - strategies for using (table), 7-16
- hole filling function, 7-22
- holes' area parameter, 8-6
- holes' perimeter parameter, 8-7
- hydraulic radius parameter, 8-15

## I

- image definition (pixel depth), 1-2
- image files, 1-5
- image histogram, 2-5 to 2-9
  - 3D view, 2-9
  - color image histogram, 2-7
  - cumulative histogram, 2-6 to 2-7
  - definition, 2-5
  - interpretation, 2-7
  - line profile, 2-8 to 2-9

- linear histogram, 2-6
  - scale of histogram, 2-7 to 2-8
- image pixel frame, 1-6 to 1-8
  - examples (figure), 1-7
  - hexagonal, 1-8
  - rectangular, 1-7
- image types and formats
  - color images, 1-3
  - complex images, 1-3 to 1-4
  - gray-level images, 1-3
- image understanding, in pattern matching, 9-8
- images. *See also* digital images.
  - definition, 1-1
- inner gradient function, binary morphology
  - analysis, 7-12
- inspection, in pattern matching, 9-3
- intensity calibration, 8-2
- intensity threshold, 8-2
- interclass variance technique, in automatic
  - thresholding, 7-6
- internal edge function, binary morphology
  - analysis
    - description, 7-12
    - examples, 7-13
- interpretation of histogram, 2-7

## L

- labeling function, 7-22 to 7-23
- Laplacian filters, 5-12 to 5-16
  - contour extraction and highlighting,
    - 5-14 to 5-15
  - contour thickness, 5-15
  - example, 5-12 to 5-13
  - kernel definition, 5-13
  - predefined kernels, 5-16
- length of digital particles, 8-8 to 8-7
  - breadth, 8-7
  - height, 8-7
  - holes' perimeter, 8-7
  - particle perimeter, 8-7
- lighting conditions, in pattern matching, 9-4
- line profile, 2-8 to 2-9
- linear histogram, 2-6
- linear or convolution filters, 5-3 to 5-21
  - classes (table), 5-3
  - Gaussian filters, 5-19 to 5-21
    - example, 5-20
    - kernel definition, 5-20
    - predefined kernels, 5-21
  - gradient filters, 5-4 to 5-12
    - edge extraction and edge
      - highlighting, 5-7 to 5-8
    - edge thickness, 5-9
    - example, 5-5
    - filter axis and direction, 5-6 to 5-7
    - gradient  $5 \times 5$  kernel, 5-12
    - gradient  $7 \times 7$  kernel, 5-12
    - kernel definition, 5-6
    - predefined gradient kernels,
      - 5-10 to 5-12
    - Prewitt filters, 5-10
    - Sobel filters, 5-11
  - Laplacian filters, 5-12 to 5-16
    - contour extraction and highlighting,
      - 5-14 to 5-15
    - contour thickness, 5-15
    - example, 5-12 to 5-13
    - kernel definition, 5-13
    - predefined kernels, 5-16
  - mathematical concepts, 5-2
  - overview, 5-3 to 5-4
  - smoothing filters, 5-17 to 5-19
    - example, 5-17
    - kernel definition, 5-17 to 5-18
    - predefined kernels, 5-19
- logarithmic and inverse gamma correction
  - description, 3-5 to 3-6
  - examples, 3-6 to 3-7
  - summary (table), 3-3
- logic operators
  - examples, 4-4 to 4-6

- extracting or removing information in
    - image (example), 4-3
    - list of operators (table), 4-2 to 4-3
    - purpose and use, 4-2
    - truth tables, 4-3 to 4-4
  - lookup table transformations
    - color image conversion, 1-3
    - examples, 3-2
    - overview, 3-1 to 3-2
  - lookup tables
    - Equalize, 3-3 to 3-5
    - exponential and gamma correction, 3-7 to 3-9
    - logarithmic and inverse gamma correction, 3-5 to 3-7
    - predefined LUTs, 3-3
    - summary (table), 3-3
  - lowpass filters
    - binary morphology analysis
      - description, 7-23
      - example, 7-24
    - classes (table), 5-3
    - definition, 5-1
    - nonlinear, 5-25 to 5-26
  - lowpass frequency (FFT) filters, 6-8 to 6-10
    - attenuation, 6-8
    - examples, 6-9 to 6-10
    - overview, 6-2
    - truncation, 6-9
  - L-skeleton function, 7-25 to 7-26
- M**
- mask FFT filters, 6-2
  - Mask operator (table), 4-2
  - max chord length parameter, 8-9
  - max chord X and max chord Y parameter, 8-9
  - max intercept parameter, 8-10
  - Max operator (table), 4-3
  - mean chord X parameter, 8-9
  - mean chord Y parameter, 8-10
  - mean intercept perpendicular parameter, 8-10
  - Mean operator (table), 4-3
  - median filter, 5-26
  - metric technique, in automatic thresholding, 7-5
  - Min operator (table), 4-3
  - minimum contrast, in pattern matching, 9-13
  - min(X, Y) and max(X, Y) parameter, 8-8
  - moments of inertia  $I_{xx}$ ,  $I_{yy}$ ,  $I_{xy}$ , 8-14
  - moments technique, in automatic thresholding, 7-5
  - morphological transformations
    - categories, 7-1
    - definition, 7-1
    - pixel frame concept and, 1-6
  - morphology analysis, 7-1 to 7-37
    - advanced binary functions, 7-22 to 7-31
      - border, 7-22
      - circle, 7-30
      - comparisons between segmentation and skiz functions, 7-28
      - convex, 7-31
      - Danielsson, 7-29 to 7-30
      - distance, 7-28
      - highpass filters, 7-23 to 7-24
      - hole filling, 7-22
      - labeling, 7-22 to 7-23
      - lowpass and highpass filter example, 7-24
      - lowpass filters, 7-23 to 7-24
      - segmentation, 7-27 to 7-28
      - separation, 7-24 to 7-25
      - skeleton, 7-25 to 7-27
    - gray-level functions, 7-32 to 7-37
      - auto-median, 7-38
      - closing, 7-34 to 7-35
      - dilation, 7-33
      - erosion, 7-32
      - erosion and dilation examples, 7-33 to 7-34
      - opening, 7-34

- opening and closing examples, 7-35
- proper-closing, 7-36 to 7-37
- proper-opening, 7-36
- overview, 7-1
- primary binary functions, 7-7 to 7-21
  - auto-median, 7-21
  - closing, 7-11
  - dilation, 7-8 to 7-9
  - erosion, 7-8
  - erosion and dilation examples, 7-9 to 7-11
  - external and internal edge
    - examples, 7-13
  - external edge, 7-12
  - hit-miss, 7-13 to 7-16
  - inner gradient, 7-12
  - internal edge, 7-12
  - opening, 7-11
  - opening and closing examples, 7-12
  - outer gradient, 7-12
  - proper-closing, 7-21
  - proper-opening, 7-20
  - thickening, 7-18 to 7-20
  - thinning, 7-16 to 7-18
- structuring element, 7-6 to 7-7
- thresholding, 7-1 to 7-6
  - automatic, 7-3 to 7-6
  - clustering, 7-3 to 7-5
  - color image, 7-3
  - entropy, 7-5
  - example, 7-2
  - interclass variance, 7-6
  - metric, 7-5
  - moments, 7-5
- M-skeleton function, 7-26
- multiple instances of patterns, 9-3 to 9-4
- Multiply operator (table), 4-2

## N

- NAND operator (table), 4-2
- National Instruments Web support, A-1 to A-2
- neighborhood operations
  - definition, 1-6
  - hexagonal frame, 1-8
  - pixel neighborhood representations (figure), 1-7
  - rectangular frame, 1-7
- noise, in pattern matching, 9-5
- nonlinear filters, 5-22 to 5-27
  - classes (table), 5-3
  - differentiation filter, 5-24 to 5-25
  - gradient filter, 5-24
  - lowpass filter, 5-25 to 5-26
  - mathematical concepts, 5-2
  - median filter, 5-26
  - Nth order filter, 5-26 to 5-27
  - Prewitt filter, 5-22
  - Roberts filter, 5-24
  - Sigma filter, 5-25
  - Sobel, 5-22 to 5-24
- NOR operator (table), 4-2
- Nth order filter
  - description, 5-26 to 5-27
  - examples, 5-27
- number of holes parameter, 8-6
- number of pixels parameter, 8-5
- number of planes, 1-2 to 1-3

## O

- online problem-solving and diagnostic resources, A-1
- opening function
  - binary morphology analysis
    - description, 7-11
    - opening and closing examples, 7-12
  - gray-level morphology analysis
    - description, 7-34

- examples, 7-35
  - operators
    - arithmetic, 4-2
    - concepts and mathematics, 4-1
    - logic, 4-2 to 4-6
  - optical representation, FFT display, 6-6 to 6-7
  - OR operator (table), 4-2
  - outer gradient function, binary morphology analysis, 7-12
- P**
- palettes, 2-1 to 2-4
    - binary palette, 2-4
    - definition, 2-1
    - gradient palette, 2-4
    - gray palette, 2-2
    - predefined color palettes, 2-2
    - purpose and use, 2-1 to 2-2
    - rainbow palette, 2-3
    - temperature palette, 2-3
  - particle area parameter, 8-5
  - particle orientation parameter, 8-10 to 8-11
  - particle perimeter parameter, 8-7
  - particles. *See* digital particles.
  - pattern matching, 9-1 to 9-19
    - applications, 9-2 to 9-3
    - binary shape matching, 9-15 to 9-16
    - blob analysis, 9-16 to 9-17
    - edge detection and alignment tools, 9-18 to 9-19
    - new techniques, 9-8 to 9-14
      - defining and creating template images, 9-10 to 9-11
      - defining search area, 9-12
      - general steps, 9-9
      - overview, 9-8 to 9-9
      - ranking method for verifying results, 9-14
      - setting matching parameters and tolerances, 9-13
      - testing search tool on test images, 9-13 to 9-14
      - training pattern matching tool using reference pattern, 9-11
    - overview, 9-1 to 9-2
    - reliability and accuracy, 9-3 to 9-5
      - ambient lighting conditions, 9-3 to 9-4
      - blur and noise conditions, 9-5
      - pattern orientation and multiple instances, 9-3 to 9-4
    - traditional techniques, 9-5 to 9-7
      - cross correlation, 9-5 to 9-7
      - pyramidal matching, 9-7
      - scale-invariant matching, 9-7
  - picture element, 1-1
  - pixel calibration, 8-1
  - pixel depth (image definition), 1-2
  - pixel frame. *See* image pixel frame.
  - pixels
    - gray-level value, 1-1
    - image pixel frame, 1-6 to 1-8
    - spatial coordinates, 1-1
  - planes, number in image, 1-2 to 1-3
  - Power 1/Y transformation (example), 3-6
  - Power Y transformation (example), 3-9
  - Prewitt filters
    - nonlinear, 5-22
    - predefined gradient filter, 5-10
  - primary binary morphology analysis functions. *See* binary morphology analysis functions.
  - problem-solving and diagnostic resources, online, A-1
  - proper-closing function
    - binary morphology analysis, 7-21
    - gray-level morphology analysis, 7-36 to 7-37

proper-opening function  
 binary morphology analysis, 7-20  
 gray-level morphology analysis, 7-36  
 pyramidal matching, 9-7

## Q

quantitative analysis, 8-1 to 8-19  
 digital particles, 8-2 to 8-4  
 area threshold, 8-4  
 connectivity, 8-3 to 8-4  
 intensity threshold, 8-2  
 intensity calibration, 8-2  
 particle measurements, 8-5 to 8-19  
 areas, 8-5 to 8-7  
 chords and axes, 8-9 to 8-11  
 coordinates, 8-7 to 8-9  
 densitometry, 8-18  
 device measurements, 8-19  
 lengths, 8-8 to 8-7  
 shape equivalence, 8-11 to 8-14  
 shape features, 8-14 to 8-18  
 spatial calibration, 8-1

## R

rainbow palette, 2-3  
 ranking method, in pattern matching, 9-14  
 ratio area parameter, 8-5  
 rectangle big side parameter, 8-13  
 rectangle ratio parameter, 8-14  
 rectangle small side parameter, 8-13  
 rectangular pixel frame, 1-7  
 reference pattern, in pattern matching, 9-11  
 Remainder operator (table), 4-2  
 Roberts filter, 5-24  
 rotation angle ranges, in pattern matching, 9-13

## S

scale of histograms, 2-7 to 2-8  
 scale-invariant matching, 9-7  
 scanned area parameter, 8-5  
 search area definition, in pattern matching, 9-12  
 segmentation function  
 compared with skiz function, 7-28  
 description, 7-27 to 7-28  
 separation function, binary morphology analysis, 7-24 to 7-25  
 shape equivalence, 8-11 to 8-14  
 ellipse major axis, 8-12  
 ellipse minor axis, 8-13  
 ellipse ratio, 8-13  
 equivalent ellipse minor axis, 8-12  
 rectangle big side, 8-13  
 rectangle ratio, 8-14  
 rectangle small side, 8-13  
 shape features, 8-14 to 8-18  
 compactness factor, 8-15  
 elongation factor, 8-14  
 Heywood circularity factor, 8-15  
 hydraulic radius, 8-15  
 moments of inertia  $I_{xx}$ ,  $I_{yy}$ ,  $I_{xy}$ , 8-14  
 Waddell disk diameter, 8-16 to 8-18  
 derived measurements (table), 8-16 to 8-18  
 primary measurements, 8-16  
 Sigma filter, 5-25  
 skeleton functions, 7-25 to 7-27  
 comparison between segmentation and skiz functions, 7-28  
 L-skeleton, 7-25 to 7-26  
 M-skeleton, 7-26  
 skiz, 7-26 to 7-27  
 skiz function  
 compared with segmentation function, 7-28  
 description, 7-26 to 7-27



- smoothing filters, 5-17 to 5-19
  - example, 5-17
  - kernel definition, 5-17 to 5-18
  - predefined kernels, 5-19
- Sobel filters
  - nonlinear, 5-22 to 5-24
  - predefined gradient filters, 5-11
- software-related resources, A-2
- spatial calibration, 8-1
- spatial filters, 5-1 to 5-27
  - categories, 5-1
  - classification summary (table), 5-3
  - concepts and mathematics, 5-1 to 5-2
  - definition, 5-1
  - linear or convolution filters, 5-3 to 5-21
    - Gaussian filters, 5-19 to 5-21
    - gradient filter, 5-4 to 5-12
    - Laplacian filters, 5-12 to 5-16
    - smoothing filters, 5-17 to 5-19
  - nonlinear filters, 5-22 to 5-27
    - differentiation filter, 5-24 to 5-25
    - gradient filter, 5-24
    - lowpass filter, 5-25 to 5-26
    - median filter, 5-26
    - Nth order filter, 5-26 to 5-27
    - Prewitt filter, 5-22
    - Roberts filter, 5-24
    - Sigma filter, 5-25
    - Sobel, 5-22 to 5-24
  - pixel frame concept and, 1-6
- spatial resolution of images, 1-2
- Square or Power Y transformation (example), 3-9
- Square Root or Power 1/Y transformation (example), 3-7
- standard representation, FFT display, 6-5 to 6-6
- structuring element, in morphology analysis, 7-6 to 7-7
- Subtract operator (table), 4-2

## T

- technical support resources, A-1 to A-2
- temperature palette, 2-3
- template images, in pattern matching, 9-10 to 9-11
- thickening function, binary morphology
  - analysis, 7-18 to 7-20
  - description, 7-18
  - examples, 7-19 to 7-20
- thinning function, binary morphology
  - analysis, 7-16 to 7-18
  - description, 7-16 to 7-17
  - examples, 7-17 to 7-18
- three-dimensional (3D) view, 2-9
- threshold
  - area threshold, 8-4
  - intensity threshold, 8-2
- thresholding, 7-1 to 7-6
  - automatic, 7-3 to 7-6
    - clustering, 7-3 to 7-5
    - entropy, 7-5
    - interclass variance, 7-6
    - metric, 7-5
    - moments, 7-5
  - color image, 7-3
  - example, 7-2
  - overview, 7-1 to 7-2
- tools and utilities, 2-1 to 2-9
  - image histogram, 2-5 to 2-9
  - palettes, 2-1 to 2-4
- total area parameter, 8-6
- truncation
  - highpass FFT filters, 6-11
  - lowpass FFT filters, 6-9
- truth tables, 4-3 to 4-4

## **W**

- Waddel disk diameter, 8-16 to 8-18
  - derived measurements (table),  
8-16 to 8-18
  - primary measurements, 8-16
- Web support from National Instruments,  
A-1 to A-2
  - online problem-solving and diagnostic  
resources, A-1
  - software-related resources, A-2
- Worldwide technical support, A-2

## **X**

- XOR operator (table), 4-2