IMAQ

Image Processing Manual

NATIONAL INSTRUMENTS™

February 1999 Edition
Part Number 372320A-01

Important Information

Contents

Chapter 3 Lookup Transformations

Chapter 4 Operators

Chapter 5 Spatial Filtering

Chapter 8
Quantitative Analysis

Tables

About This Manual

The *Image Processing Manual* contains the information you need to get started with IMAQ controls.

This manual contains step-by-step instructions for building applications with IMAQ Vision. You can modify these sample applications to suit your needs. This manual does not show you how to use every control or solve every possible programming problem. Use the online reference for further, function-specific information.

To use this manual, you already should be familiar with one of the supported programming environments and Windows 95/98 or Windows NT.

This manual presents the basics of computer-based vision applications.

Conventions

The following conventions appear in this manual:

<>	Angle brackets that contain numbers separated by an ellipsis represent a range of values associated with a bit or signal name—for example, DBIO<3..0>.
»	The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
✎	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.
<Control>	Key names are capitalized.
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.

`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.
paths	Paths in this manual are denoted using backslashes (\) to separate drive names, directories, folders, and files.

1 Images

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

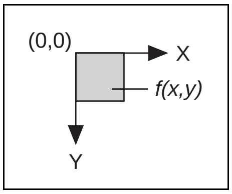
Introduction to Digital Images

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

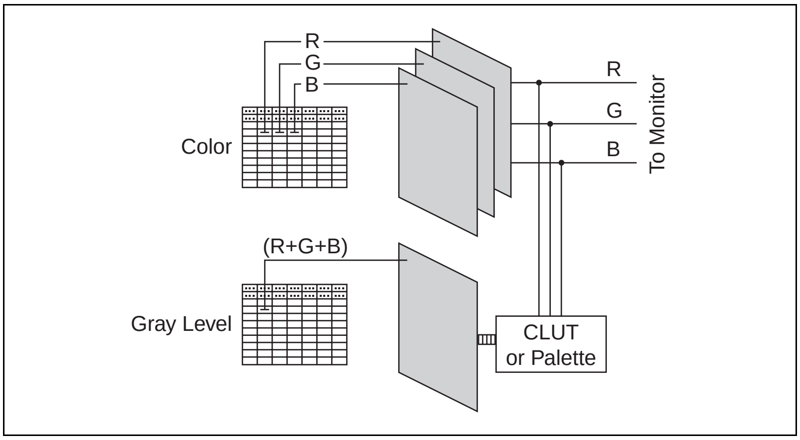
The *spatial resolution* of an image is its number of rows and columns of pixels. An image composed of *m* rows and *n* columns has a resolution of *mn*. This image has *n* pixels along its horizontal axis and *m* pixels along its vertical axis.

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 lookup 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 *PICT* (Macintosh only), *BMP* (PC only), *TIFF*, *RASTR*, and *AIPD*. Standard 16-bit gray-level formats are TIFF 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, 16-bit, or 32-bit pixel depth can be imported into the IMAQ Vision libraries.

Color Images

Color images are composed of three planes of pixels in which each pixel has a red, green, and blue intensity, each coded on 8-bit planes. Color images coded using the *RGB-chunky* standard contain an extra 8-bit plane, called the *alpha channel*. These images have a definition of 32 bit or 4×8 bit. Standard color formats are PICT, BMP, TIFF, 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.

The following table 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

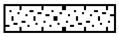
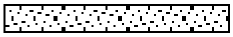
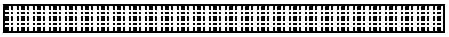
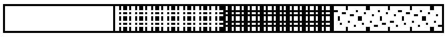
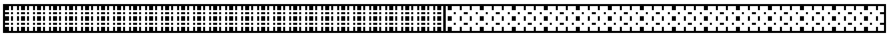
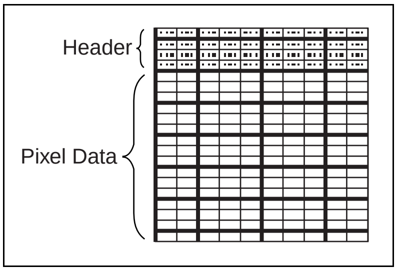
Image Type	Number of Bytes Per Pixel Data
8-bit (Unsigned) Integer Gray-Level (1 byte or 8-bit)	 8-bit for the gray-level intensity
16-bit (Signed) Integer Gray-Level (2 bytes or 16-bit)	 16-bit for the gray-level intensity
32-bit Floating-Point Gray-Level (4 bytes or 32-bit)	 32-bit floating for the gray-level intensity
Color (3 bytes or 24-bit)	 8-bit for the alpha value (not used) · 8-bit for the red intensity · 8-bit for the green intensity · 8-bit for the blue intensity
Complex (8 bytes or 64-bit)	 32-bit floating for the real part · 32-bit floating for the imaginary part

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), *HSL* (hue, saturation, and lightness), 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.

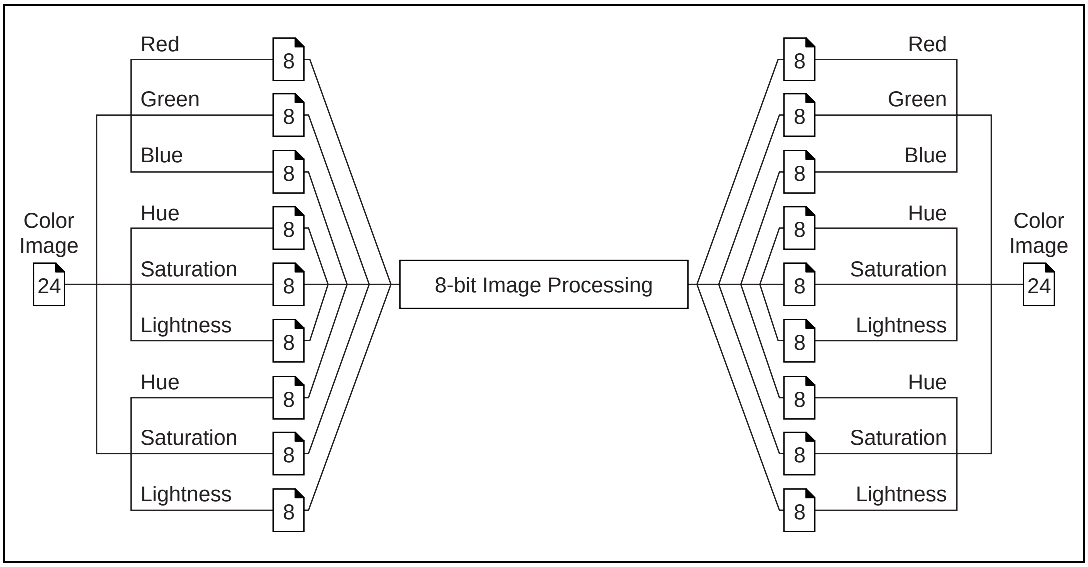
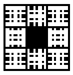
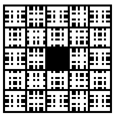
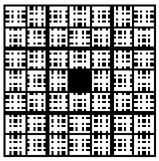
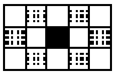
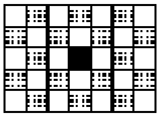
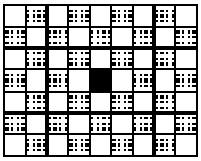


Image Pixel Frame

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 the following table. The pixels considered as neighbors of the given pixel (shown in solid) are indicated by the shaded pattern.

Pixel Frame	Neighborhood Size		
Rectangular	3 × 3 	5 × 5 	7 × 7 
Hexagonal	5 × 3 	7 × 5 	9 × 7 

Rectangular Frame

Each pixel in a rectangular frame is surrounded by eight neighbors.

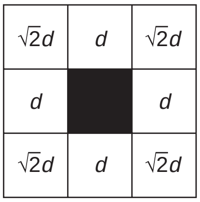


Figure 1-1. Rectangular Frame

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

Each pixel in a hexagonal frame is surrounded by six neighbors. Each neighbor is found at an equal distance *d* from the central pixel.

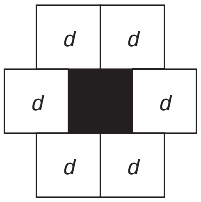


Figure 1-2. Hexagonal Frame

2

Tools and Utilities

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

Palettes

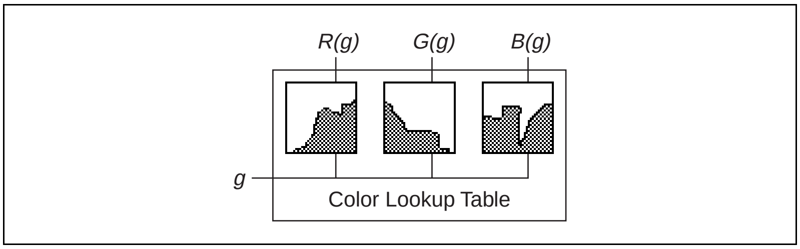
At the time an image is displayed on the screen, the value of each pixel is converted into a red, green, and blue intensity that produces a color. This conversion is defined in a table called a color lookup table (CLUT). For 8-bit images, the CLUT associates a color to each gray-level value and produces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

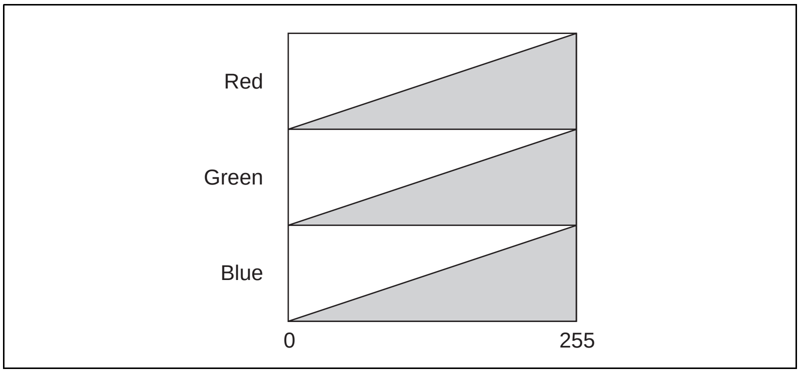
- Gray level 0 is assigned to black.
- Gray level 255 is assigned to white.

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 lookup 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.

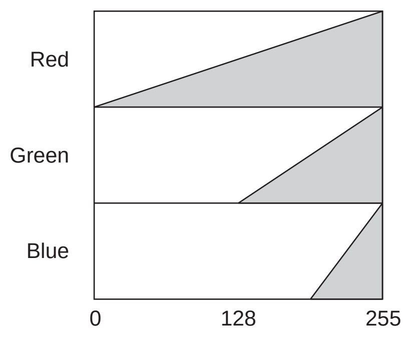
B&W (Gray) Palette

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



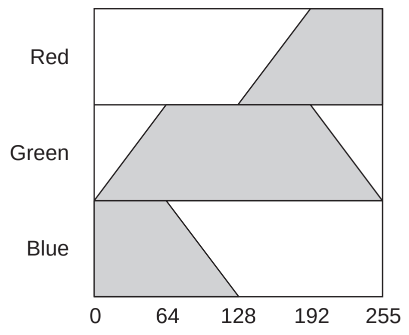
Temperature Palette

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



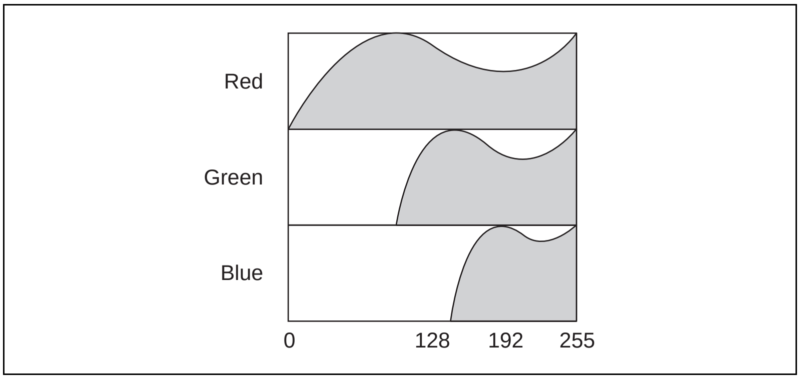
Rainbow Palette

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

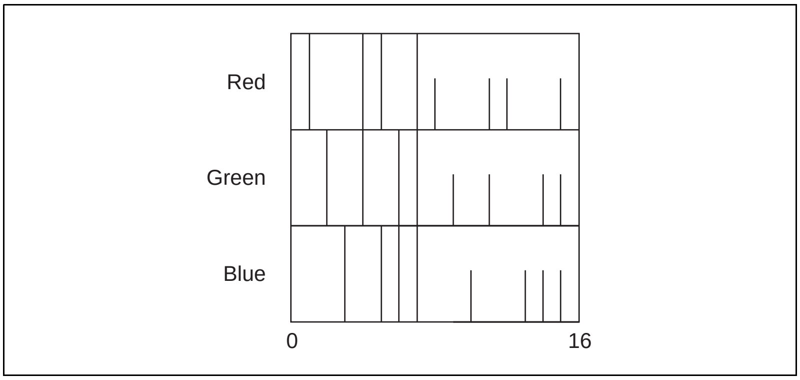
This palette has a gradation from red to white with a prominent range of light blue in the upper value range. 0 is black and 255 is white.



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 gray-scale range $[0, \ldots, k, \ldots, 255]$ such that the number of pixels equal to the gray-level value k is

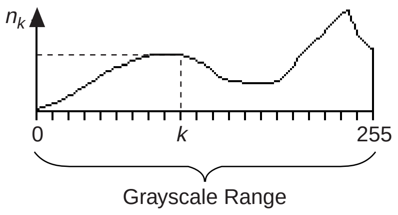
$$H(k) = n_k$$

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 plotted per image: 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.

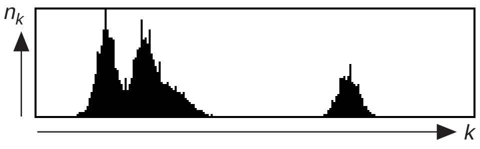


Figure 2-1. Linear Vertical Scale

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.

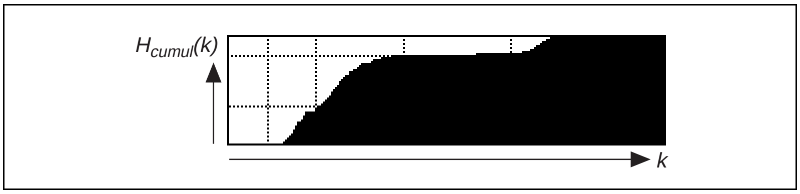


Figure 2-2. Linear Cumulative Scale

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 the previous example, the linear histogram reveals that the image is composed of three major elements. The cumulative histogram shows that the two left-most peaks compose approximately 80 percent of the image, while the remaining 20 percent 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 *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.

The following two figures illustrate 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.

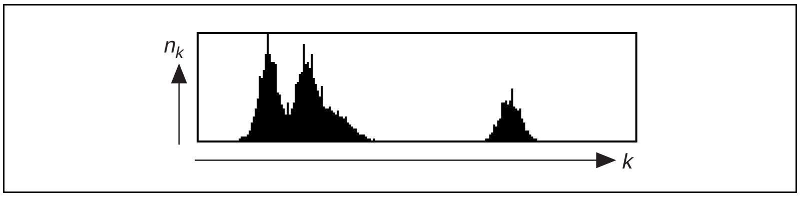


Figure 2-3. Linear Vertical Scale

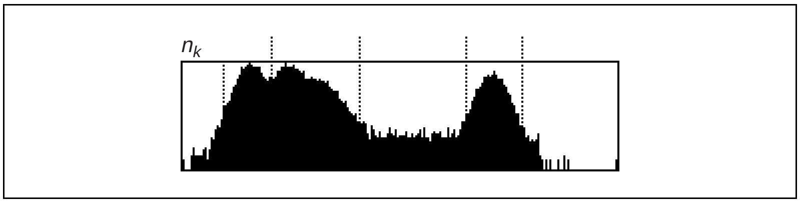
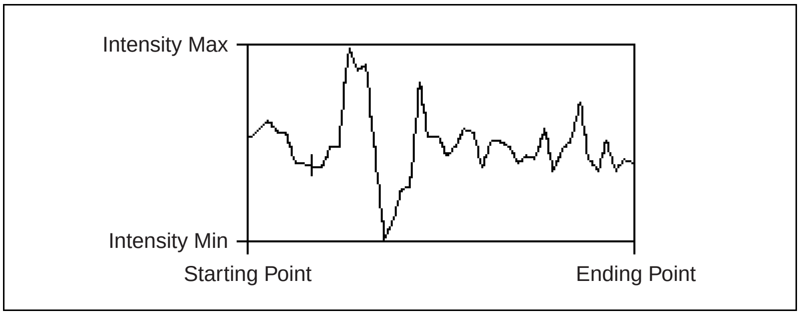


Figure 2-4. Logarithmic Vertical Scale

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.

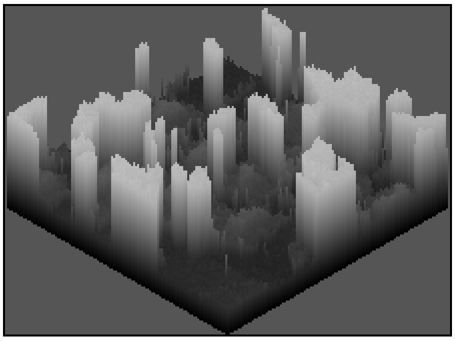


The peaks and valleys reveal increases and decreases of the light intensity along the line selected in the image. Their width and magnitude are proportional to the size and intensity of their related regions.

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.



3 Lookup Transformations

This chapter provides an overview of lookup table transformations.

About Lookup Table Transformations

The *lookup table* (LUT) transformations are basic image-processing functions that you can use to improve the contrast and brightness of an image by modifying the intensity dynamic of regions with poor contrast. The 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*.

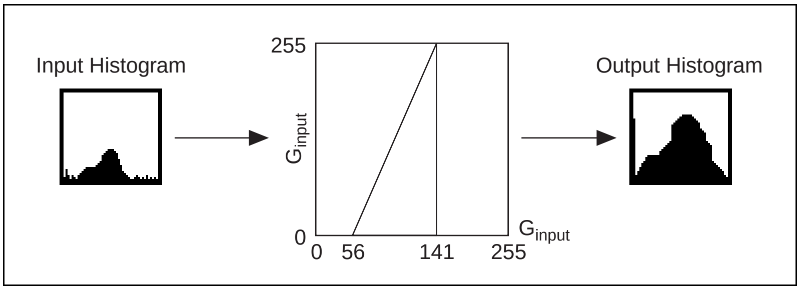
An LUT transformation converts input gray-level values (those from the source image) into other gray-level values (in the transformed image). The transfer function has an intended effect on the brightness and contrast of the image.

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

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

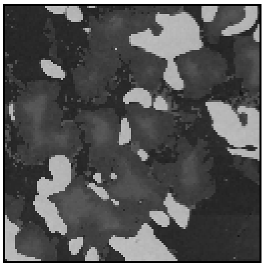
where F is a linear or nonlinear, continuous or discontinuous transfer function defined over the interval [0, max].

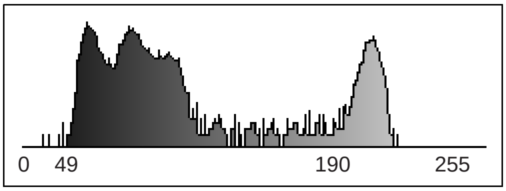
In the case of an 8-bit resolution, an LUT is a table of 256 elements. Each element of the array represents an input gray-level value. Its content indicates the output value.



Example

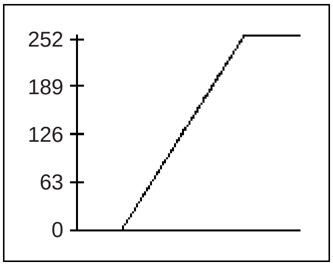
This example uses the following source image. In the histogram of the source image, the gray-level intervals [0, 49] and [191, 255] do not contain significant information.



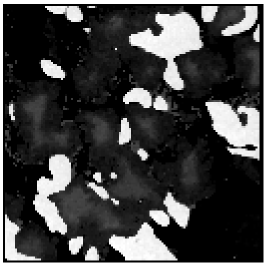


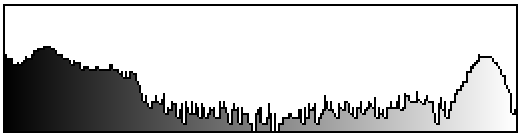
Using the following LUT transformation, any pixel with a value less than 49 is set to 0, and any pixel with a value greater than 191 is set to 255. The interval [50, 190] expands to [1, 255], increasing the intensity dynamic of the regions with a concentration of pixels in the gray-level range [50, 190].

If G_{input} is between [0, 49],
then $F(G_{input}) = 0$,
If G_{input} is between [191, 255],
then $F(G_{input}) = 255$,
else $F(G_{input}) = 1.8 \times G_{input} - 91$.



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

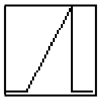
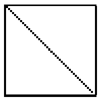
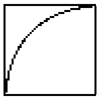
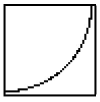




Predefined Lookup Tables

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

The following table 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).

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.
Reverse		Reverses the pixel values, producing a photometric negative of the 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 increases the contrast in bright regions. Decreases the contrast in the dark regions.

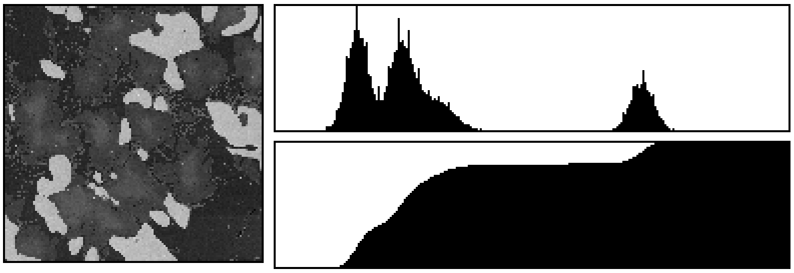
Equalize

The *Equalize* function alters the gray-level value of pixels so they become distributed evenly in the defined gray-scale 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 gray-level intervals are not used.

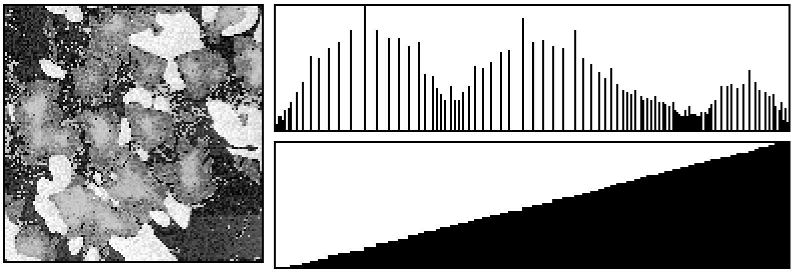
The equalization can be limited to a gray-level interval, also called the equalization range. In this case, the function evenly distributes the pixels belonging to the equalization range over the full interval (0 to 255 for an 8-bit image) and the other pixels are set to 0. The image produced reveals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



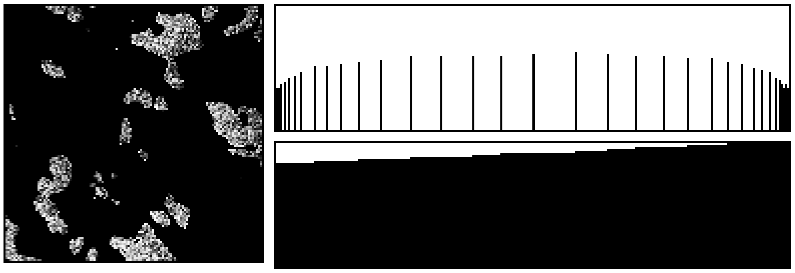
An equalization from [0, 255] to [0, 255] produces the following image and histograms.



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 [1, 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.



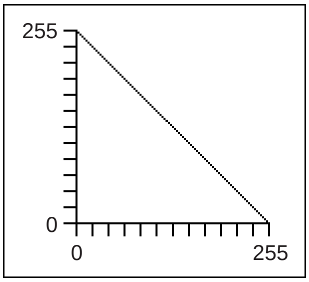
Reverse

The *Reverse* function displays the photometric negative of an image.

$G_{output} = Maximum - G_{input}$

For an 8-bit image, *Maximum* = 255. Therefore,

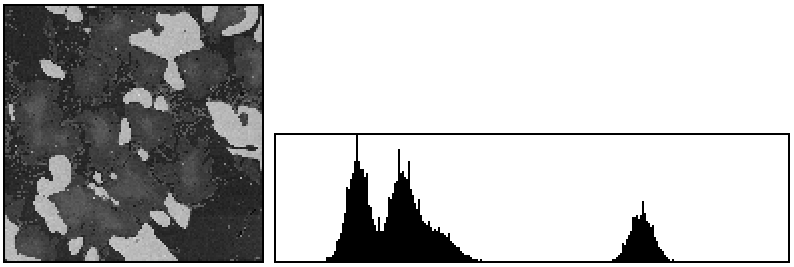
G_{output}	=	$255 - G_{input}$
0	corresponds to	255
1	corresponds to	254
2	corresponds to	253
...		
128	corresponds to	128
...		
253	corresponds to	2
254	corresponds to	1
255	corresponds to	0



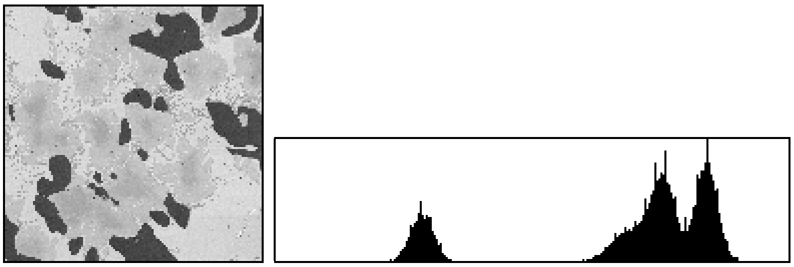
The histogram of a reversed image is equal to the histogram of the original image after a vertical symmetry centered on the gray-level value 128 (when processing an 8-bit image).

Example

This example uses the following original image and histogram.



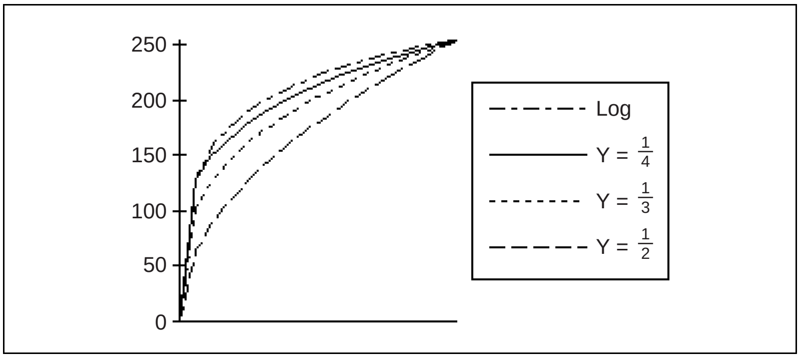
A Reverse transformation produces the following image and histogram.



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 B&W (or 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 lookup curve is plotted horizontally to give an output value.

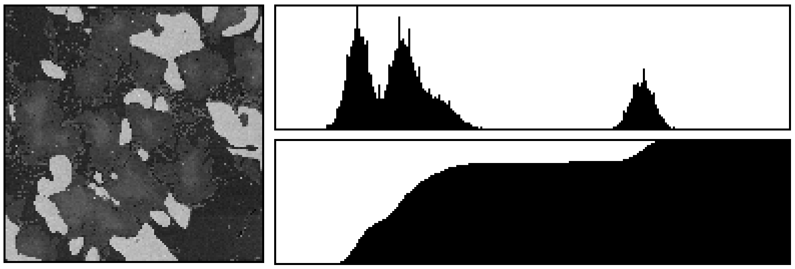


The *Logarithmic*, *Square Root*, and *Power 1/Y* functions expand intervals containing low gray-level values while compressing intervals containing high gray-level values.

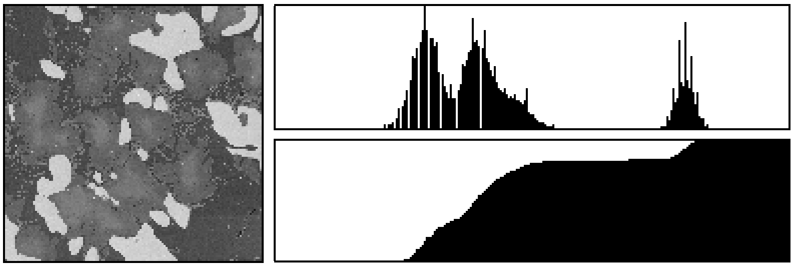
The higher the gamma coefficient *Y*, the stronger the intensity correction. The Logarithmic correction has a stronger effect than the Power 1/Y function.

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.

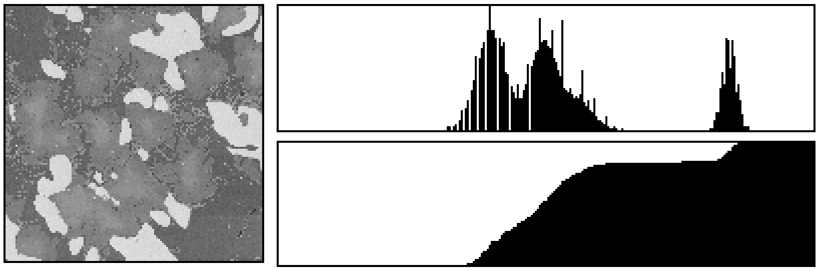
The following graphic shows the original image and histograms.



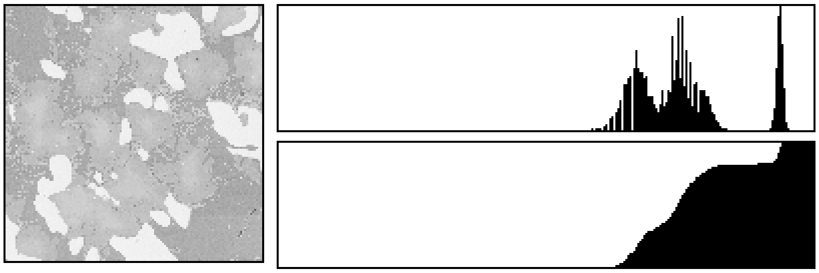
A Power 1/Y transformation (where *Y* = 1.5) produces the following image and histograms.



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



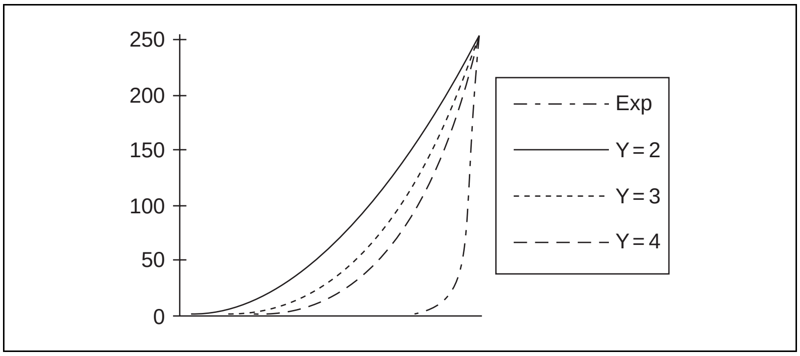
A Logarithm transformation produces the following image and histograms.



Exponential and Gamma Correction

The *exponential and gamma corrections* expand high gray-level ranges while compressing low gray-level ranges. When using the B&W (or 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 lookup curve then is plotted horizontally to give an output value.

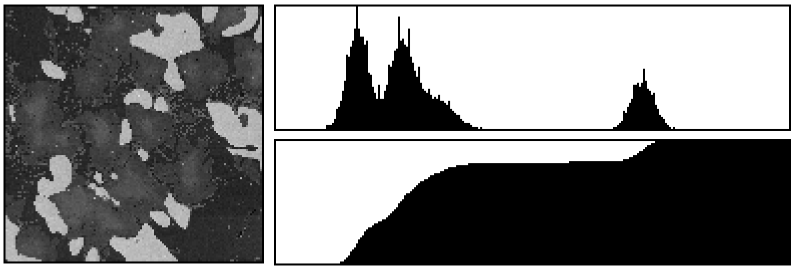


The *Exponential*, *Square*, and *Power Y* functions expand intervals containing high gray-level values while compressing intervals containing low gray-level values.

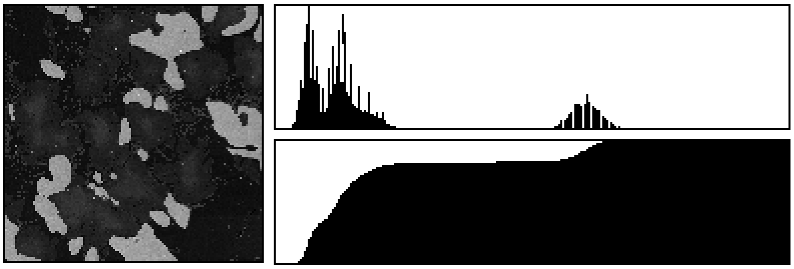
The higher the gamma coefficient *Y*, the stronger the intensity correction. The Exponential correction has a stronger effect than the Power Y function.

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.

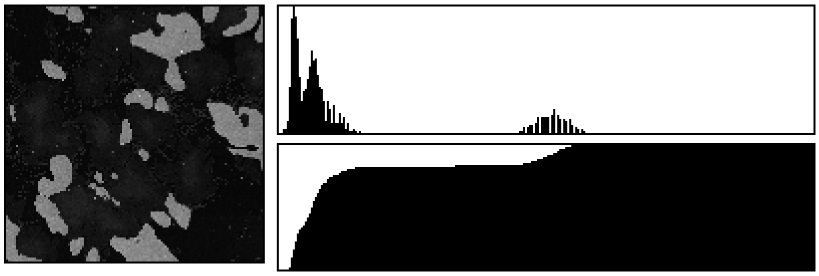
The following graphic shows the original image and histograms.



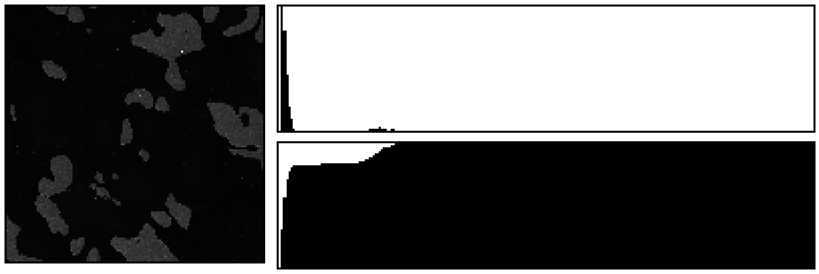
A Power Y transformation (where *Y* = 1.5) produces the following image and histograms.



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



An Exponential transformation produces the following image and histograms.



4 Operators

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

Concepts and Mathematics

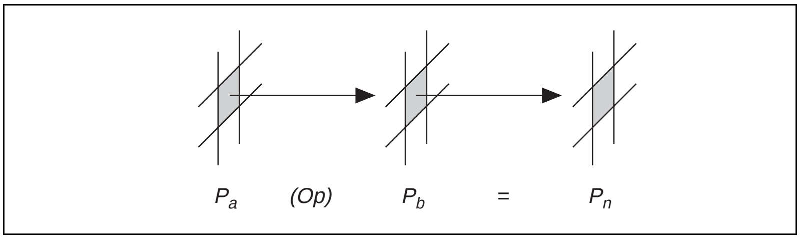
Arithmetic and logic *operators* mask, combine, and compare images. Common applications of these operators include time-lapse comparisons, identification of the union or intersection between images, and comparisons between several images and a model. Operators also can be used to *threshold* or *mask*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

In the case of images with 8-bit resolution, the following equations describe the usage of the *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	$p_n = p_a \mathrm{mod} p_b$

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 bit-wise operators. They manipulate gray-level values coded on one byte at the bit level. The *truth tables* for logic operators are presented in the *Truth Tables* section.

Operator	Equation
AND	$p_n = p_a$ AND p_b
NAND	$p_n = p_a$ NAND p_b
OR	$p_n = p_a$ OR p_b
NOR	$p_n = p_a$ NOR p_b
XOR	$p_n = p_a$ XOR p_b
Difference	$p_n = p_a$ AND (NOT p_b)

Mask	*if* $p_b = 0$,
	then $p_n = 0$,
	else $p_n = p_a$

Operator	Equation
Mean	p_n = mean[p_a, p_b]
Max	p_n = max[p_a, p_b]
Min	p_n = min[p_a, p_b]

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

The following table illustrates how logic operations can be used to extract or remove information in an image.

For a given p_a	***If p_b = 255, then***	***If p_b = 0, then***
AND	p_a AND 255 = p_a	p_a AND 0 = 0
NAND	p_a NAND 255 = NOT p_a	p_a NAND 0 = 255
OR	p_a OR 255 = 255	p_a OR 0 = p_a
NOR	p_a NOR 255 = 0	p_a NOR 0 = NOT p_a
XOR	p_a XOR 255 = NOT p_a	p_a XOR 0 = p_a
Logic Difference	p_a – NOT 255 = p_a	p_a – NOT 0 = 0

Truth Tables

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

AND

	***b* = 0**	***b* = 1**
***a* = 0**	0	0
***a* = 1**	0	1

NAND

	***b* = 0**	***b* = 1**
***a* = 0**	1	1
***a* = 1**	1	0

OR

	***b* = 0**	***b* = 1**
***a* = 0**	0	1
***a* = 1**	1	1

NOR

	***b* = 0**	***b* = 1**
***a* = 0**	1	0
***a* = 1**	0	0

XOR

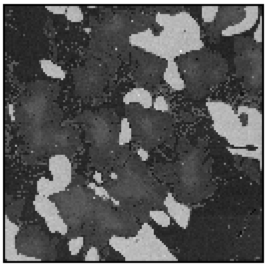
	***b* = 0**	***b* = 1**
***a* = 0**	0	1
***a* = 1**	1	0

NOT

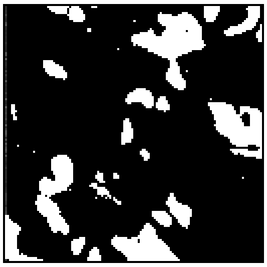
	NOT *a*
***a* = 0**	1
***a* = 1**	0

Example 1

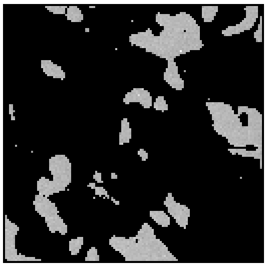
The following series of graphics illustrates images in which regions of interest have been isolated in a binary format, retouched with morphological manipulations, and finally multiplied by 255. The following gray-level source image is used for this example.



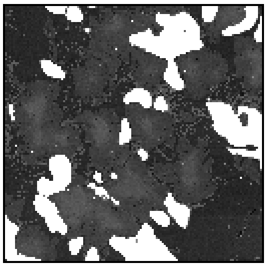
The following *mask image* results.



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

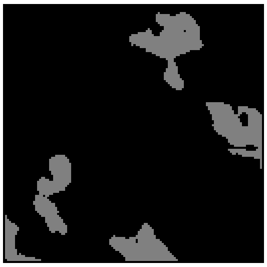


The operation (*source image* OR *mask image*) has the effect of restoring the original intensity of the background region in the mask.

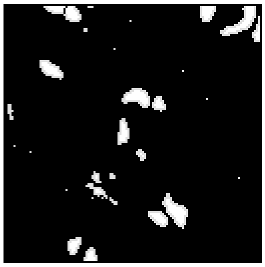


Example 2

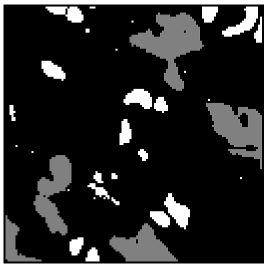
An image reveals two groups of objects that require different processing results in two binary images. Multiplying each binary image by a constant and applying an OR operation produces an image that shows their union, as illustrated in the following series of graphics. The following image illustrates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



The following image illustrates *Object Group #2* × 255.



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



5 Spatial Filtering

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

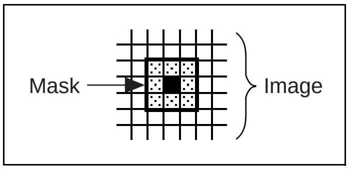
Concept and Mathematics

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 the detection of edges along a specific direction, the contouring of patterns, noise reduction, and detail outlining or smoothing.

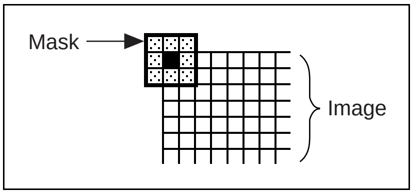
Spatial filters fall into two categories:

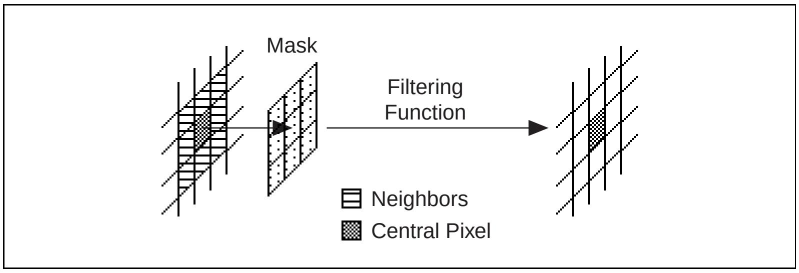
- *Highpass filters* emphasize significant variations of the light intensity usually found at the boundary of objects.
- *Lowpass filters* attenuate variations of the light intensity. They have the tendency to smooth images by eliminating details and blurring edges.

In the case of a 3 × 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 × 5 matrix specifies 24 neighbors, a 7 × 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 × 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

The following table describes the classification of spatial filters.

	Highpass Filters	**Lowpass Filters**
Linear Filters	Gradient, Laplacian	Smoothing, Gaussian
Nonlinear Filters	Gradient, Roberts, Sobel, Prewitt, Differentiation, Sigma	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 × 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 $(1/N)\Sigma\, K_{(a,b)}P_{(a,b)}$, 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 $\Sigma\, K_{(a,b)}$ 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 null, the neighbor $P_{(a,b)}$ does not contribute to the new value of $P_{(i,j)}$ (notice that $P_{(a,b)}$ might be $P_{(i,j)}$ itself).

If the convolution kernel is

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

then

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

If the convolution kernel is

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

then

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

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.

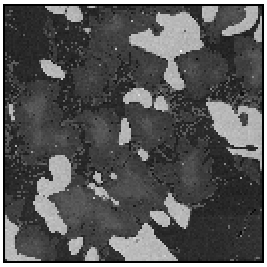
If all coefficients in the kernel are positive, the transfer function is equivalent to a weighted summation and produces a smoothing or lowpass filter. Typical lowpass filters include smoothing and Gaussian filters.

Gradient Filter

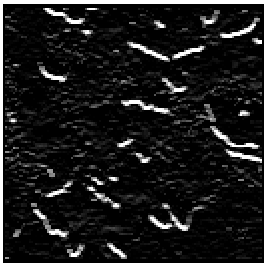
A *gradient filter* highlights the variations of light intensity along a specific direction, which has the effect of outlining edges and revealing texture.

Example

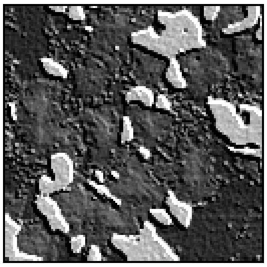
This example uses the following source image.



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 and its kernel uses the following model:

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

where *a*, *b*, and *c* 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:

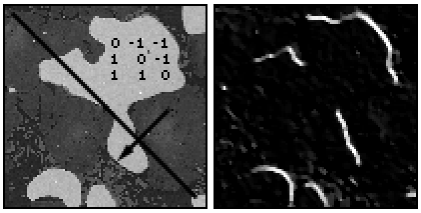
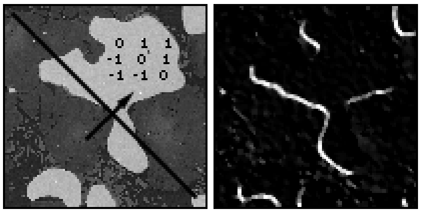
$$\begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix}$$

The axis of symmetry is at 135 degrees.

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 degrees.

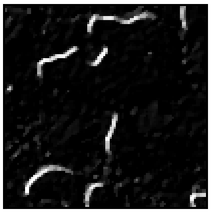
Gradient #1	Gradient #2
$\begin{matrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{matrix}$	$\begin{matrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{matrix}$
Gradient #1 highlights pixels where the light intensity increases along the direction going from northeast to southwest. It darkens pixels 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.	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.
	

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

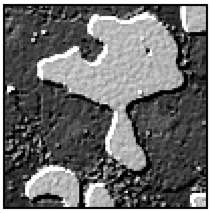
Edge Extraction and Edge Highlighting

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

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

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

- 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:

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

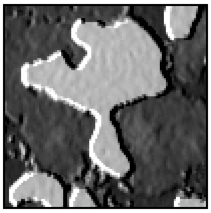
Note The convolution filter using the second kernel on the right side of the equation reproduces the source image. All neighboring pixels are multiplied by 0 and the central pixel remains equal to itself: ($P_{(i, j)} = 1 \times P_{(i, j)}$).

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

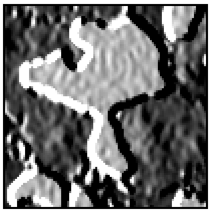
Gradient #2 = Gradient #1 + Source Image

Edge Thickness

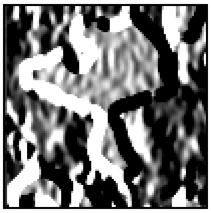
The larger the kernel, the larger the edges. The following image illustrates gradient west–east 3 × 3.



The following image illustrates gradient west–east 5 × 5.



Finally, the following image illustrates gradient west–east 7 × 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 5-1. Prewitt Filters

W/Edge	W/Image	SW/Edge	SW/Image
–1 0 1	–1 0 1	0 1 1	0 1 1
–1 0 1	–1 1 1	–1 0 1	–1 1 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 0 0	0 1 0	1 0 –1	1 1 –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 0 –1	1 1 –1	1 0 –1	1 1 –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 0 0	0 1 0	–1 0 1	–1 1 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 5-2. Sobel Filters

W/Edge	**W/Image**	**SW/Edge**	**SW/Image**
–1 0 1	–1 0 1	0 1 2	0 1 2
–2 0 2	–2 1 2	–1 0 1	–1 1 1
–1 0 1	–1 0 1	–2 –1 0	–2 –1 0
S/Edge	**S/Image**	**SE/Edge**	**SE/Image**
1 2 1	1 2 1	2 1 0	2 1 0
0 0 0	0 1 0	1 0 –1	1 1 –1
–1 –2 –1	–1 –2 –1	0 –1 –2	0 –1 –2
E/Edge	**E/Image**	**NE/Edge**	**NE/Image**
1 0 –1	1 0 –1	0 –1 –2	0 –1 –2
2 0 –2	2 1 –2	1 0 –1	1 1 –1
1 0 –1	1 0 –1	2 1 0	2 1 0
N/Edge	**N/Image**	**NW/Edge**	**NW/Image**
–1 –2 –1	–1 –2 –1	–2 –1 0	–2 –1 0
0 0 0	0 1 0	–1 0 1	–1 1 1
1 2 1	1 2 1	0 1 2	0 1 2

The following tables list the predefined gradient 5 × 5 and 7 × 7 kernels.

Table 5-3. Gradient 5 × 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	0	2	1	–1	–2	1	2	1
–1	–2	0	2	1	–1	–2	0	2	1
0	–1	0	1	0	0	–1	0	1	0

Table 5-4. Gradient 7 × 7

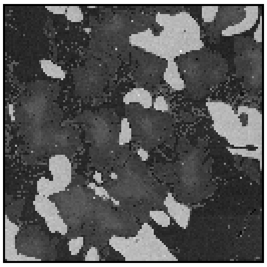
W/Edge							W/Image						
0	–1	–1	0	1	1	0	0	–1	–1	0	1	1	0
–1	–2	–2	0	2	2	1	–1	–2	–2	0	2	2	1
–1	–2	–3	0	3	2	1	–1	–2	–3	0	3	2	1
–1	–2	–3	0	3	2	1	–1	–2	–3	1	3	2	1
–1	–2	–3	0	3	2	1	–1	–2	–3	0	3	2	1
–1	–2	–3	0	3	2	1	–1	–2	–3	0	3	2	1
0	–1	–1	0	1	1	0	0	–1	–1	0	1	1	0

Laplacian Filters

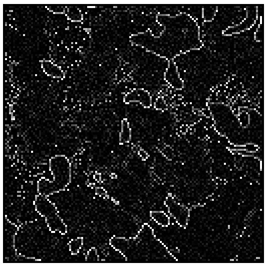
A *Laplacian filter* highlights the variation of the light intensity surrounding a pixel. The filter extracts the contour of objects and outlines details. Unlike the gradient filter, it is omnidirectional.

Example

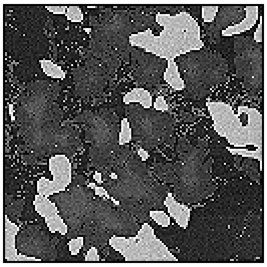
This example uses the following source image.



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:

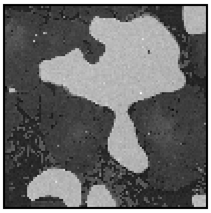
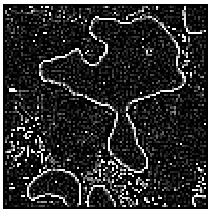
$$x \geq 2(|a| + |b| + |c| + |d|)$$

Contour Extraction and Highlighting

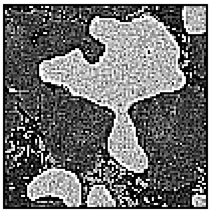
If the central coefficient is equal to this sum $(x = 2(|a| + |b| + |c| + |d|))$, the Laplacian filter extracts the pixels where significant variations of light intensity are found. The presence of sharp edges, boundaries between objects, modification in the texture of a background, noise, and other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

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

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

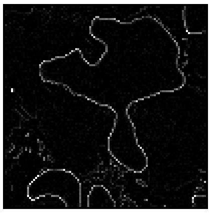
This equation indicates that the Laplacian #2 kernel adds the contours extracted by the Laplacian #1 kernel to the source image.

Laplacian #2 = Laplacian #1 + Source Image

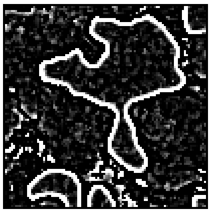
For example, if the central coefficient of Laplacian #2 kernel is 10, the Laplacian filter adds the contours extracted by Laplacian #1 kernel to the source image times 2, and so forth. A greater central coefficient corresponds to less-prominent contours and details highlighted by the filter.

Contour Thickness

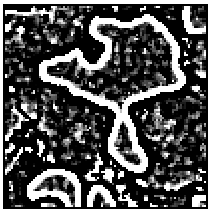
Larger kernels correspond to larger contours. The following image is a Laplacian 3 × 3.



The following image is a Laplacian 5 × 5.



The following image is a Laplacian 7 × 7.



Predefined Laplacian Kernels

The following tables list the predefined Laplacian kernels.

Table 5-5. Laplacian 3 × 3

Contour 4	+ Image × 1	+ Image × 2
0 –1 0	0 –1 0	0 –1 0
–1 4 –1	–1 5 –1	–1 6 –1
0 –1 0	0 –1 0	0 –1 0
Contour 8	**+ Image × 1**	**+ Image × 2**
–1 –1 –1	–1 –1 –1	–1 –1 –1
–1 8 –1	–1 9 –1	–1 10 –1
–1 –1 –1	–1 –1 –1	–1 –1 –1
Contour 12	**+ Image × 1**	
–1 –2 –1	–1 –2 –1	
–2 12 –2	–2 13 –2	
–1 –2 –1	–1 –2 –1	

Table 5-6. Laplacian 5 × 5

Contour 24	+ Image × 1
–1 –1 –1 –1 –1	–1 –1 –1 –1 –1
–1 –1 –1 –1 –1	–1 –1 –1 –1 –1
–1 –1 24 –1 –1	–1 –1 25 –1 –1
–1 –1 –1 –1 –1	–1 –1 –1 –1 –1
–1 –1 –1 –1 –1	–1 –1 –1 –1 –1

Table 5-7. Laplacian 7 × 7

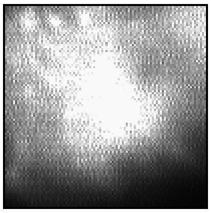
Contour 48	+ Image × 1
–1 –1 –1 –1 –1 –1 –1	–1 –1 –1 –1 –1 –1 –1
–1 –1 –1 –1 –1 –1 –1	–1 –1 –1 –1 –1 –1 –1
–1 –1 –1 –1 –1 –1 –1	–1 –1 –1 –1 –1 –1 –1
–1 –1 –1 48 –1 –1 –1	–1 –1 –1 49 –1 –1 –1
–1 –1 –1 –1 –1 –1 –1	–1 –1 –1 –1 –1 –1 –1
–1 –1 –1 –1 –1 –1 –1	–1 –1 –1 –1 –1 –1 –1
–1 –1 –1 –1 –1 –1 –1	–1 –1 –1 –1 –1 –1 –1

Smoothing Filter

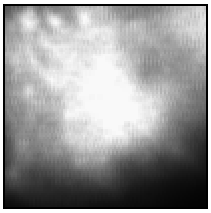
A *smoothing filter* attenuates the variations of light intensity in the neighborhood of a pixel. It smooths the overall shape of objects, blurs edges, and removes details.

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:

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

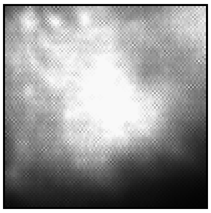
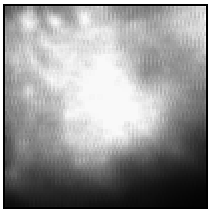
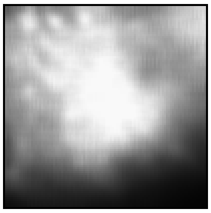
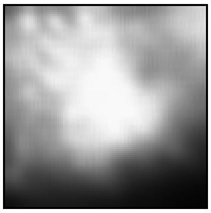
where a, b, c, and d are integers and x = 0 or 1.

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.

Kernel #1	Filtered Image
$\begin{matrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{matrix}$	
Kernel #2	**Filtered Image**
$\begin{matrix} 2 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 2 \end{matrix}$	
Kernel #3	**Filtered Image**
$\begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$	
Kernel #4	**Filtered Image**
$\begin{matrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{matrix}$	

Predefined Smoothing Kernels

The following tables list the predefined smoothing kernels.

Table 5-8. Smoothing 3 × 3

0	1	0	0	1	0	0	2	0	0	4	0
1	0	1	1	1	1	2	1	2	4	1	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	0	1	1	1	1	2	1	2	4	1	4
1	1	1	1	1	1	2	2	2	4	4	4

Table 5-9. Smoothing 5 × 5

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	0	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

Table 5-10. Smoothing 7 × 7

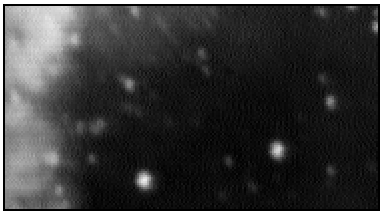
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Gaussian Filters

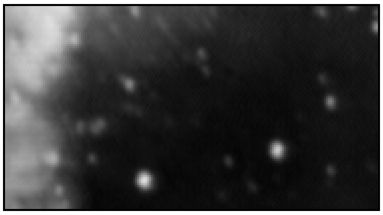
A *Gaussian filter* attenuates the variations of light intensity in the neighborhood of a pixel. It smooths the overall shape of objects and attenuates details. It is similar to a smoothing filter, but its blurring effect is more subdued.

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 $x > 1$.

Since 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 5-11. Gaussian 3 × 3

0	1	0		0	1	0		1	1	1
1	2	1		1	4	1		1	2	1
0	1	0		0	1	0		1	1	1
1	1	1		1	2	1		1	4	1
1	4	1		2	4	2		4	16	4
1	1	1		1	2	1		1	4	1

Table 5-12. Gaussian 5 × 5

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

Table 5-13. Gaussian 7 × 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	16	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 × 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 × 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

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

Kernel #2

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

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

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:

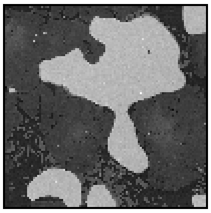
Kernel #1				**Kernel #2**		
–1	0	1		–1	–2	–1
–2	0	2		0	0	0
–1	0	1		1	2	1

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

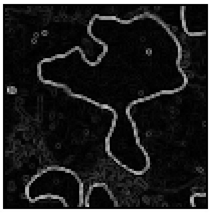
$$P_{(i,j)} = \max[|P_{(i+1,j-1)} - P_{(i-1,j-1)} + \mathbf{2}P_{(i+1,j)} - \mathbf{2}P_{(i-1,j)} + P_{(i+1,j+1)} - P_{(i-1,j+1)}|, \\ |P_{(i-1,j+1)} - P_{(i-1,j-1)} + \mathbf{2}P_{(i,j+1)} - \mathbf{2}P_{(i,j-1)} + P_{(i+1,j+1)} - P_{(i+1,j-1)}|]$$

Example

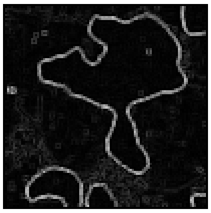
This example uses the following source image.



A nonlinear Prewitt filter produces the following image.



A nonlinear Sobel filter produces the following image.



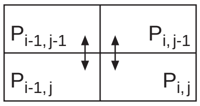
Both filters outline the contours of the objects. Because of the different convolution kernels they combine, the nonlinear Prewitt has the tendency to outline curved contours while the nonlinear Sobel extracts square contours. This difference is noticeable when observing the outlines of isolated pixels.

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.

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

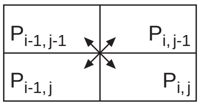


Roberts Filter

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.

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

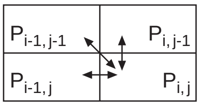


Differentiation Filter

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.

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

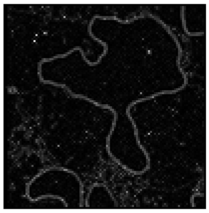


Sigma Filter

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.

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 $[M - S, M + S]$.

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

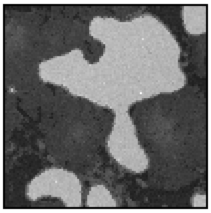


Lowpass Filter

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.

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 $[M - S, M + S]$.

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



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.

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

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 × 3 neighborhood:

$$\begin{array}{|c|c|c|}\hline P_{(i-1,j-1)} & P_{(i,j-1)} & P_{(i+1,j-1)} \\ \hline P_{(i-1,j)} & P_{(i,j)} & P_{(i+1,j)} \\ \hline P_{(i-1,j+1)} & P_{(i,j+1)} & P_{(i+1,j+1)} \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline 13 & 10 & 9 \\ \hline 12 & 4 & 8 \\ \hline 5 & 5 & 6 \\ \hline \end{array}$$

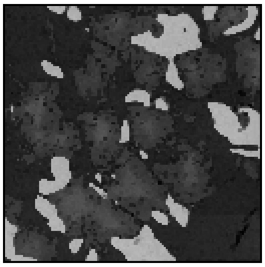
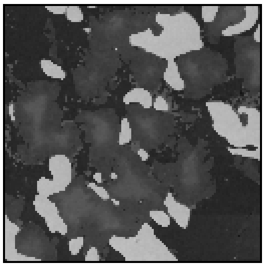
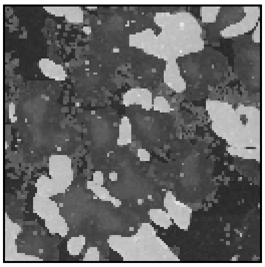
The following table shows the new output value of the central pixel for each *N*th order value:

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

Note 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 Nth order filter, notice the example of an image with bright objects and a dark background. When viewing this image with the B&W (or 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 × 3	
• If $N < (f^2 - 1)/2$, the Nth order filter has the tendency to erode bright regions (or dilate dark regions). • If $N = 0$, each pixel is replaced by its local minimum.	Order 0 (smooths image, erodes bright objects)	
• If $N = (f^2 - 1)/2$, 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.	Order 4 (equivalent to a median filter)	
• If $N > (f^2 - 1)/2$, the Nth order filter has the tendency to dilate bright regions (or erode dark regions). • If $N = f^2 - 1$, each pixel is replaced by its local maximum.	Order 8 (smooths image, dilates bright objects)	

6

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.

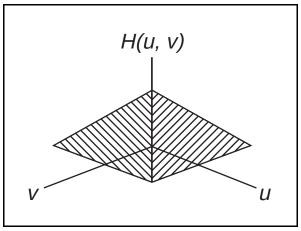
$$\begin{tikzcd} \boxed{f(x, y)} \arrow[r, "FFT"] & \boxed{F(u, v)} \arrow[r, "Filter"] & \boxed{H(u, v)} \arrow[r, "Inverse\ FFT"] & \boxed{g(x, y)} \end{tikzcd}$$

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

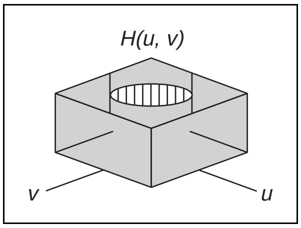
Lowpass FFT Filters

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.



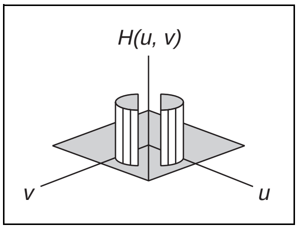
Highpass FFT Filters

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 filter* removes frequencies contained in a mask specified by the user. Depending on the mask definition, this filter may 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

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

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), note 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 (Δ x, Δ y) and the frequency domain (Δ u, Δ v) is

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

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

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

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

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

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

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

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

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

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

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

The FFT of an image has the following two properties:

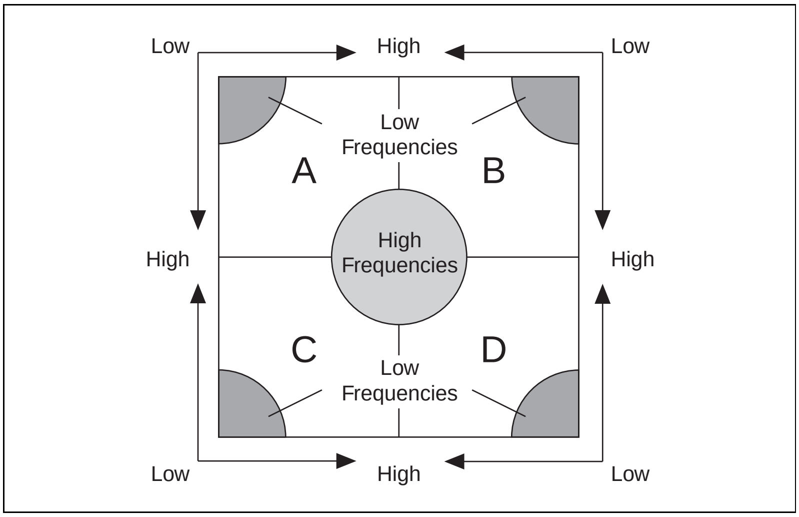
- It is periodic: $F(u, v) = F(u + N, v + M)$
- It is conjugate-symmetric: $F(u, v) = F^*(-u, -v)$

These properties result in two possible representations of the Fast Fourier Transform of an image: the *standard representation* and the *optical representation*.

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

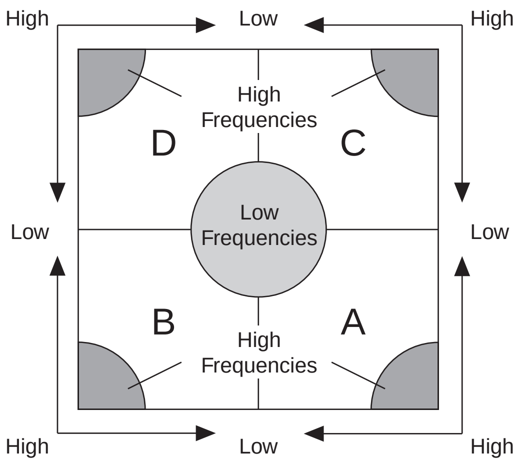
$$0, N\Delta u] \times [0, M\Delta v$$



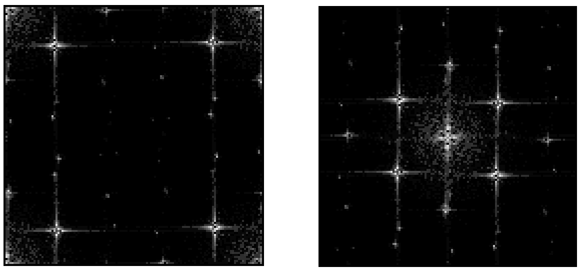
Optical Representation

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 the standard representation to the optical representation by permuting the *A*, *B*, *C*, and *D* quarters.



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

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, the Inverse

FFT command produces an image in which noise, details, texture, and sharp edges are smoothed.

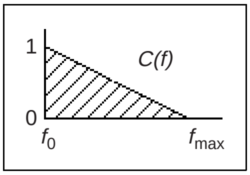
A lowpass frequency filter removes or attenuates spatial frequencies located outside a frequency range centered on the fundamental (or null) frequency.

Lowpass Attenuation

Lowpass attenuation applies a linear attenuation to the full frequency range, decreasing from f_0 to $f_{\max}$. 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 $C(f_{\max}) = 0$



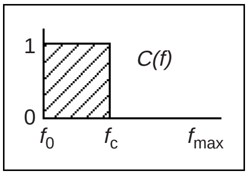
Lowpass Truncation

Lowpass truncation removes a frequency f if it falls outside the truncation range $[f_0, f_c]$. 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.

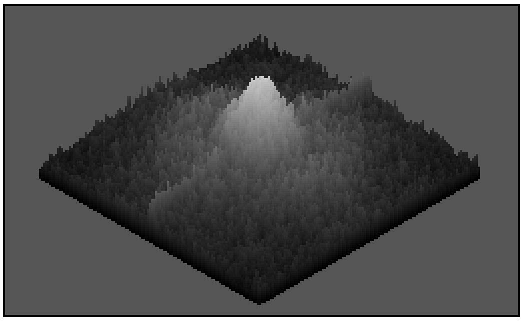
If $f > f_c$

then $C(f) = 0$

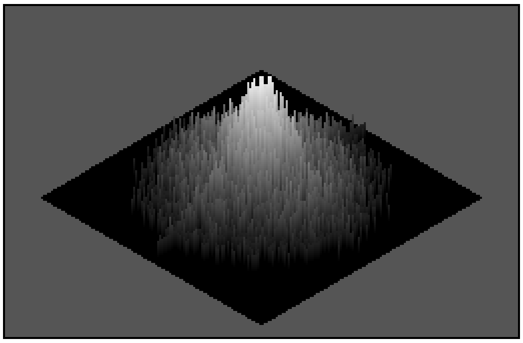
else $C(f) = 1$



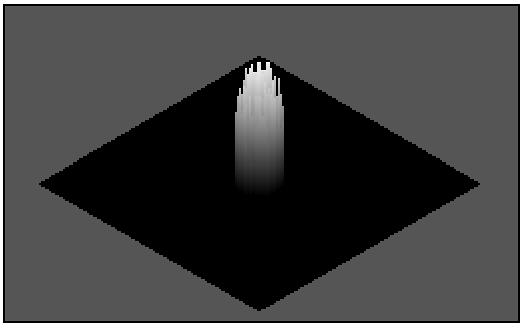
The following series of graphics illustrates the behavior of each type of filter. They give the 3D-view profile of the magnitude of the FFT. This example uses the following original FFT.



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

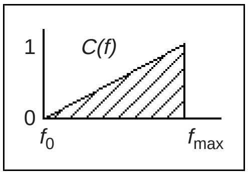
A *highpass frequency filter* attenuates or removes low frequencies present in the FFT plane. It has the effect of suppressing information related to slow variations of light intensities in the spatial image. In this case, the inverse FFT produces an image in which overall patterns are attenuated and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

Highpass Attenuation

Highpass attenuation applies a linear attenuation to the full frequency range, decreasing from f_{max} to 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 $C(f_{\text{max}}) = 0$.



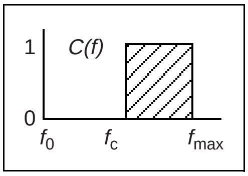
Highpass Truncation

Highpass truncation removes a frequency f if it falls inside the truncation range $[f_0, f_c]$. 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.

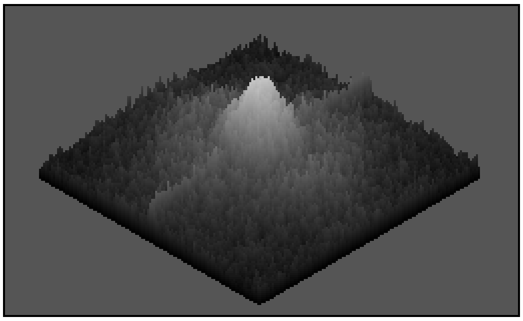
If $f < f_c$

then $C(f) = 0$

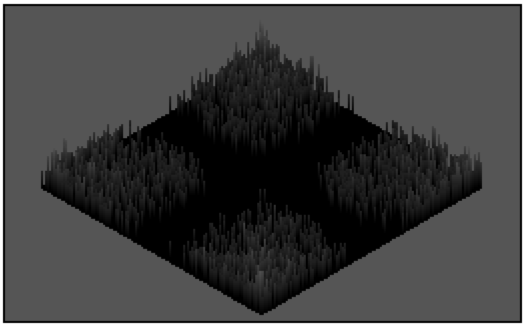
else $C(f) = 1$



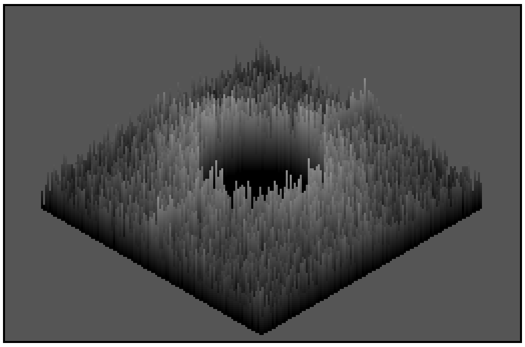
The following series of graphics illustrates the behavior of each type of filter. They give the 3D-view profile of the magnitude of the FFT. This example uses the following original FFT image.



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.



7 Morphology Analysis

This chapter provides an overview of morphology image analysis.

Morphological transformations extract and alter the structure of objects in an image. You can use these transformations to prepare objects for quantitative analysis, observe the geometry of regions, extract the simplest forms for modeling and identification purposes, and so forth.

The morphological transformations can be used for expanding or reducing objects, filling holes, closing inclusions, smoothing borders, removing dendrites, and more. They fall into two categories:

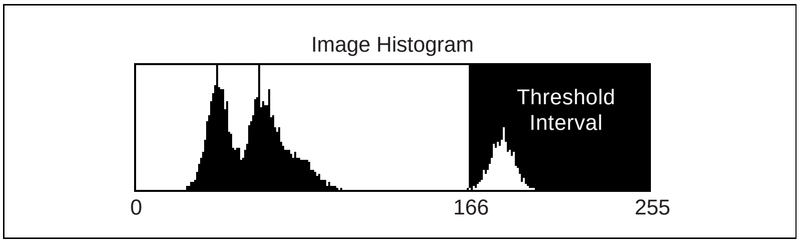
- *Gray-level morphology* functions, which apply to gray-level images
- *Binary Morphology* functions, which apply to binary images

A *binary image* is an image that has been segmented into an object region (pixels equal to 1) and a background region (pixels equal to 0). Such an image is generated by the *thresholding* process.

Thresholding

Thresholding consists of segmenting an image into two regions: an object region and a background region. This is performed by setting to 1 all pixels that belong to a gray-level interval, called the threshold interval. All other pixels in the image are set to 0.

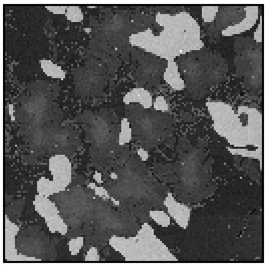
You can use this operation to extract areas that correspond to significant structures in an image and to focus the analysis on these areas.



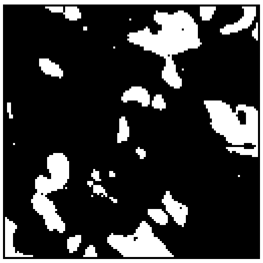
Pixels outside the threshold interval are set to 0 and considered as part of the background area. Pixels inside the threshold interval are set to 1 and considered as part of an object area.

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 an object 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 prior to thresholding, so as to outline where the correct borders lie. Observing the intensity profile of a line crossing a boundary area can also be helpful in selecting a correct threshold value. Finally, keep in mind that morphological transformations can help you retouch the shape of binary objects 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. The final binary image is the intersection of the three binary images obtained by thresholding each color component separately.

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 minimal and maximal 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 region you want to threshold is black and the background you want to eliminate is red (when viewing with a binary palette).

Clustering is the only multi-class thresholding method available. Clustering operates on multiple classes so you can create tertiary or even 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

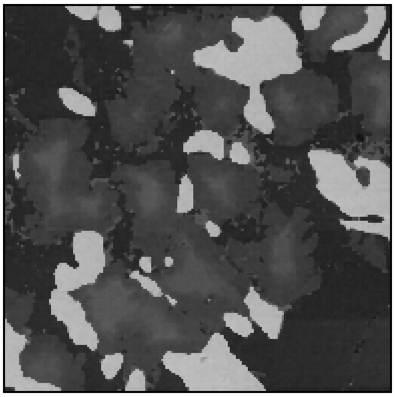
In this rapid technique, the image is sorted randomly within a discrete number of classes corresponding to the number of phases perceived in an image. The gray values are determined and a *barycenter* is determined for each class. This process is repeated until a value is obtained that represents the center of mass for each phase or class.

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

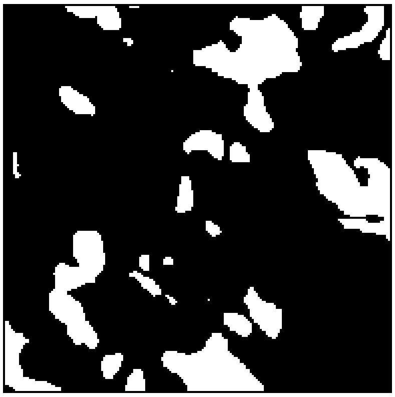
Example of Clustering

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.

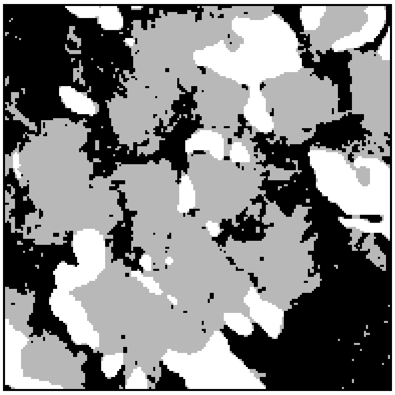
This example uses the following original image.



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 objects that are present in minuscule proportions on the image. For example, this function would be suitable for default detection.

Metric

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

Moments

This technique is suited for images that have poor contrast (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.

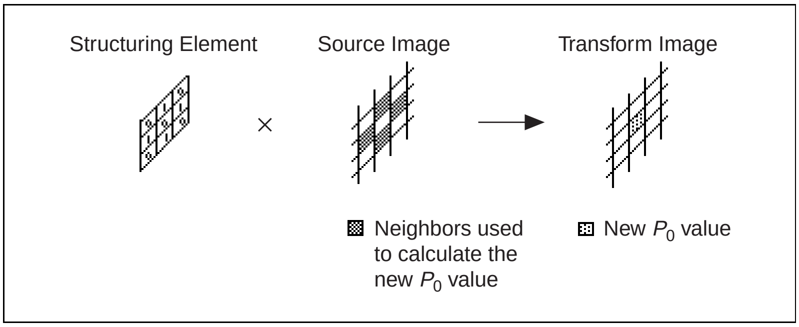
Interclass Variance

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 five percent 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 objects.

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.

The following graphic illustrates a morphological transformation using a structuring element. This example uses a 3 × 3 image that has a rectangular frame.

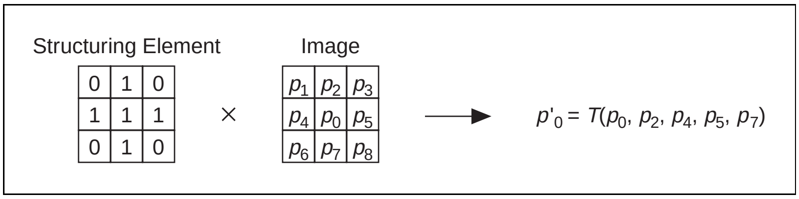


Figure 7-1. Rectangular Frame, Neighborhood 3 × 3

The next graphic illustrates a morphological transformation using a structuring element for an image that has a hexagonal frame. This example uses a 5 × 3 image.

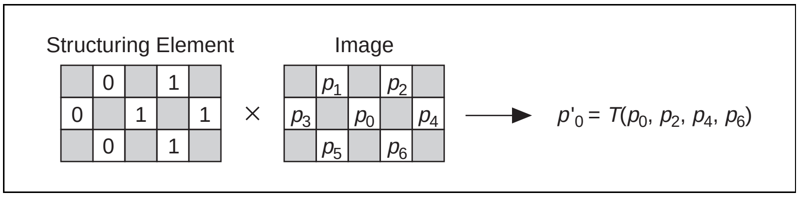


Figure 7-2. Hexagonal Frame, Neighborhood 5 × 3

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

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

Primary Binary Morphology Functions

The *primary morphology* functions apply to binary images in which objects 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 **object** denotes a group of pixels equal to 1.

Erosion Function

An *erosion* eliminates pixels isolated in the background and erodes the contour of objects 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. In the example of a structuring element 3 × 3, the P_i can range from P_0 itself to P_8.

1. 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.
2. If 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 objects is equivalent to eroding the background. This function eliminates tiny holes isolated in objects and expands the contour of the objects with respect to the template defined by the structuring element.

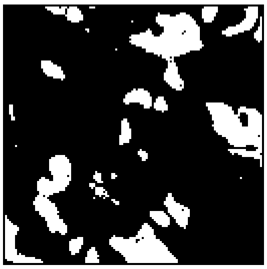
Concept and Mathematics

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. In the example of a structuring element 3 × 3, the P_i can range from P_0 itself to P_8.

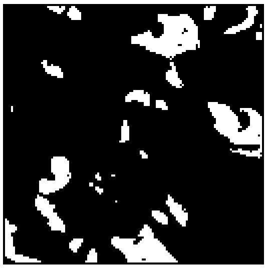
1. If the value of one pixel P_i is equal to 1, *then* P_0 is set to 1, *else* P_0 is set to 0.
2. If OR(P_i) = 1, *then* $P_0 = 1$, *else* $P_0 = 0$.

Erosion and Dilation Examples

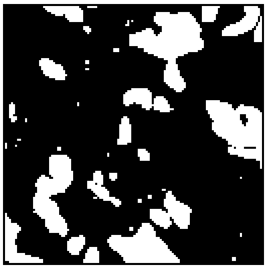
This example uses the following binary source image.



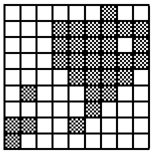
The erosion function produces the following image.



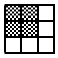
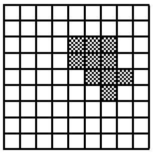
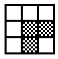
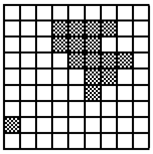
The dilation function produces the following image.

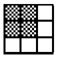
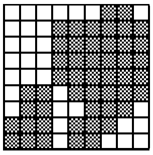
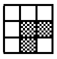
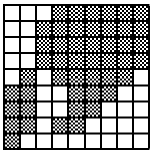


The next example uses the following source image. Gray cells indicate pixels equal to 1.



The following tables show how the structuring element can be used to control the effect of an erosion or a dilation. The larger the structuring element, the more templates can be edited and the more selective the effect.

Structuring Element	After Erosion	Description
		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 objects.
		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 objects, but retains the corners.

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 objects.
		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 objects.

Opening Function

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

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

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

Closing Function

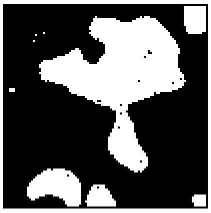
The *closing function* is a dilation followed by an erosion. It fills tiny holes and smooths boundaries. If *I* is an image,

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

This operation does not significantly alter the area and shape of objects 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.

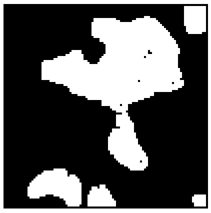
Opening and Closing Examples

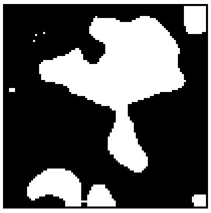
The following series of graphics illustrate examples of openings and closings.


Original Image

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

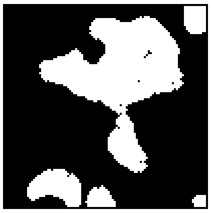
Structuring Element


After Opening


After Closing

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

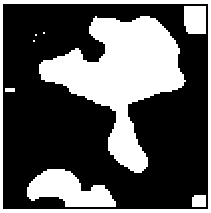
Structuring Element



After Opening

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

Structuring Element



After Closing

External Edge 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,

$$\textit{external edge}(I) = \textit{dilation}(I) - I = \mathrm{XOR}(I, \textit{dilation}(I))$$

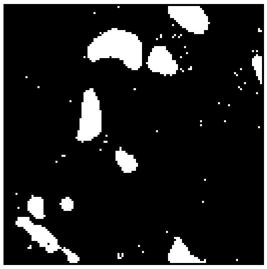
Internal Edge Function

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,

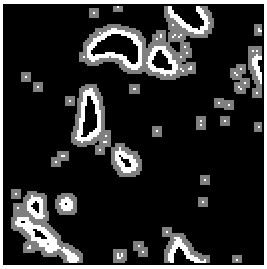
$$\textit{internal edge}(I) = I - \textit{erosion}(I) = \mathrm{XOR}(I, \textit{dilation}(I))$$

External and Internal Edge Example

This example uses the following binary source image.



Extraction using a 5 × 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 be used to locate single isolated pixels, cross-shape or longitudinal patterns, right angles along the edges of objects, and other user-specified shapes. The larger the size of the structuring element, the more specific the researched template can be.

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. In the example of a structuring element 3 × 3, the P_i range from P_0 to P_8.

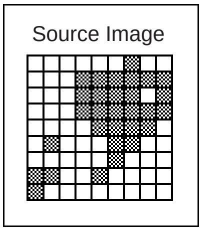
If the value of each pixel P_i is equal to the coefficient of the structuring element placed on top of it, *then* the pixel P_0 is set to 1, *else* the pixel P_0 is set to 0.

In other words, *if* the pixels P_i define the exact same template as the structuring element, *then* P_0 is set to 1, *else* P_0 is set to 0.

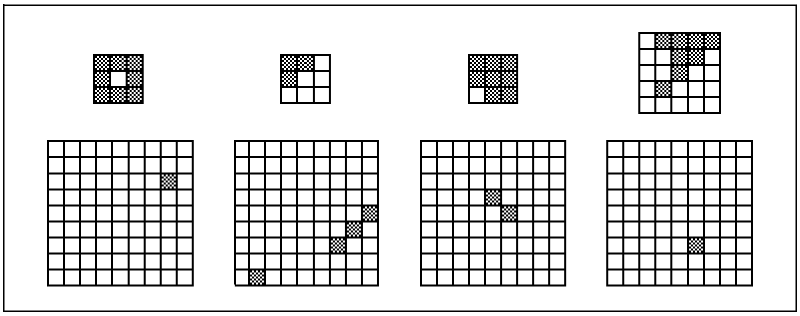
A hit-miss function using a structuring element with a central coefficient equal to 0 changes all pixels set to 1 in the source image to the value 0.

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 be used to locate pixels surrounded by various patterns specified through the structuring element.

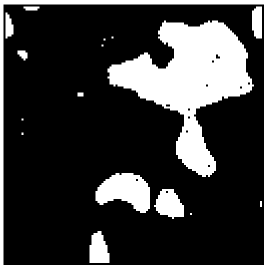
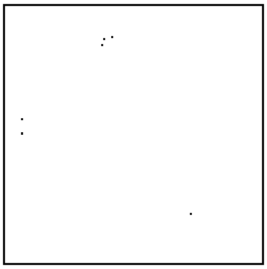
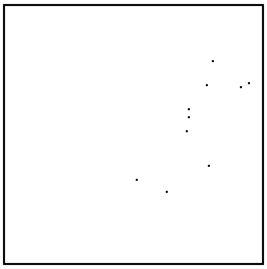
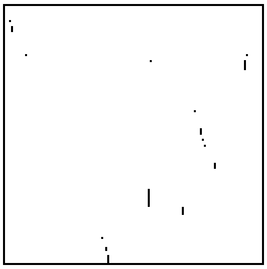


Table 7-1. Using the Hit-Miss Function

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

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 objects. 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.

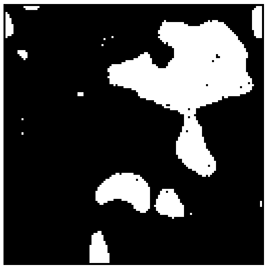
If I is an image,

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

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

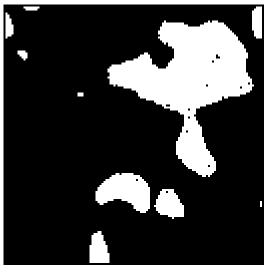
This example uses the following binary source image.



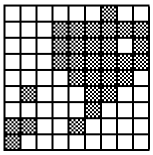
This example uses the thinning function and the following structuring element:

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

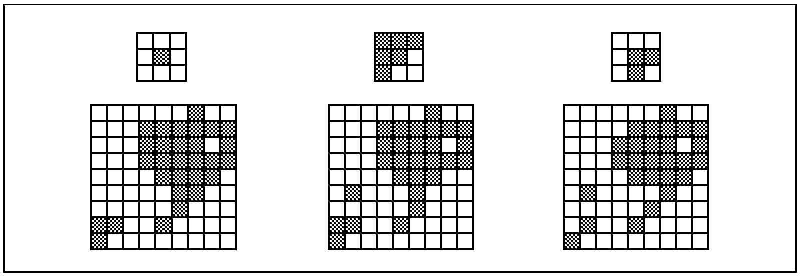
Thinning produces the following image. Single pixels in the background of this image have been removed.



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 objects, and so forth. The larger the size of the structuring element, the more specific the template can be.

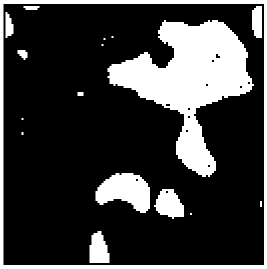
The thickening function extracts the union between a source image and its transformed image after a hit-miss function that uses the structuring element specified for the thickening. In binary terms, the operation adds a hit-miss transformation to a source image. If I is an image,

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

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 objects from 1 to 0. However, the addition of the thickening function resets these pixels to 1 anyway.

Examples

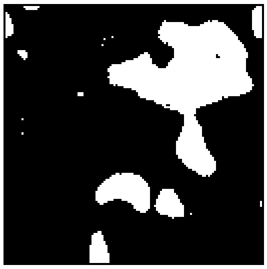
This example uses the following binary source image.



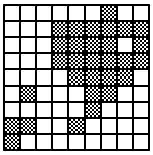
Thickening using the structuring element

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

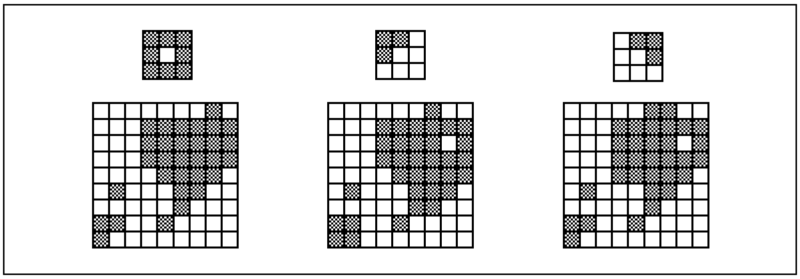
produces the following image. Single pixel holes are filled.



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 objects with respect to the template defined by the structuring element.

If I is the source image, the proper-opening 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.

$$\textit{proper-opening}(I) = \mathrm{AND}(I, OCO(I)) \text{ or}$$
$$\textit{proper-opening}(I) = \mathrm{AND}(I, DEEDDE(I))$$

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 objects with respect to the template defined by the structuring element.

If *I* is the source image, the proper-closing 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.

$$\textit{proper-closing}(I) = \mathrm{OR}(I, COC(I)) \text{ or}$$
$$\textit{proper-closing}(I) = \mathrm{OR}(I, EDDEED(I))$$

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 objects 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*.

$$\textit{auto-median}(I) = \mathrm{AND}(OCO(I), COC(I)) \text{ or}$$
$$\textit{auto-median}(I) = \mathrm{AND}(DEEDDE(I), EDDEED(I))$$

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 objects 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 **object** denotes a group of pixels equal to 1.

Border Function

The *border function* removes objects that touch the border of the image. These objects may 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 objects.

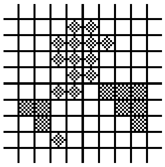
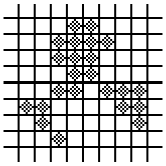
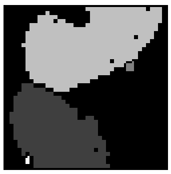
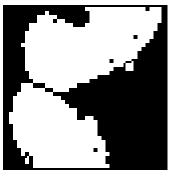
Labeling Function

The *labeling function* assigns a different gray-level value to each object. The image produced is not a binary image, but a labeled image using a number of gray-level values equal to the number of objects in the image plus the gray level 0 used in the background area.

The labeling function can identify objects using connectivity-4 or connectivity-8 criteria.

Lowpass Filters

The *lowpass filter* removes small objects with respect to their width (specified by a parameter called *filter size*).

	Connectivity-4	**Connectivity-8**
Definition Two pixels are considered as part of the same object if they are horizontally or vertically adjacent.	The pixels are considered as part of two different objects if they are diagonally adjacent.	The pixels are considered as part of the same object if they are horizontally, vertically, or diagonally adjacent.
Illustration For a same pixel pattern, different sets of objects can be identified.		
Example		

For a given filter size *N*, the lowpass filter eliminates objects with a width less than or equal to (*N* – 1) pixels. These objects are those that would disappear after (*N* – 1)/2 erosions.

Highpass Filters

The *highpass filter* removes large objects with respect to their width (specified by a parameter called filter size).

For a given filter size *N*, the highpass filter eliminates objects with a width greater than or equal to *N* pixels. These objects are those that would not disappear after (*N*/2 + 1) erosions.

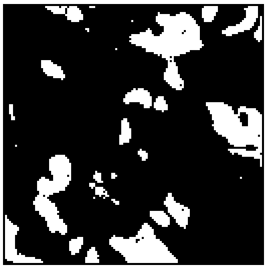
Both the highpass and lowpass morphological filters use erosions to determine if an object is to be removed. Therefore, they cannot discriminate objects with a width of 2*k* pixels from objects with a width of 2*k* – 1 pixels. For example, one erosion eliminates both objects that are two pixels wide and one pixel wide.

The precision of the filters then depends on the parity of the filter size *N*.

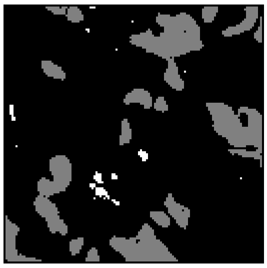
	Highpass Filter	**Lowpass Filter**
***If N* is an even number (*N* = 2*k*)**	• removes objects with a width greater than or equal to 2*k* • uses *k* – 1 erosions	• removes objects with a width less than or equal to 2*k* – 2 • uses *k* – 1 erosions
***If N* is an odd number (*N* = 2*k* + 1)**	• removes objects with a width greater than or equal to 2*k* + 1 • uses *k* erosions	• removes objects with a width less than or equal to 2*k* • uses *k* erosions

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 objects and white objects are filtered out by a lowpass and highpass filter, respectively.



Separation Function

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

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

For a given filter size N, the separation function segments objects having a narrowing shorter than or equal to $(N – 1)$ pixels. These objects 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 two pixels wide and a narrowing that is one 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:

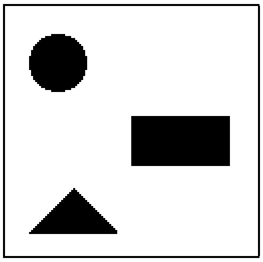
- If *N* is an even number (2*k*), the separation breaks a narrowing with a width smaller than or equal to (2*k* – 2) pixels. It uses (*k* – 1) erosions.
- If *N* is an odd number (2*k* + 1), the separation breaks a narrowing with a width smaller than or equal to 2*k*. It uses *k* erosions.

Skeleton Functions

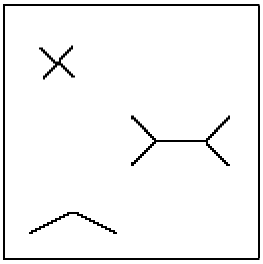
A *skeleton function* applies a succession of thinnings until the width of each object becomes equal to one pixel. The skeleton functions are both time- and memory-consuming. They are based on conditional applications of thinnings and openings using various configurations of structuring elements.

L-Skeleton Function

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

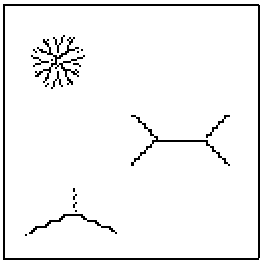


The L-skeleton function produces the following rectangle pixel frame image.



M-Skeleton Function

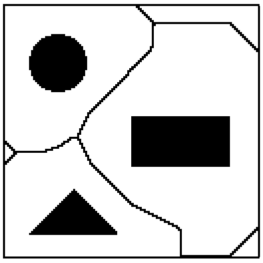
The *M-skeleton* (M-shaped structuring element) function extracts a skeleton with more dendrites or branches. Using the same original image as in the previous example, the M-skeleton function produces the following image.



Skiz Function

The *skiz* (skeleton of influence zones) function behaves like an L-skeleton applied to the background regions, instead of the object regions. It produces median lines that are at an equal distance from the objects.

Using the same source image as in the previous example, the skiz function produces the following image (shown after superimposition on top of the source image).

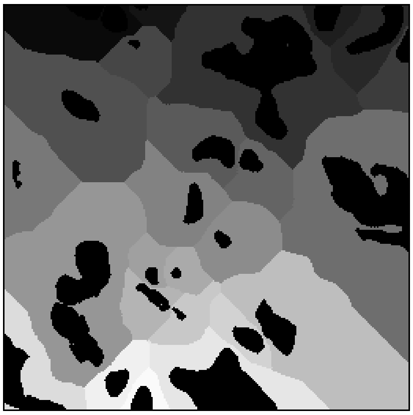


Segmentation Function

The *segmentation function* is applied only to labeled images. It partitions an image into segments, each centered on an object, such that they do not overlap each other or leave empty zones. This result is obtained by dilating objects 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 objects (shown in black) are superimposed on top of the segments (shown in gray shades).



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

Comparisons Between Segmentation and Skiz Functions

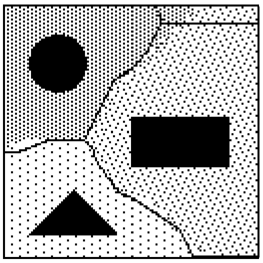
The segmentation function extracts segments that each contain one object and represent the area in which this object can be moved without intercepting another object (assuming that all objects move at the same speed).

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

Segments are obtained by successive dilations of objects until they touch each other and cover the entire image. The final image contains as many segments as there were objects 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 objects originally present in the image. Notice the upper-right region in the following example.

The following image shows:

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



Distance Function

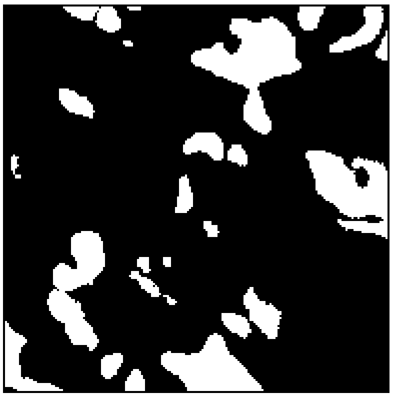
The *distance function* assigns to each pixel a gray-level value equal to the shortest distance to the border of the object. That distance may be equal to the distance to the outer border of the object or to a hole within the object.

Danielsson Function

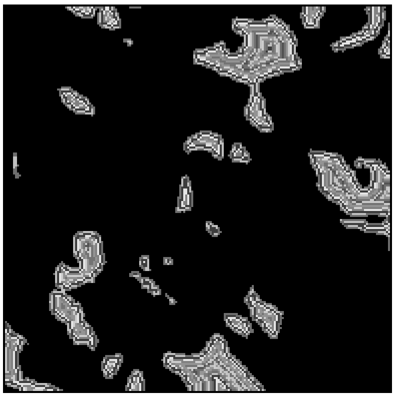
The *Danielsson function* also creates a distance map but is a more accurate algorithm than the classical distance function. Use the Danielsson function instead of the distance function when possible.

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.



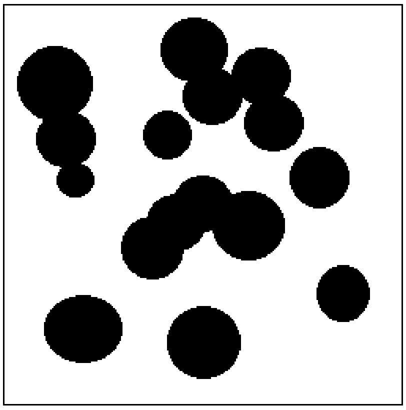
It is useful to view this final image with a binary palette. In this case, each level corresponds to a different color. The user easily can determine the relation of a set of pixels to the border of an object. The first layer (the layer that forms the border) is colored red. The second layer (the layer closest to the border) is green, the third layer is blue, and so forth.

Circle Function

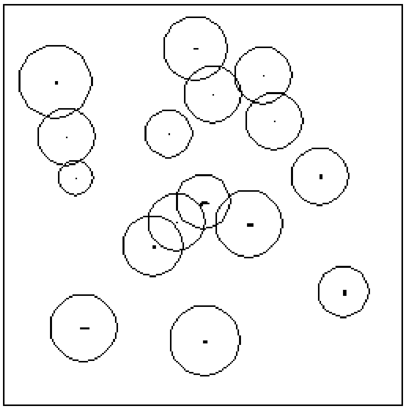
The *circle* function enables the user to separate overlapping circular objects. The circle function uses the Danielsson coefficient to reconstitute the form of an object, provided that the objects are essentially circular. The objects are treated as a set of overlapping discs that is then separated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

Example

This example uses the following source image.



The circle function produces the following processed image.



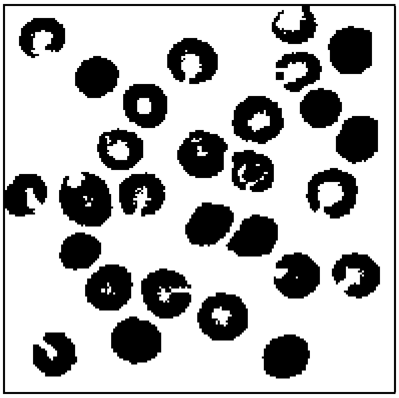
Convex Function

The *convex function* is useful for closing particles so that measurements can be made on the particle, even though the contour of the object is discontinuous. This command is usually needed in cases in which the sample object is cut because of the acquisition process.

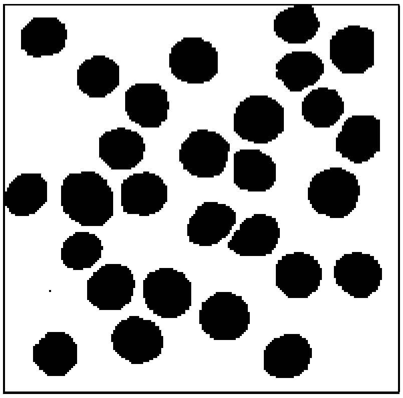
The convex function calculates a convex envelope around the perimeter of each object, effectively closing the object. The image to be treated must be both binary and labeled.

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 the 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 the 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. In the example of a 3 × 3 structuring element, P_i can range from P_0 to P_8.

$$P_0 = \min(P_i)$$

Note A gray-level erosion using a structuring element f × f with all its coefficients set to 1 is equivalent to an Nth order filter with a filter size f × f and the value N equal to 0 (refer to the nonlinear spatial filters).

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 the structuring element.

Concept and Mathematics

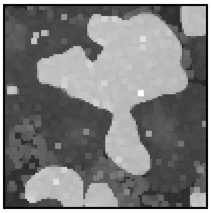
Each pixel P_0 in an image becomes equal to the maximum value of its neighbors. For a given pixel P_0, the structuring element is centered on P_0. The pixels masked by a coefficient of the structuring element equal to 1 are then referred as P_i. In the example of a structuring element 3 × 3, P_i can range from P_0 to P_8.

$$P_0 = \max(P_i)$$

Note A gray-level dilation using a structuring element f × f with all its coefficients set to 1 is equivalent to an Nth order filter with a filter size f × f and the value N equal to f × f – 1 (refer to the nonlinear spatial filters).

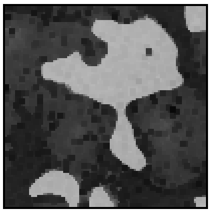
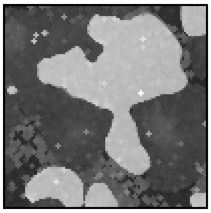
Erosion and Dilation Examples

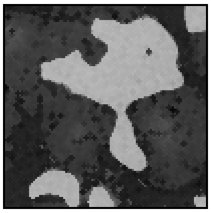
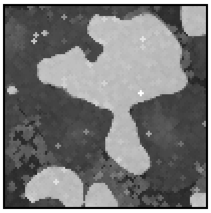
This example uses the following source image.



The following table provides example structuring elements, and the corresponding eroded and dilated images.

Structuring Element	Erosion	Dilation

1 1 1
1 1 1
1 1 1
		

0 1 0
1 1 1
0 1 0
		

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.

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

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

Closing Function

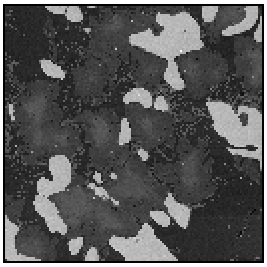
The gray-level *closing function* consists of a gray-level dilation followed by a gray-level erosion. It removes dark spots isolated in bright regions and smooths boundaries. The effects of the function are moderated by the configuration of the structuring element.

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

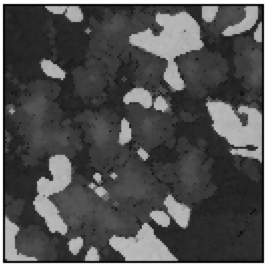
This operation does not significantly alter the area and shape of objects because dilation and erosion are morphological opposites. Bright borders expanded by the dilation are reduced by the erosion. However, small dark objects that vanish during the dilation do not reappear after the erosion.

Opening and Closing Examples

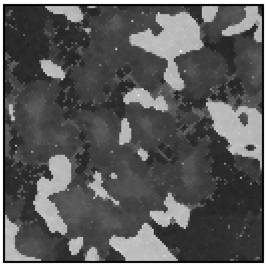
This example uses the following source image.



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 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.

$$\textit{proper-opening}(I) = \min(I, OCO\,(I)) \text{ or}$$
$$\textit{proper-opening}(I) = \min(I, DEEDDE(I))$$

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 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.

$$\textit{proper-closing}(I) = \max(I, COC(I)) \text{ or}$$
$$\textit{proper-closing}(I) = \max(I, EDDEED(I))$$

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 objects that have fewer details.

If I is the source image, the auto-median 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.

$$\mathit{auto\text{-}median}(I) = \min(\mathit{OCO}(I), \mathit{COC}(I)) \text{ or}$$
$$\mathit{auto\text{-}median}(I) = \min(\mathit{DEEDDE}(I), \mathit{EDDEED}(I))$$

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.

8

Quantitative Analysis

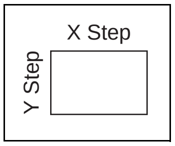
This chapter provides an overview of quantitative image analysis. The *quantitative analysis* of an image consists of obtaining *densitometry* and object measurements. Before starting this analysis, it is necessary to calibrate the image spatial dimensions and intensity scale to obtain measurements expressed in real units.

Spatial Calibration

Spatial calibration consists of correlating the area 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

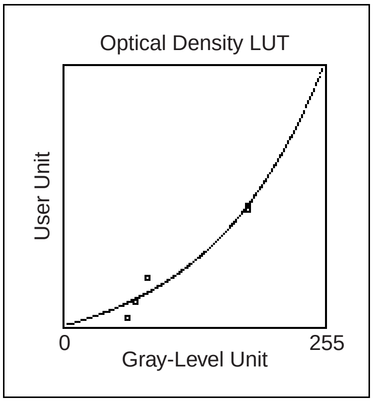
Intensity Calibration

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.



Definition of a Digital Object

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

Intensity Threshold

Objects 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 objects the Threshold Interval is [1, 1].

Connectivity

After the pixels belonging to a specified intensity threshold are identified, they are grouped into objects. 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.

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

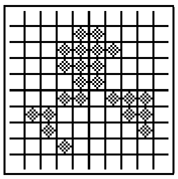
If D is the distance from P_0 to P_1, then the distances between P_0 and its eight neighbors can range from D to $\sqrt{2}D$, as shown in the following graphic.

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

Connectivity-8

A pixel belongs to an object if it is at a distance D or $\sqrt{2}\,D$ from another pixel in the object.

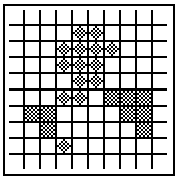
Two pixels are considered as part of the same object if they are horizontally, vertically, or diagonally adjacent. In the following image, the object count equals 1.



Connectivity-4

A pixel belongs to an object if it is at a distance *D* from another pixel in the object.

Two pixels are considered as part of the same object if they are horizontally or vertically adjacent. They are considered as part of two different objects if they are diagonally adjacent. In the following image, the object count equals 4.



Area Threshold

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

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

Examples

In the following example, 1 pixel = 1 square inch.

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

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

Object Measurements

A digital object 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 object area to the entire image area
- **Number of holes**—Number of holes within the object
- **Holes’ area**—Total area of the holes
- **Total area**—Area of the object including its holes’ area (equals **Particle Area + Holes’ Area**)

Particle Number

Identification number assigned to an object. Particles are numbered starting from 1 in increasing order from the upper-left corner of the image to the lower-right corner.

Number of Pixels

Number of pixels in an object. This value gives the area of an object, without holes, in pixel units.

Particle Area

Area of an object expressed in real units. This value is equal to **Number of pixels** when the spatial calibration is such that one pixel represents one 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 objects.

$$\text{Ratio} = \frac{\text{particle area}}{\text{scanned area}}$$

Number of Holes

Number of holes inside an object. The software detects holes inside an object as small as 1 pixel.

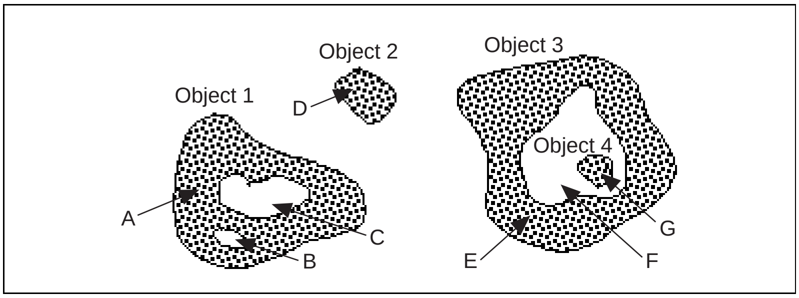
Holes' Area

Total area of the holes within an object.

Total Area

Area of an object including the area of its holes. This value is equal to (**Particle Area** + **Holes' Area**).

Note An object located inside a hole of a bigger object is identified as a separate object. The area of a hole that contains an object includes the area covered by the object.



Object #	Particle Area	Holes’ Area	Total Area
Object 1	A	B + C	A + B + C
Object 2	D	0	D
Object 3	E	F + G	E + F + G
Object 4	G	0	G

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

Particle Perimeter

Length of the outer contour of an object.

Holes’ Perimeter

Sum of the perimeters of the holes within an object.

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 ÷ Particle 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 an object, or $\max(X_i) - \min(X_i)$. It is also equal to the horizontal side of the smallest horizontal rectangle containing the object, or the difference $\max X - \min X$.

Height

Distance between the upper-most and lower-most pixels in an object, or max(Y_i) – min(Y_i). It is also equal to the vertical side of the smallest horizontal rectangle containing the object, 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 object
- **Max X, Max Y**—Lower-right corner of the smallest horizontal rectangle containing the object
- **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 an object. The center of gravity of an object 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 an object.

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 an object.

Note G can be located outside an object if the latter has a convex shape.

Min(X, Y) and Max(X, Y)

Coordinates of the upper-left and lower-right corners of the smallest horizontal rectangle containing an object.

The origin (0, 0) has two pixels that have the coordinates (min*X*, min*Y*) and (max*X*, max*Y*) such that

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

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

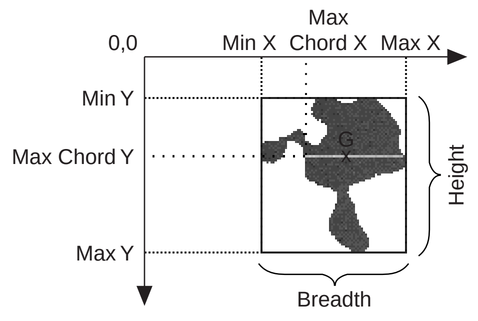
$$\max X = \max(X_i)$$

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

where X_i and Y_i are the coordinates of the pixels P_i in an object.

Max Chord X and Max Chord Y

Coordinates of the left-most pixel along the longest horizontal chord in an object.



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 degree with respect to the horizontal axis

Max Chord Length

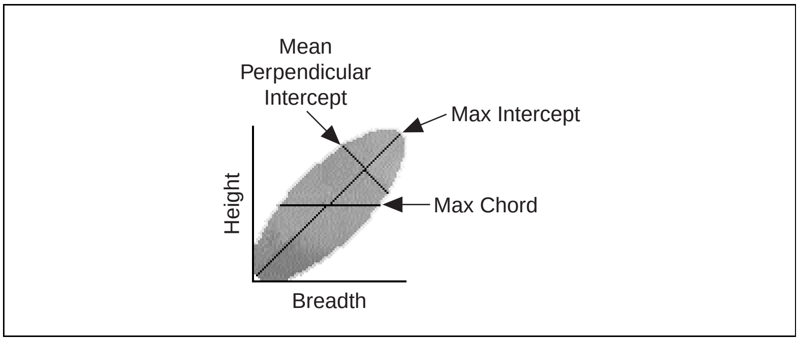
Length of the longest horizontal chord in an object.

Mean Chord X

Mean length of horizontal segments in an object.

Mean Chord Y

Mean length of vertical segments in an object.



Max Intercept

Length of the longest segment in an object (in all possible directions of projection).

Mean Intercept Perpendicular

Mean length of the segments in an object perpendicular to the max intercept.

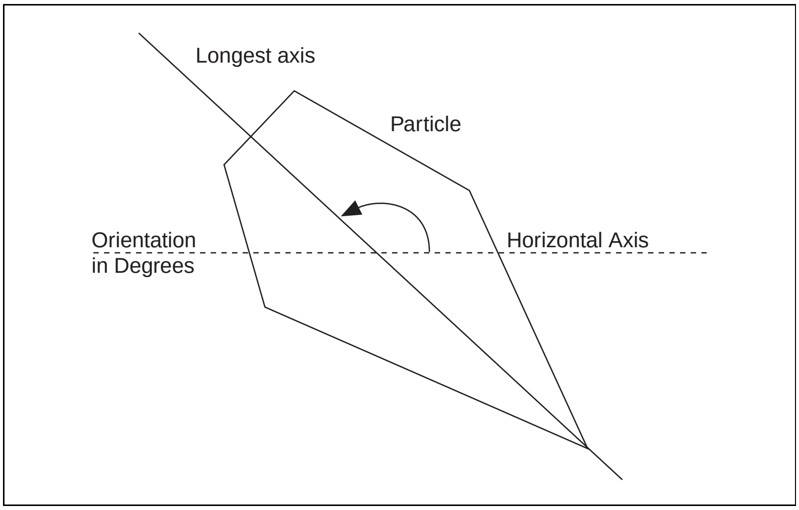
$$\text{Mean intercept perpendicular} = \frac{\text{particle area}}{\text{max intercept}}$$

Particle Orientation

The angle of the longest axis with respect to the horizontal axis. The value can be between 0° and 180°.

Notice that this value does not give information regarding the symmetry of the particle.

Therefore, an angle of 190° is considered the same as 10°.



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 object 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 object
- **Ellipse minor axis—**Minor axis of the ellipse that has the same area and same perimeter as the object
- **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 object

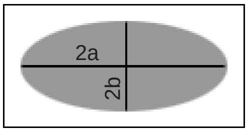
- **Rectangle small side**—Small side of the rectangle that has the same area and same perimeter as the object
- **Rectangle ratio**—Ratio of the big side of the equivalent rectangle to its small side

Equivalent Ellipse Minor Axis

The *equivalent ellipse minor axis* is the minor axis of the ellipse that has the same area as the object and a major axis equal to half the max intercept of the object.

This definition gives the following set of equations:

$$\text{particle area} = \pi ab \text{ and}$$
$$\text{max intercept} = 2a$$



The equivalent ellipse minor axis is defined as

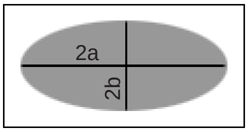
$$2b = \frac{4 \times \text{particle area}}{\pi \times \text{max intercept}}$$

Ellipse Major Axis

The *ellipse major axis* is the total length of the major axis of the ellipse that has the same area and same perimeter as an object. This length is equal to $2a$.

This definition gives the following set of equations:

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



This set of equations can be expressed so that the sum $a + b$ and the product ab become functions of the parameters **Particle Area** and **Particle Perimeter**. a and b then become the two solutions of the polynomial equation $X^2 - (a + b)X + ab = 0$.

Notice that for a given area and perimeter, only one solution (a, b) exists.

Ellipse Minor Axis

The *ellipse minor axis* is the total length of the minor axis of the ellipse that has the same area and same perimeter as an object. 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 $\frac{\text{ellipse major axis}}{\text{ellipse minor axis}} = \frac{a}{b}$

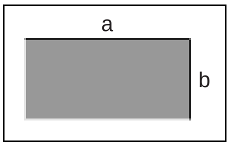
The more elongated the equivalent ellipse, the higher the ellipse ratio. The closer the equivalent ellipse is to a circle, the closer to 1 the ellipse ratio.

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 an object.

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 an object. 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 object to the mean length of the perpendicular segments
- **Compactness factor**—Ratio of the object area to the area of the smallest rectangle containing the object
- **Heywood Circularity factor**—Ratio of the object perimeter to the perimeter of the circle with the same area
- **Hydraulic Radius**—Ratio of the object area to its perimeter
- **Waddel Disk Diameter**—Diameter of the disk with the same area as the object

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

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

Elongation Factor

The *elongation factor* is the ratio of the longest segment within an object 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 an object, the higher its elongation factor.

Compactness Factor

The *compactness factor* is the ratio of an object area to the area of the smallest rectangle containing the object. 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 an object is to a rectangle, the closer to 1 the compactness factor.

Heywood Circularity Factor

The *Heywood circularity factor* is the ratio of an object 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 an object is to a disk, the closer the Heywood circularity factor to 1.

Hydraulic Radius

The *hydraulic radius* is the ratio of an object 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	Left-most point
Top	Top-most point
Right	Right-most point
Bottom	Bottom-most point
P_x	Projection *x*
P_y	Projection *y*

Derived Measurements

Symbol	Derived Measurement	Primary Measurement
l	**Width**	Right – Left
h	**Height**	Bottom – Top
d	**Diagonal**	$\sqrt{l^2 + h^2}$
M_x	**Center of Mass X**	$(\Sigma x)/A$
M_y	**Center of Mass Y**	$(\Sigma y)/A$
I_{xx}	**Inertia XX**	$(\Sigma x^2) - A \times M_x{}^2$
I_{yy}	**Inertia YY**	$(\Sigma y^2) - A \times M_y{}^2$
I_{xy}	**Inertia XY**	$(\Sigma xy) - A \times M_x \times M_y$
C_x	**Mean Chord X**	A/P_y
C_y	**Mean Chord X**	A/P_x
$S_{\max}$	**Max Intercept**	$(C_{\max}/l)^2 \times \max(h, l) + d(1 - (C_{\max}/l)^2)$
C	**Mean Perpendicular Intercept**	$A/S_{\max}$
A_{2b}	**Equivalent Ellipse Minor Axis**	$4 \times A / (\pi\, S_{\max})$
$d°$	**Orientation**	If $I_{xx} = I_{yy}$, then $d° = 45$, *else* $d° =$ $\dfrac{90}{\operatorname{atan}(2 \times I_{XY} \div (I_{XX} - I_{YY}))}$ If $I_{xx} \geq I_{yy}$ and $I_{xy} \geq 0$, then $d° = 180 - d°$ If $I_{xx} \geq I_{yy}$ and $I_{xy} < 0$, then $d° = -d°$ If $I_{xx} < I_{yy}$, then $d° = 90 - d°$ If $d° < 0$, then $d° = 0°$
E_{2a}	**Ellipse major axis (2*a*)**	$E_{2a} = \sqrt{\dfrac{p^2}{2\pi^2} + \dfrac{2\pi}{A}} + \sqrt{\dfrac{p^2}{2\pi^2} - \dfrac{2\pi}{A}}$

Symbol	Derived Measurement	Primary Measurement
E_{2b}	**Ellipse minor axis (2*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}	**Ellipse Ratio**	E_{2a} / E_{2b}
R_c	**Rectangle big side**	$\frac{1}{4}(p + t')$ where $t' = \sqrt{p^2 - 16A}$
r_c	**Rectangle small side**	$\frac{1}{4}(p - t')$ where $t' = \sqrt{p^2 - 16A}$
R_{Rr}	**Rectangle Ratio**	R_c / r_c
F_e	**Elongation factor**	$S_{\max} / C\pi$
F_c	**Compactness factor**	$A/(h \times l)$
F_H	**Heywood Circularity factor**	$\frac{p}{2\sqrt{\pi A}}$
F_t	**Type factor**	$\frac{A^2}{4\pi\sqrt{I_{XX} \times I_{YY}}}$
R_h	**Hydraulic Radius**	A/p
R_d	**Waddel Disk Diameter**	$2\sqrt{\frac{A}{\pi}}$

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 object expressed in gray-level units
- **Mean Gray Value**—Mean intensity value in the object 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 object expressed in user units
- **Mean User Value** —Mean intensity value in the object 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 *x*y 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

A

Technical Support Resources

National Instruments offers technical support through electronic, fax, and telephone systems. The electronic services include our Web site, an FTP site, and a fax-on-demand system. If you have a hardware or software problem, please first try the electronic support systems. If the information available on these systems does not answer your questions, contact one of our technical support centers, which are staffed by applications engineers, for support by telephone and fax. To comment on the documentation supplied with our products, send e-mail to `techpubs@natinst.com`.

Web Site

The InstrumentationWeb address is `http://www.natinst.com`.

From this Web site you can connect to our Web sites around the world (`http://www.natinst.com/niglobal/`) and access technical support (`http://www.natinst.com/support/`).

FTP Site

To access our FTP site, log in to our Internet host, `ftp.natinst.com`, as `anonymous` and use your e-mail address, such as `yourname@anywhere.com`, as your password. The support files and documents are located in the `\support` directories.

Fax-on-Demand Support

Fax-on-Demand is a 24-hour information retrieval system containing a library of documents in English on a wide range of technical information. You can access Fax-on-Demand from a touch-tone telephone at 512 418 1111.

E-Mail Support

You can submit technical support questions to the applications engineering team through e-mail at `support@natinst.com`. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

Telephone and Fax Support

National Instruments has branch offices all over the world. Use the following list to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

Country	Telephone	Fax
Australia	03 9879 5166	03 9879 6277
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Brazil	011 284 5011	011 288 8528
Canada (Ontario)	905 694 0085	905 785 0086
Canada (Québec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	09 725 725 11	09 725 725 55
France	0 1 48 14 24 24	0 1 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
India	91805275406	91805275410
Israel	03 6120092	03 6120095
Italy	02 413091	02 4139215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico (D.F.)	5 280 7625	5 520 3282
Mexico (Monterrey)	8 357 7695	8 365 8543
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain (Madrid)	91 640 0085	91 640 0533
Spain (Barcelona)	93 582 0251	93 582 4370
Sweden	08 587 895 00	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 2377 1200	02 2737 4644
United Kingdom	01635 523545	01635 523154
United States	512 795 8248	512 794 5678

Glossary

Prefix	Meanings	Value
p-	pico-	10^{-12}
n-	nano-	10^{-9}
µ-	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

ActiveX	Set of Microsoft technologies for reusable software components. Formerly called *OLE*.
ActiveX control	Standard software tool that adds additional functionality to any compatible ActiveX container. An ActiveX control has properties, methods, objects, and events.
AIPD	The National Instruments internal image format used for saving calibration information associated with an image and for saving complex images.

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 prior to 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

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.
BMP	Image format commonly used for 8-bit images on PCs.
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.
collection	Control property and object that contains a number of objects of the same type, such as pointers, axes, and plots. The type name of the collection is the plural of the type name of the object in the collection. For example, a collection of CWAxis objects is called CWAxes. To reference an object in the collection, you must specify the object as part of the collection, usually by index. For example, `CWGraph.Axes.Item(2)` is the second axis in the CWAxes collection of a graph.

color images	Images containing color information, usually encoded in the RGB form.
color lookup table	Table for converting the value of a pixel in an image into a red, green, and blue (RGB) intensity.
complex images	Save information obtained from the FFT of an image. The complex numbers which 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.
control	Standard software tool that adds additional functionality to a compatible environment. ComponentWorks is a collection of ActiveX controls that have properties, methods, objects, and events.
convex function	Computes the convex regions of objects in a binary image.
convolution	*See* linear filter.
convolution kernel	Simple 3×3, 5×5, or 7×7 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.
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.
driver	Software that controls a specific hardware device, such as a data acquisition board.

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 level between a rising and a falling edge.
edge steepness	The number of pixels that corresponds to the slope or transition area of an edge.
Equalize function	*See* histogram equalization.
erosion	Reduces the size of an object along its boundary and eliminates isolated points in the image.
event	Object-generated response to some action or change in state, such as a mouse click or x number of points being acquired. The event calls an event handler (callback function), which processes the event. Events are defined as part of an OLE control object.
event handler	*See* callback (function) *and* event.
exception	Error message generated by a control and sent directly to the application or programming environment containing the control.

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

Fast Fourier Transform	A method used to compute the Fourier transform of an image.
FFT	Fast Fourier Transform.
form	Window or area on the screen on which you place controls and indicators to create the user interface for your program.
Fourier spectrum	The magnitude information of the Fourier transform of an image.
Fourier Transform	Transforms an image from the spatial domain to the frequency domain.
frequency filters	Counterparts of spatial filters in the frequency domain. For images, frequency information is in the form of spatial frequency.
FTP	File Transfer Protocol. Protocol based on TCP/IP to exchange files between computers.

G

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	*See* 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.
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.
HSL	Color encoding scheme in Hue, Saturation, and Lightness.
HSV	Color encoding scheme in Hue, Saturation, and Value.

I

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 which you can apply to an image.
image visualization	The presentation (display) of an image (image data) to the user.
inner gradient	Finds the inner boundary of objects.
inspection functions	Detects specific features in an image. The features detected include 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 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	Is the technique used to find values in between known values when resampling an image or array of pixels.

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 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.
lookup table	Table containing values used to transform the gray-level values of an image. For each gray-level value in the image, the corresponding new value is obtained from the lookup table.
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.

M

mask	Isolates parts of an image for further processing.
mask 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.
mechanical action	Specifies how a zone is activated. In the **Switch** mode, the first click on a zone turns the zone to TRUE and a second click turns it to FALSE. In the **Latch** mode, a click causes the zone to be temporarily TRUE.
median filter	A low pass filter that assigns to each pixel the median value of its neighbors. This filter effectively removes isolated pixels without blurring the contours of objects.
method	Function that performs a specific action on or with an object. The operation of the method often depends on the values of the object properties.
mile	An Imperial unit of length equal to 5,280 feet or 1,609.344 meters. Also known as a statute mile to discern from a nautical mile. *See also* nautical mile.
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

nautical mile	International unit of length used for sea and air navigation equal to 6,076.115 feet or 1,852 meters. *See also* mile.
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 *N*th pixel value, where *N* is the order of the filter.

O

object	Software tool for accomplishing tasks in different programming environments. An object can have properties, methods, and events. You change an object’s state by changing the values of its properties. An object’s behavior consists of the operations (methods) that can be performed on it and the accompanying state changes. *See* property, method, event.
Object Browser	Dialog window that displays the available properties and methods for the controls that are loaded. The object browser shows the hierarchy within a group of objects. To activate the object browser in Visual Basic, press <F2>.
OCX	OLE Control eXtension. Another name for ActiveX controls, reflected by the .OCX file extension of ActiveX control files.
OLE	Object Linking and Embedding. *See* ActiveX.
OLE control	*See* ActiveX control.
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 representation	Contains the low-frequency information at the center and the high-frequency information at the corners of an FFT-transformed image.
outer gradient	Finds the outer boundary of objects.

P

palette	The gradation of colors used to display an image on screen, usually defined by a color lookup table.
PICT	Image format commonly used for 8-bit images on Macintosh and Power Macintosh platforms.
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.
Power 1/Y function	Similar to a logarithmic function but with a weaker effect.
Power Y function	*See* exponential function.
Prewitt filter	Extracts the contours (edge detection) in gray-level values using a 3 × 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.
property	Attribute that controls the appearance or behavior of an object. The property can be a specific value or another object with its own properties and methods. For example, a value property is the color (property) of a plot (object), while an object property is a specific Y axis (property) on a graph (object). The Y axis itself is another object with properties, such as minimum and maximum values.

Q

quantitative analysis	Obtaining various measurements of objects in an image.

R

reference	Link to an external code source in Visual Basic. References are anything that add additional code to your program, such as OLE controls, DLLs, objects, and type libraries. You can add references by selecting the **Tools»References…** menu.
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.
Reverse function	Inverts the pixel values in an image, producing a photometric negative of the image.
RGB	Color image encoding using red, green, and blue colors.
RGB chunky	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.
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

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.
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 × 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	*See* exponential function.
Square Root function	*See* logarithmic function.
standard representation	Contains the low-frequency information at the corners and high-frequency information at the center of an FFT-transformed image.
start condition	Condition on a data acquisition process that determines when the actual acquisition starts. The condition can be a software trigger, an analog hardware trigger, or a digital hardware trigger.
statute mile	*See* mile.
stop condition	Condition on a data acquisition process that determines when the acquisition stops. The condition can be none (the acquisition stops when all points have been acquired), continuous (the acquisition runs continuously), software analog trigger, hardware analog trigger, or hardware digital trigger.
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.
TIFF	Image format commonly used for encoding 8-bit and 16-bit images and color images on both Macintosh and PC platforms.
truth table	A table associated with a logic operator which describes the rules used for that operation.

Z

zones	Areas in a displayed image that respond to user clicks.

Index