

# IMAQ™

---

## IMAQ Vision for Measurement Studio™ User Manual

Visual Basic

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

ni.com

### **National Instruments Corporate Headquarters**

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

### **Worldwide Offices**

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,  
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 514 694 8521,  
Canada (Toronto) 905 785 0085, China (Shanghai) 021 6555 7838, China (ShenZhen) 0755 3904939,  
Denmark 45 76 26 00, Finland 09 725 725 11, France 01 48 14 24 24, Germany 089 741 31 30,  
Greece 30 1 42 96 427, Hong Kong 2645 3186, India 91805275406, Israel 03 6120092, Italy 02 413091,  
Japan 03 5472 2970, Korea 02 596 7456, Malaysia 603 9596711, Mexico 5 280 7625, Netherlands 0348 433466,  
New Zealand 09 914 0488, Norway 32 27 73 00, Poland 0 22 528 94 06, Portugal 351 1 726 9011,  
Singapore 2265886, Spain 91 640 0085, Sweden 08 587 895 00, Switzerland 056 200 51 51,  
Taiwan 02 2528 7227, United Kingdom 01635 523545

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

Copyright © 2001 National Instruments Corporation. All rights reserved.

# Important Information

---

## Warranty

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

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

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

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

## Copyright

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

## Trademarks

CVI™, IMAQ™, LabVIEW™, Measurement Studio™, National Instruments™, NI™, ni.com™, and NI-IMAQ™ are trademarks of National Instruments Corporation.

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

## WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

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

# Conventions

---

The following conventions are used in this manual:

» 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 tip, which alerts you to advisory information.



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.

*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 paths, directories, programs, procedures, functions, operations, variables, filenames and extensions, and code excerpts.

# Contents

---

## Chapter 1

### Introduction to IMAQ Vision

About IMAQ Vision .....	1-1
Documentation and Examples .....	1-1
IMAQ Vision for Visual Basic Organization .....	1-3
cwimaq.ocx.....	1-3
CWIMAQ Control .....	1-3
CWIMAQVision Control.....	1-3
CWIMAQViewer Control.....	1-3
cwmv.ocx.....	1-4
CWMachineVision control .....	1-4
ActiveX Objects .....	1-4
Creating IMAQ Vision Applications .....	1-5

## Chapter 2

### Getting Measurement-Ready Images

Set Up Your Imaging System .....	2-1
Calibrate Your Imaging System .....	2-2
Create an Image .....	2-2
Acquire or Read an Image .....	2-4
Acquiring an Image .....	2-4
One-Shot Acquisition.....	2-4
Continuous Acquisition.....	2-5
Reading a File.....	2-6
Converting an Array to an Image .....	2-6
Display an Image .....	2-6
Attach Calibration Information.....	2-7
Analyze an Image .....	2-7
Improve an Image .....	2-9
Lookup Tables .....	2-9
Filters.....	2-9
Convolution Filter .....	2-10
Nth Order Filter.....	2-10
Grayscale Morphology .....	2-10
FFT .....	2-11
Complex Image Operations .....	2-12

## Chapter 3 Grayscale and Color Measurements

Define Regions of Interest.....	3-1
Interactively Defining Regions .....	3-1
Programmatically Defining Regions.....	3-5
Defining Regions with Masks.....	3-6
Measure Grayscale Statistics.....	3-7
Measure Color Statistics.....	3-7
Comparing Colors.....	3-8
Learning Color Information .....	3-9
Choosing the Right Color Information .....	3-9
Specifying the Color Information to Learn .....	3-10
Choosing a Color Representation Sensitivity .....	3-12
Ignoring Learned Colors.....	3-13

## Chapter 4 Blob Analysis

Correct Image Distortion.....	4-2
Create a Binary Image.....	4-2
Improve the Binary Image.....	4-3
Removing Unwanted Blobs .....	4-3
Separating Touching Blobs.....	4-4
Improving Blob Shapes.....	4-4
Make Particle Measurements .....	4-4
Convert Pixel Coordinates to Real-World Coordinates .....	4-7

## Chapter 5 Machine Vision

Locate Objects to Inspect .....	5-2
Using Edge Detection to Build a Coordinate Transformation.....	5-3
Using Pattern Matching to Build a Coordinate Transformation .....	5-6
Choosing a Method to Build the Coordinate Transformation.....	5-7
Set Search Areas.....	5-8
Interactively Defining Regions .....	5-8
Programmatically Defining Regions.....	5-9
Find Measurement Points .....	5-9
Finding Features Using Edge Detection .....	5-9
Finding Lines or Circles .....	5-9
Finding Edge Points Along One Search Contour .....	5-11
Finding Edge Points Along Multiple Search Contours .....	5-12

Finding Points Using Pattern Matching .....	5-12
Defining and Creating Good Template Images.....	5-13
Training the Pattern Matching Algorithm.....	5-15
Defining a Search Area .....	5-15
Setting Matching Parameters and Tolerances.....	5-16
Testing the Search Algorithm on Test Images.....	5-17
Using a Ranking Method to Verify Results .....	5-17
Finding Points Using Color Pattern Matching .....	5-18
Defining and Creating Good Color Template Images .....	5-18
Training the Color Pattern Matching Algorithm.....	5-19
Defining a Search Area .....	5-20
Setting Matching Parameters and Tolerances.....	5-21
Testing the Search Algorithm on Test Images.....	5-24
Finding Points Using Color Location.....	5-24
Convert Pixel Coordinates to Real-World Coordinates.....	5-25
Make Measurements .....	5-25
Distance Measurements.....	5-25
Analytic Geometry Measurements .....	5-26
Instrument Reader Measurements .....	5-26
Display Results .....	5-27

## Chapter 6

### Calibration

Perspective and Nonlinear Distortion Calibration .....	6-1
Defining a Calibration Template .....	6-2
Defining a Reference Coordinate System .....	6-3
Learning Calibration Information.....	6-5
Specifying Scaling Factors.....	6-6
Choosing a Region of Interest.....	6-6
Choosing a Learning Algorithm .....	6-6
Using the Learning Score.....	6-7
Learning the Error Map.....	6-8
Learning the Correction Table .....	6-8
Setting the Scaling Mode .....	6-8
Calibration Invalidation .....	6-9
Simple Calibration .....	6-9
Save Calibration Information.....	6-10
Attach Calibration Information.....	6-10

**Appendix A**  
**Technical Support Resources**

**Glossary**

**Index**



---

# Introduction to IMAQ Vision

This chapter describes the IMAQ Vision for Visual Basic software and associated software products, discusses the documentation and examples available, outlines the IMAQ Vision for Visual Basic organization, and lists the steps for making a machine vision application.



**Note** For information about the system requirements and installation procedure for IMAQ Vision for Visual Basic, see the *IMAQ Vision for Measurement Studio Release Notes* that came with your software.

---

## About IMAQ Vision

IMAQ Vision for Measurement Studio is a National Instruments product comprised of IMAQ Vision for Visual Basic and IMAQ Vision for LabWindows/CVI. IMAQ Vision for Visual Basic is a collection of ActiveX controls that you can use to develop machine vision and scientific imaging applications. IMAQ Vision for LabWindows/CVI is a library of C functions that offer the same imaging functionality as IMAQ Vision for Visual Basic. National Instruments also offers IMAQ Vision for LabVIEW, which is a library of LabVIEW VIs for developing machine vision and scientific imaging applications. IMAQ Vision Builder, another software product from National Instruments, allows you to prototype your application strategy quickly without having to do any programming.

---

## Documentation and Examples

This manual assumes that you are familiar with Visual Basic and can use ActiveX controls in Visual Basic. The following are good sources of information about Visual Basic and ActiveX controls:

- [msdn.microsoft.com](http://msdn.microsoft.com)
- documentation that accompanies Visual Studio

In addition to this manual, several documentation resources are available to help you create your vision application:

- *IMAQ Vision Concepts Manual*—If you are new to machine vision and imaging, read this manual to understand the concepts behind IMAQ Vision.
- *IMAQ Vision for Visual Basic Reference*—If you need information about individual methods, properties, or objects, see this help file. Access this file from within Visual Basic or from the **Start** menu by selecting **Programs»National Instruments»Vision»Documentation**.
- *NI-IMAQ User Manual*—If you have a National Instruments image acquisition (IMAQ) device and need information about the functions that control your IMAQ device, see this portable document (PDF) file which was installed at the following location when you installed NI-IMAQ: **Start»Programs»National Instruments»Vision»Documentation**. You need Adobe Acrobat Reader to open this file.
- *Example programs*—If you want examples of how to create specific applications, go to `MeasurementStudio\VB\Samples\IMAQ`.
- *CWMachineVision source code*—If you want to see the source code for the `CWMachineVision` control, go to `MeasurementStudio\VB\SourceCode`.
- *Application Notes*—If you want to know more about advanced IMAQ Vision concepts and applications, refer to the Application Notes located on the National Instruments Web site at `ni.com/appnotes.nsf/`.
- *NI Developer Zone (NIDZ)*—If you want even more information about developing your vision application, visit the NI Developer Zone at `ni.com/zone`. The NI Developer Zone contains example programs, tutorials, technical presentations, the Instrument Driver Network, a measurement glossary, an online magazine, a product advisor, and a community area where you can share ideas, questions, and source code with vision developers around the world.

# IMAQ Vision for Visual Basic Organization

---

IMAQ Vision for Visual Basic consists of four ActiveX controls contained in two files: `cwimaq.ocx` and `cwmv.ocx`.

## `cwimaq.ocx`

`cwimaq.ocx` contains three ActiveX controls and a collection of ActiveX objects. The ActiveX controls are the CWIMAQ control, CWIMAQVision control, and the CWIMAQViewer control. See the [ActiveX Objects](#) section for more information about the ActiveX objects.

### CWIMAQ Control

Use this control to configure and perform an acquisition from your IMAQ acquisition device. The CWIMAQ control has property pages that allow you to modify various parameters to configure your acquisition and gather information about your IMAQ device. Most of the functionality available from the property pages during design time is also available through the properties of the CWIMAQ control during run-time. The control has methods that allow you to perform and control acquisitions, as well.



**Note** You must have the NI-IMAQ device driver installed on the target system for the CWIMAQ control to function. For more information about NI-IMAQ, see the *NI-IMAQ User Manual* that came with your IMAQ device.

### CWIMAQVision Control

Use this control to analyze and process images and their related data. The CWIMAQVision control provides methods for reading and writing images to and from files, analyzing images, and performing a variety of image processing algorithms on images.

### CWIMAQViewer Control

Use this control to display images and provide the interface through which the user will interact with the displayed image. This includes the ability to zoom and pan images and to draw regions of interest (ROIs) on an image. The CWIMAQViewer control has property pages that allow you to configure the viewer's appearance and behavior during design time as well as properties that you can configure during run-time. The control has methods that allow you to attach images to and detach images from the viewer for display purposes.



**Note** The CWIMAQViewer control is referred to as a *viewer* in the rest of this document.

## CWMV.OCX

`cwmv.ocx` contains one ActiveX control and a collection of ActiveX objects. See the *ActiveX Objects* section for more information about ActiveX objects.

## CWMachineVision control

Use this control to perform high-level machine vision tasks, such as measuring distances. This control is written entirely in Visual Basic using the methods on the CWIMAQVision and CWIMAQViewer controls. The source code for the CWMachineVision control is included in the product. For more information about CWMachineVision methods, see Chapter 5, [Machine Vision](#).



**Tip** See the source code of the CWMachineVision control for an example of how to use the CWIMAQVision methods.

## ActiveX Objects

ActiveX objects are classified as input and output objects. Use the objects to group related input parameters and output parameters to certain methods, thus reducing the number of parameters that you actually need to pass to those methods.



**Note** ActiveX objects in `cwimaq.ocx` have a CWIMAQ prefix, and objects in `cwmv.ocx` have a CWMV prefix.

You must create an ActiveX object before you can use it. You can use the `New` keyword in Visual Basic to create these objects. For example, use the following syntax to create and store an image in a variable named `image`:

```
Dim image as New CWIMAQImage
```



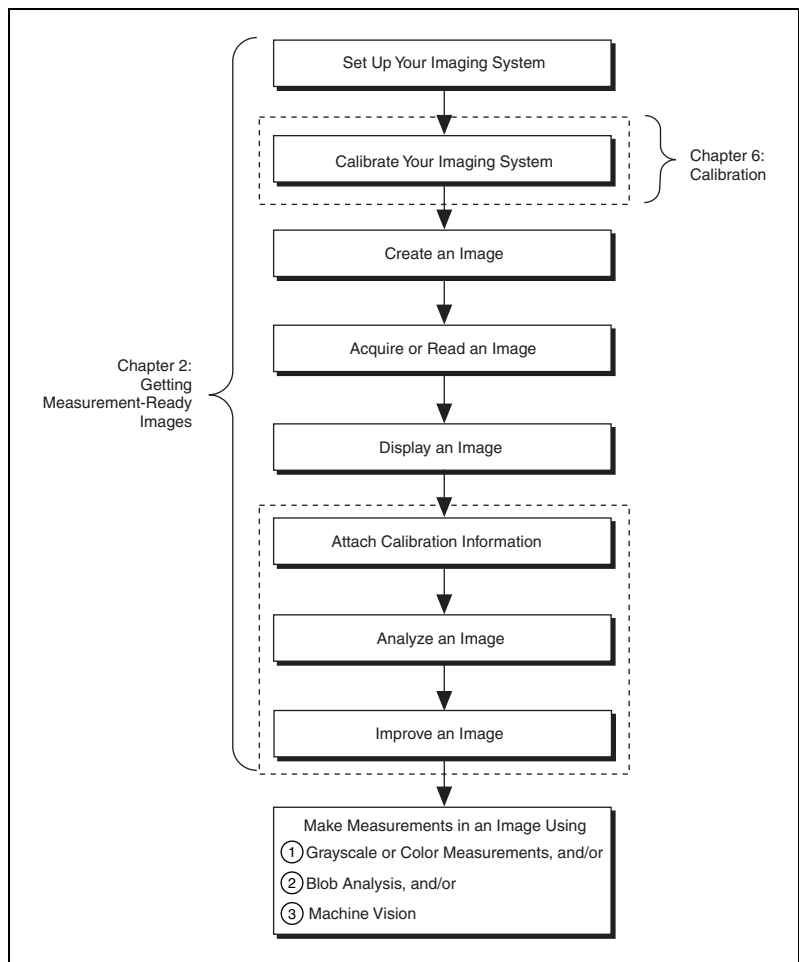
**Tip** If you intend to develop your application in Visual C++, National Instruments recommends that you use IMAQ Vision for LabWindows/CVI. However, if you decide to use IMAQ Vision for Visual Basic to develop applications for Visual C++, you can create objects using the respective `Create` methods on the CWIMAQVision control or CWMachineVision control. For example, to create a CWIMAQImage object, use the `CWIMAQVision.CreateCWIMAQImage` method.

# Creating IMAQ Vision Applications

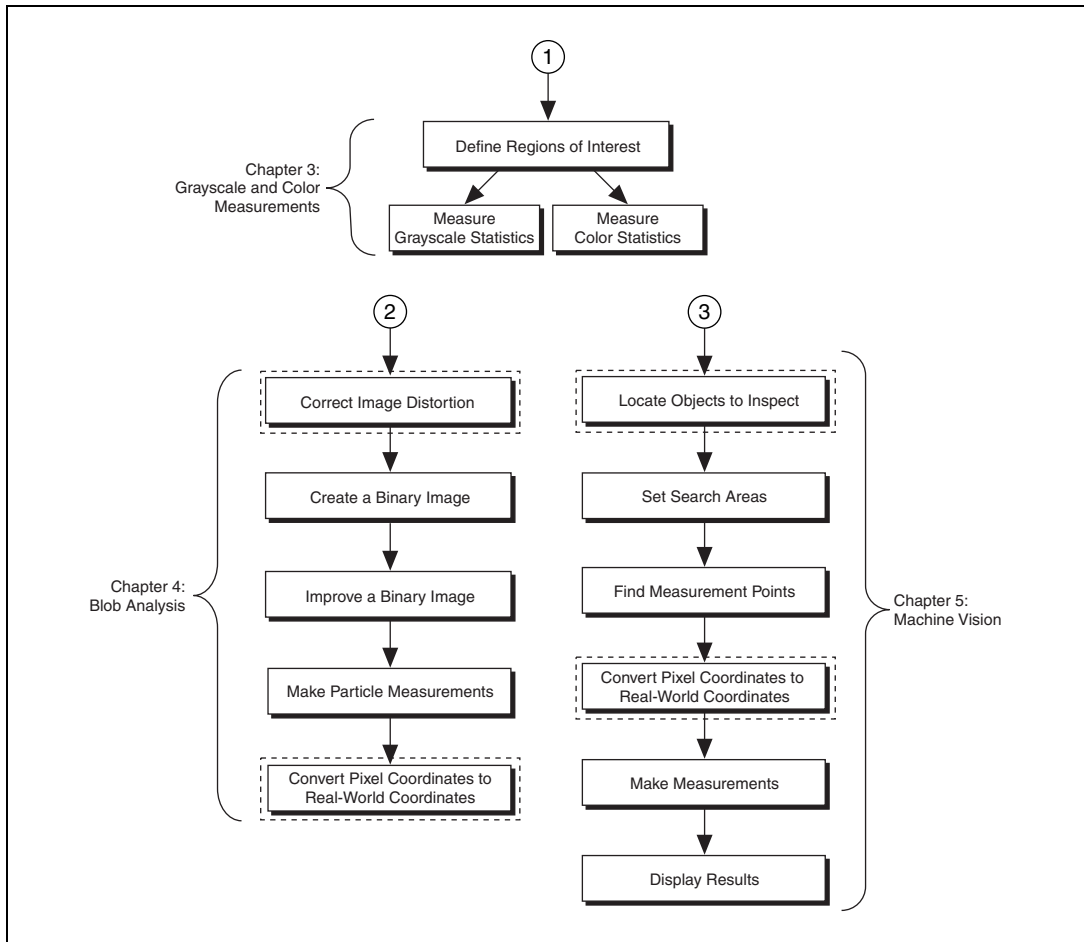
Figures 1-1 and 1-2 illustrate the steps for creating an application with IMAQ Vision. Figure 1-1 describes the general steps to designing a Vision application. The last step in Figure 1-1 is expanded upon in Figure 1-2. You can use a combination of the items in the last step to create your IMAQ Vision application. For more information about items in either diagram, see the corresponding chapter listed to the right of the item.



**Note** Diagram items enclosed with dashed lines are optional steps.



**Figure 1-1.** General Steps to Designing a Vision Application



**Figure 1-2.** Inspection Steps for Building a Vision Application

---

# Getting Measurement-Ready Images

This chapter describes how to set up your imaging system, acquire and display an image, analyze the image, and prepare the image for additional processing.

## Set Up Your Imaging System

---

Before you acquire, analyze, and process images, you must set up your imaging system. The manner in which you set up your system depends on your imaging environment and the type of analysis and processing you need to do. Your imaging system should produce images with high enough quality so that you can extract the information you need from the images.

Follow the guidelines below to setup your imaging system.

1. Determine the type of equipment you need given your space constraints and the size of the object you need to inspect. For more information, see Chapter 3, *System Setup and Calibration*, of the *IMAQ Vision Concepts Manual*.
  - a. Make sure your camera sensor is large enough to satisfy your minimum resolution requirement.
  - b. Make sure your lens has a depth of field high enough to keep all of your objects in focus regardless of their distance from the lens. Also, make sure your lens has a focal length that meets your needs.
  - c. Make sure your lighting provides enough contrast between the object under inspection and the background for you to extract the information you need from the image.
2. Position your camera so that it is parallel to the object under inspection. If your camera acquires images of the object from an angle, perspective errors occur. You can compensate for these errors with software, but using a parallel inspection angle obtains the fastest and most accurate results.

3. Select an image acquisition device that meets your needs. National Instruments offers several image acquisition (IMAQ) devices, such as analog color and monochrome IMAQ devices as well as digital devices. Visit [ni.com/imaq](http://ni.com/imaq) for more information about IMAQ devices.



4. Configure the driver software for your image acquisition device. If you have a National Instruments image acquisition device, configure your NI-IMAQ driver software through Measurement & Automation Explorer (MAX). Open MAX by double-clicking the Measurement & Automation Explorer icon on your desktop. For more information, see the *NI-IMAQ User Manual* and the MAX online help.

## Calibrate Your Imaging System

---

After you set up your imaging system, you may want to calibrate your system. Calibrate your imaging system to assign real-world coordinates to pixel coordinates and compensate for perspective and nonlinear errors inherent in your imaging system.

Perspective errors occur when your camera axis is not perpendicular to the object under inspection. Nonlinear distortion may occur from aberrations in the camera lens. Perspective errors and lens aberrations cause images to appear distorted. This distortion displaces information in an image, but it does not necessarily destroy the information in the image.

Use simple calibration if you only want to assign real-world coordinates to pixel coordinates. Use perspective and nonlinear distortion calibration if you need to compensate for perspective errors and nonlinear lens distortion. For detailed information about calibration, see Chapter 6, [Calibration](#).

## Create an Image

---

The `CWIMAQImage` object encapsulates all the information required to represent an image.



**Note** `CWIMAQImage` is referred to as an *image* in the rest of this document.

An image can be one of many types depending on the data it stores. The following image types are valid:

- 8-bit
- 16-bit



- Single-precision floating point
- Complex
- 32-bit RGB
- 32-bit HSL

When you create an image, it is an 8-bit image by default. You can set the `Type` property on the image object to change the image type.

If you plan to use filtering or blob analysis methods on the image, set the appropriate border size for the image. The default border size is 2.

When you create an image, no memory is allocated to store the image pixels. IMAQ Vision methods automatically allocate the appropriate amount of memory when the image size is modified. For example, methods that acquire or resample an image alter the image size, so they allocate the appropriate memory space for the image pixels.

Most methods belonging to the IMAQ Vision library require an input of one or more image objects. The number of images a method takes depends on the image processing function and the type of image you want to use.

IMAQ Vision methods that analyze the image but do not modify the image contents require the input of one source image. Methods that process the contents of the image require one or more source images and a destination image. Exceptions to the preceding sentences are methods that take a mask image as input.

The presence of a `MaskImage` parameter indicates that the processing or analysis is dependent on the contents of another image (the mask image). The only pixels in the source image that are processed are those whose corresponding pixels in the mask image are non-zero. If a mask image pixel is 0, the corresponding source image pixel is not processed or analyzed. The mask image must be an 8-bit image.

If you want to apply a processing or analysis method to the entire image, do not supply the optional mask image. Using the same image for both the source image and mask image also has the same effect as not using the mask image, except in this case the source image must be an 8-bit image.

Most operations between two images require that the images have the same type and size. However, arithmetic operations work between two different types of images. For example, an arithmetic operation between an 8-bit image and 16-bit image would result in a 16-bit image.

## Acquire or Read an Image

---

After you create an image, you can acquire an image into your imaging system in three ways. You can acquire an image with a camera through your image acquisition device, load an image from a file stored on your computer, or convert the data stored in a 2D array to an image. Methods that acquire images, load images from file, or convert data from a 2D array automatically allocate the memory space required to accommodate the image data.

### Acquiring an Image

Use the CWIMAQ control to acquire images with a National Instruments IMAQ device. You can use IMAQ Vision for Visual Basic to perform one-shot and continuous acquisitions. You can choose the acquisition type during design time by setting the value of the **Acquisition Type** combo box to **One-Shot** or **Continuous**. The **Acquisition Type** combo box is located on the Acquisition property page of the CWIMAQ control. You can set the value at run-time by setting the `CWIMAQ.AcquisitionType` property to `cwimaqAcquisitionOneShot` or `cwimaqAcquisitionContinuous`.

### One-Shot Acquisition

Use a one-shot acquisition to start an acquisition, perform the acquisition, and stop the acquisition using a single method. The number of frames acquired is equal to the number of images in the images collection. Use the `CWIMAQ.AcquireImage` method to perform this operation synchronously. Use the `CWIMAQ.Start` method to perform this operation asynchronously. For information about synchronous and asynchronous acquisitions, see the *NI-IMAQ User Manual*.

If you want to acquire a single field or frame into a buffer, set the image count to 1. This operation is also referred to as a *snap*. Use a snap for low-speed or single capture applications. The following code illustrates a synchronous snap:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType = cwimaqAcquisitionOneShot
    CWIMAQ1.AcquireImage
End Sub
```

If you want to acquire multiple frames, then set the image count to the number of frames you wish to acquire. This operation is called a *sequence*. Use a sequence for applications that process multiple images. The

following code illustrates an asynchronous sequence, where `numberOfImages` is the number of images that you want to process:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType = cwimaqAcquisitionOneShot
    CWIMAQ1.Images.RemoveAll
    CWIMAQ1.Images.Add numberOfImages
    CWIMAQ1.Start
End Sub
```

## Continuous Acquisition

Use a continuous acquisition to start an acquisition and continuously acquire frames into the image buffers, and then explicitly stop the acquisition. Use the `CWIMAQ.Start` method to start the acquisition. Use the `CWIMAQ.Stop` method to stop the acquisition. If you use a single buffer for the acquisition, this operation is called a *grab*. The following code illustrates a grab:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType = _cwimaqAcquisitionContinuous
    CWIMAQ1.Start
End Sub

Private Sub Stop_Click()
    CWIMAQ1.Stop
End Sub
```

A *ring* operation uses multiple buffers for the acquisition. Use a ring for high-speed applications that require processing on every image. The following code illustrates a ring, where `numberOfImages` is the number of images that you want to process:

```
Private Sub Start_Click()
    CWIMAQ1.AcquisitionType = _cwimaqAcquisitionContinuous
    CWIMAQ1.Images.RemoveAll
    CWIMAQ1.Images.Add numberOfImages
    CWIMAQ1.Start
End Sub
```

```
Private Sub Stop_Click()
    CWIMAQ1.Stop
End Sub
```

## Reading a File

Use the `CWIMAQVision.ReadImage` method to open and read data from a file stored on your computer into the image reference. You can read from image files stored in a standard format (BMP, TIFF, JPEG, PNG, and AIPD) or a nonstandard format you specify. In all cases, the software automatically converts the pixels it reads into the type of image you pass in.

Use the `CWIMAQVision.ReadImageAndVisionInfo` method to open an image file containing additional information, such as calibration information, template information for pattern matching, or overlay information. For more information about pattern matching templates and overlays, see Chapter 5, *Machine Vision*.

You can also use the `CWIMAQVision.GetFileInformation` method to retrieve image properties—image size, pixel depth, recommended image type, and calibration units—without actually reading all the image data.

## Converting an Array to an Image

Use the `CWIMAQImage.ArrayToImage` method to convert an array to an image. You can also use the `CWIMAQImage.ImageToArray` method to convert an image to an array.

## Display an Image

---

Display an image using the `CWIMAQViewer` control. Use `CWIMAQViewer.Attach` to attach the image you want the viewer to display. Once you attach an image to a viewer, the image automatically updates the viewer whenever an operation modifies the contents of the image. You can access the image attached to the viewer using the `CWIMAQViewer.Image` property. Before you attach an image to the viewer, the viewer already has an image attached by default. Therefore, the viewer has an image attached to it at all times. You can use the attached image as either a source image, destination image, or both using the `Image` property.

You can use the `CWIMAQViewer.Palette` property to access the `CWIMAQPalette` object associated with the viewer. Use the `CWIMAQPalette` object to programmatically apply a color palette to the viewer. You can set the `CWIMAQPalette.Type` property to apply predefined color palettes. For example, if you need to display a binary image—an image containing particle regions with pixel values of 1 and a background region with pixel values of 0—set the `Type` property to `cwimaqPaletteBinary`. For more information about color palettes, see Chapter 2, *Display*, of the *IMAQ Vision Concepts Manual*.

You can also set a default palette during design time using the `Menu` property page. Users can change the color palette during run time by using the pop-up menu on the viewer.

## Attach Calibration Information

---

If you want to attach the calibration information of the current setup to each image you acquire, use `CWIMAQVision.SetCalibrationInformation`. This method takes in a source image containing the calibration information and a destination image that you want to calibrate. The output image is your inspection image with the calibration information attached to it. For detailed information about calibration, see Chapter 6, *Calibration*.



**Note** Because calibration information is part of the image, it is propagated throughout the processing and analysis of the image. Methods that modify the image size (such as geometrical transforms) void the calibration information. Use `CWIMAQVision.WriteImageAndVisionInfo` to save the image and all of the attached calibration information to a file.

## Analyze an Image

---

Once you acquire and display an image, you may want to analyze the contents of the image for the following reasons:

- To determine whether the image quality is high enough for your inspection task.
- To obtain the values of parameters that you want to use in processing methods during the inspection process.

The histogram and line profile tools can help you analyze the quality of your images.

Use `CWIMAVision.Histogram2` to analyze the overall grayscale distribution in the image. Use the histogram of the image to analyze two important criteria that define the quality of an image—saturation and contrast. If your image is underexposed (does not have enough light) the majority of your pixels will have low intensity values, which appear as a concentration of peaks on the left side of your histogram. If your image is overexposed (has too much light) the majority of your pixels will have a high intensity values, which appear as a concentration of peaks on the right side of your histogram. If your image has an appropriate amount of contrast, your histogram will have distinct regions of pixel concentrations. Use the histogram information to decide if the image quality is high enough to separate objects of interest from the background.

If the image quality meets your needs, use the histogram to determine the range of pixel values that correspond to objects in the image. You can use this range in processing methods, such as determining a threshold range during blob analysis.

If the image quality does not meet your needs, try to improve the imaging conditions to get the desired image quality. You may need to re-evaluate and modify each component of your imaging setup: lighting equipment and setup, lens tuning, camera operation mode, and acquisition board parameters. If you reach the best possible conditions with your setup but the image quality still does not meet your needs, try to improve the image quality using the image processing techniques described in the [Improve an Image](#) section of this chapter.

Use `CWIMAVision.LineProfile2` to get the pixel distribution along a line in the image, or use `CWIMAVision.RegionsProfile` to get the pixel distribution along a one-dimensional path in the image. By looking at the pixel distribution, you can determine if the image quality is high enough to provide you with sharp edges at object boundaries. Also, you can determine if the image is noisy, and identify the characteristics of the noise.

If the image quality meets your needs, use the pixel distribution information to determine some parameters of the inspection methods you want to use. For example, use the information from the line profile to determine the strength of the edge at the boundary of an object. You can input this information into `CWIMAVision.FindEdges2` to find the edges of objects along the line.

# Improve an Image

---

Using the information you gathered from analyzing your image, you may want to improve the quality of your image for inspection. You can improve your image with lookup tables, filters, grayscale morphology, and Fast Fourier transforms.

## Lookup Tables

Apply *lookup table* (LUT) transformations to highlight image details in areas containing significant information at the expense of other areas. A LUT transformation converts input grayscale values in the source image into other grayscale values in the transformed image. IMAQ Vision provides four methods that directly or indirectly apply lookup tables to images:

- `CWIMAQVision.MathLookup`—Converts the pixel values of an image by replacing them with values from a predefined lookup table. IMAQ Vision has seven predefined lookup tables based on mathematical transformations. For more information about these lookup tables, see Chapter 5, *Image Processing*, in the *IMAQ Vision Concepts Manual*.
- `CWIMAQVision.UserLookup`—Converts the pixel values of an image by replacing them with values from a user-defined lookup table.
- `CWIMAQVision.Equalize2`—Distributes the grayscale values evenly within a given grayscale range. Use this method to increase the contrast in images containing few grayscale values.
- `CWIMAQVision.Inverse`—Inverts the pixel intensities of an image to compute the negative of the image. For example, use this method before applying an automatic threshold to your image if the background pixels are brighter than the object pixels.

## Filters

Filter your image when you need to improve the sharpness of transitions in the image or increase the overall signal-to-noise ratio of the image. You can choose either a lowpass or highpass filter, depending on your needs.

Lowpass filters remove insignificant details by smoothing the image, removing sharp details, and smoothing the edges between the objects and the background. You can use `CWIMAQVision.LowPass` or define your own lowpass filter with `CWIMAQVision.Convolute` or `CWIMAQVision.NthOrder`.

Highpass filters emphasize details, such as edges, object boundaries, or cracks. These details represent sharp transitions in intensity value. You can define your own highpass filter with `CWIMAQVision.Convolute` or `CWIMAQVision.NthOrder`, or you can use a predefined highpass filter with `CWIMAQVision.EdgeFilter` or `CWIMAQVision.CannyEdgeFilter`. `CWIMAQVision.EdgeFilter` allows you to find edges in an image using predefined edge detection kernels, such as the Sobel, Prewitt, and Roberts kernels.

## Convolution Filter

`CWIMAQVision.Convolute` allows you to use a predefined set of lowpass and highpass filters. Each filter is defined by a kernel of coefficients. Use the `CWIMAQKernel` object to define the filter. Use `CWIMAQKernel.LoadKernel` to load a predefined kernel into the object. If the predefined kernels do not meet your needs, use the `CWIMAQKernel.SetSize` method to set the size of the kernel and the `CWIMAQKernel.Element` property to set the data in the kernel.

## Nth Order Filter

`CWIMAQVision.NthOrder` allows you to define a lowpass or highpass filter depending on the value of  $N$  that you choose. One specific Nth order filter, the median filter, removes speckle noise, which appears as small black and white dots. For more information about Nth order filters, see Chapter 5, *Image Processing*, of the *IMAQ Vision Concepts Manual*.

## Grayscale Morphology

Perform grayscale morphology when you want to filter grayscale features of an image. Grayscale morphology helps you remove or enhance isolated features, such as bright pixels on a dark background. Use these transformations on a grayscale image to enhance non-distinct features before thresholding the image in preparation for blob analysis.

Use `CWIMAQVision.GrayMorphology` to perform one of the following seven transformations:

- Erosion—Reduces the brightness of pixels that are surrounded by neighbors with a lower intensity.
- Dilation—Increases the brightness of pixels surrounded by neighbors with a higher intensity. A dilation has the opposite effect as an erosion.
- Opening—Removes bright pixels isolated in dark regions and smooths boundaries.



- Closing—Removes dark pixels isolated in bright regions and smooths boundaries.
- Proper-opening—Removes bright pixels isolated in dark regions and smooths the inner contours of particles.
- Proper-closing—Removes dark pixels isolated in bright regions and smooths the inner contours of particles.
- Auto-median—Generates simpler particles that have fewer details.

For more information about grayscale morphology transformations, see Chapter 5, *Image Processing*, of the *IMAQ Vision Concepts Manual*.

## FFT

Use the Fast Fourier Transform (FFT) to convert an image into its frequency domain. In an image, details and sharp edges are associated with mid to high spatial frequencies because they introduce significant gray-level variations over short distances. Gradually varying patterns are associated with low spatial frequencies.

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. Also, the imaging setup may produce non-uniform lighting of the field of view, which produces an image with a light drift superimposed on the information you want to analyze. In the frequency domain, the light drift appears as a limited set of low frequencies around the average intensity of the image (DC component).

You can use algorithms working in the frequency domain to isolate and remove these unwanted frequencies from your image. Follow these steps to obtain an image in which the unwanted pattern has disappeared but the overall features remain:

1. Use `CWIMAQVision.FFT` to convert an image from the spatial domain to the frequency domain. This method computes the Fast Fourier Transform (FFT) of the image and results in a complex image representing the frequency information of your image.
2. Improve your image in the frequency domain with a lowpass or highpass frequency filter. Specify which type of filter to use with `CWIMAQVision.CxAttenuate` or `CWIMAQVision.CxTruncate`. Lowpass filters smooth noise, details, textures, and sharp edges in an image. Highpass filters emphasize details, textures, and sharp edges in images, but they also emphasize noise.

- Lowpass attenuation—The amount of attenuation is directly proportional to the frequency information. At low frequencies, there is little attenuation. As the frequencies increase, the attenuation increases. This operation preserves all of the zero frequency information. Zero frequency information corresponds to the DC component of the image or the average intensity of the image in the spatial domain.
  - Highpass attenuation—The amount of attenuation is inversely proportional to the frequency information. At high frequencies, there is little attenuation. As the frequencies decrease, the attenuation increases. The zero frequency component is removed entirely.
  - Lowpass truncation—Specify a frequency. The frequency components above the ideal cutoff frequency are removed, and the frequencies below it remain unaltered.
  - Highpass truncation—Specify a frequency. The frequency components above the ideal cutoff frequency remain unaltered, and the frequencies below it are removed.
3. To transform your image back to the spatial domain, use `CWIMAQVision.InverseFFT`.

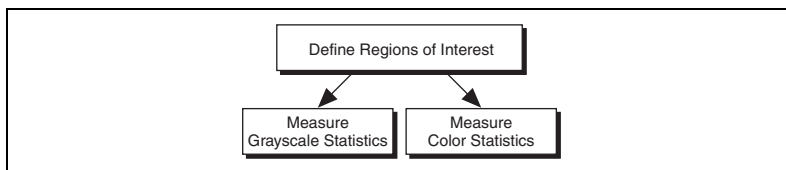
## Complex Image Operations

`CWIMAQVision.ReplaceComplexPlane` and `CWIMAQVision.ExtractComplexPlane` allow you to access, process, and update independently the magnitude, phase, real, and imaginary planes of a complex image. You can also convert a complex image to an array and back with `CWIMAQImage.ImageToArray` and `CWIMAQImage.ArrayToImage`.

---

# Grayscale and Color Measurements

This chapter describes how to take measurements from grayscale and color images. You can make inspection decisions based on image statistics, such as the mean intensity level in a region. Based on the image statistics, you can perform many machine vision inspection tasks on grayscale or color images, such as detecting the presence or absence of components, detecting flaws in parts, and comparing a color component with a reference. Figure 3-1 illustrates the basic steps involved in making grayscale and color measurements.



**Figure 3-1.** Steps to Taking Grayscale and Color Measurements

---

## Define Regions of Interest

A region of interest (ROI) is an area of an image in which you want to focus your image analysis. You can define an ROI interactively, programmatically, or with an image mask.

### Interactively Defining Regions

You can interactively define an ROI in a viewer that displays an image. Use the tools from the pop-up menu to interactively define and manipulate the ROIs. Table 3-1 describes each of the tools and the manner in which you use them.

**Table 3-1.** Tools Palette Functions

<b>Tool Name</b>	<b>Function</b>
None	Disable the tools.
Selection Tool	Select an ROI in the image and adjust the position of its control points and contours.  Action: Click on the desired ROI or control points.
Point	Select a pixel in the image.  Action: Click on the desired position.
Line	Draw a line in the image.  Action: Click on the initial position and click again on the final position.
Rectangle	Draw a rectangle or square in the image.  Action: Click on one corner and drag to the opposite corner.
Rotated Rectangle	Draw a rotated rectangle in the image.  Action: Click on one corner and drag to the opposite corner to create the rectangle. Then, click on the lines inside the rectangle and drag to adjust the rotation angle.
Oval	Draw an oval or circle in the image.  Action: Click on the center position and drag to the desired size.
Annulus	Draw an annulus in the image.  Action: Click on the center position and drag to the desired size. Adjust the inner and outer radii, and adjust the start and end angle.
Broken Line	Draw a broken line in the image.  Action: Click to place a new vertex and double-click to complete the ROI element.

**Table 3-1.** Tools Palette Functions (Continued)

Tool Name	Function
Polygon	Draw a polygon in the image. Action: Click to place a new vertex and double-click to complete the ROI element.
Freeline	Draw a freehand line in the image. Action: Click on the initial position, drag to the desired shape and release the mouse button to complete the shape.
Free Region	Draw a freehand region in the image. Action: Click on the initial position, drag to the desired shape and release the mouse button to complete the shape.
Zoom	Zoom in or zoom out in an image. Action: Click on the image to zoom in. Hold down <Shift> and click to zoom out.
Pan	Pan around an image. Action: Click on an initial position, drag to the desired position, and release the mouse button to complete the pan.

Hold down <Shift> when drawing an ROI if you want to constrain the ROI to the horizontal, vertical, or diagonal axes, when possible. Use the selection tool to position an ROI by its control points or vertices. ROIs are context sensitive, meaning that the cursor actions differ depending on the ROI with which you interact. For example, if you move your cursor over the side of a rectangle, the cursor changes to indicate that you can click and drag the side to resize the rectangle. If you want to draw more than one ROI in a window, hold down <Ctrl> while drawing additional ROIs.

In the status bar of the viewer, you can display tool information about the characteristics of ROIs you draw, as shown in Figure 3-2. Check the **Show Tool Info** check box on the Status Bar property page during design time, or set the `CWIMAQViewer.ShowToolInfo` property to `True` during run time to display tool information. You can also show or hide the tool information from the pop-up menu.

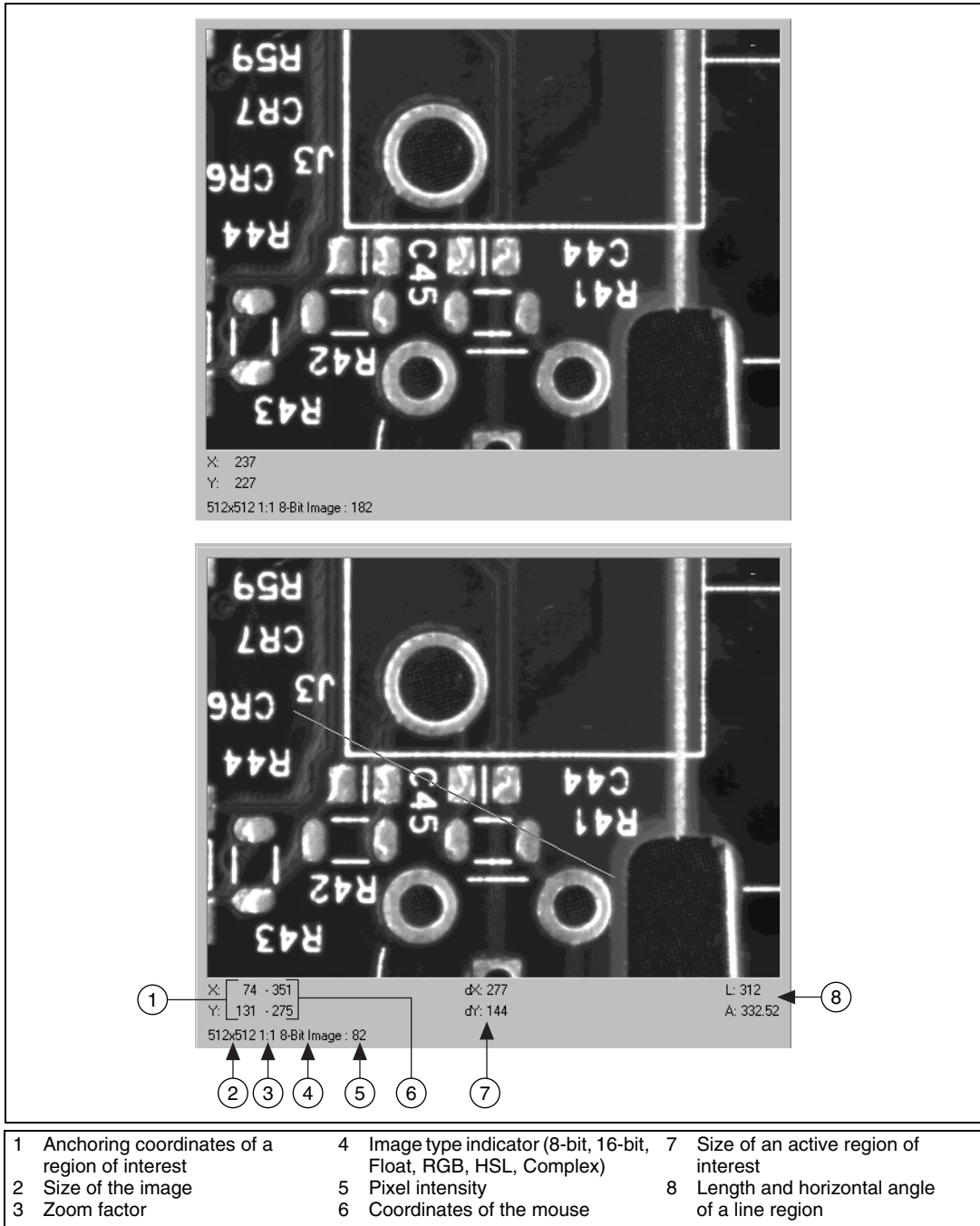


Figure 3-2. Tools Information

During design time, use the Menu property page to select which tools appear in the pop-up menu. You can also designate a default tool from this property page. During run time, set the `CWIMAQViewer.MenuItems` to select the tools to display, and set `CWIMAQViewer.Tool` to select the default tool.

## Programmatically Defining Regions

You can define ROIs programmatically using the `CWIMAQRegions` collection. Shapes in IMAQ Vision are represented by shape objects. For example, `CWIMAQPoint` represents a point, and `CWIMAQLine` represents a line. Use the following methods listed in Table 3-2 to add various shapes to your regions.

**Table 3-2.** Methods that Add Shapes to Regions

Method	Description
<code>AddPoint</code>	adds a point to the ROI
<code>AddLine</code>	adds a line to the ROI
<code>AddRectangle</code>	adds a rectangle to the ROI
<code>AddRotatedRectangle</code>	adds a rotated rectangle to the ROI
<code>AddOval</code>	adds an oval to the ROI
<code>AddAnnulus</code>	adds an annulus to the ROI
<code>AddBrokenLine</code>	adds a broken line to the ROI
<code>AddPolygon</code>	adds a polygon to the ROI
<code>AddFreeline</code>	adds a free line to the ROI
<code>AddFreeregion</code>	adds a free region to the ROI
<code>AddRegion</code>	adds a region object to the ROI

Use the `CWIMAQRegions.CopyTo` method to copy all the data from one `CWIMAQRegions` object to another.

You can define the regions on a viewer and access the regions using the `CWIMAQViewer.Regions` property.

The individual `CWIMAQRegion` objects provide access to the shapes in the collection. Each region has one shape object associated with it. Use the `CWIMAQRegion.Shape` property to determine what type of shape the

CWIMAQRegion contains. Once you know the type of shape that the region contains, you can set the region into a shape variable and use that variable to manipulate the shape's properties. For example, the following code resizes a rectangle selected on the viewer:

```
Dim MyRectangle As CWIMAQRectangle
Set MyRectangle = CWIMAQViewer1.Regions(1)
MyRectangle.Width = 100
MyRectangle.Height = 100
```

You can also pass CWIMAQRegion objects to any IMAQ Vision method that takes a shape as a parameter. However, if the CWIMAQRegion does not contain the type of shape object that the method requires, a type mismatch error results.

## Defining Regions with Masks

You can define regions to process with image masks. An image mask is an 8-bit image of the same size as or smaller than the image you want to process. Pixels in the mask image determine whether the corresponding pixel in the source image needs to be processed. If a pixel in the image mask has a value other than 0, the corresponding pixel in the source image is processed. If a pixel in the image mask has a value of 0, the corresponding pixel in the source image is left unchanged.

You can use a mask to define particles in a grayscale image when you need to make intensity measurements on those particles. First, threshold your image to make a new binary image. For more information on binary images, see Chapter 4, *Blob Analysis*. You can input the binary image or a labeled version of the binary image as a mask image to the intensity measurement method. If you want to make color comparisons, convert the binary image into a CWIMAQRegions collection using `CWIMAQVision.MaskToRegions`.



## Measure Grayscale Statistics

---

You can measure grayscale statistics in images using light meters or quantitative analysis methods. You can obtain the center of energy for an image with the centroid method.

Use `CWMachineVision.LightMeterPoint` to measure the light intensity at a point in the image. Use `CWMachineVision.LightMeterLine` to get the pixel value statistics along a line in the image, such as mean intensity, standard deviation, minimum intensity, and maximum intensity. Use `CWMachineVision.LightMeterRectangle` to get the pixel value statistics within a rectangular region in an image.

Use `CWIMAQVision.Quantify` to obtain the following statistics about the entire image or individual regions in the image: mean intensity, standard deviation, minimum intensity, maximum intensity, area, and the percentage of the image that you analyzed. You can specify regions in the image with a labeled image mask. A labeled image mask is a binary image that has been processed so that each region in the image mask has a unique intensity value. Use `CWIMAQVision.Label` to label your image mask.

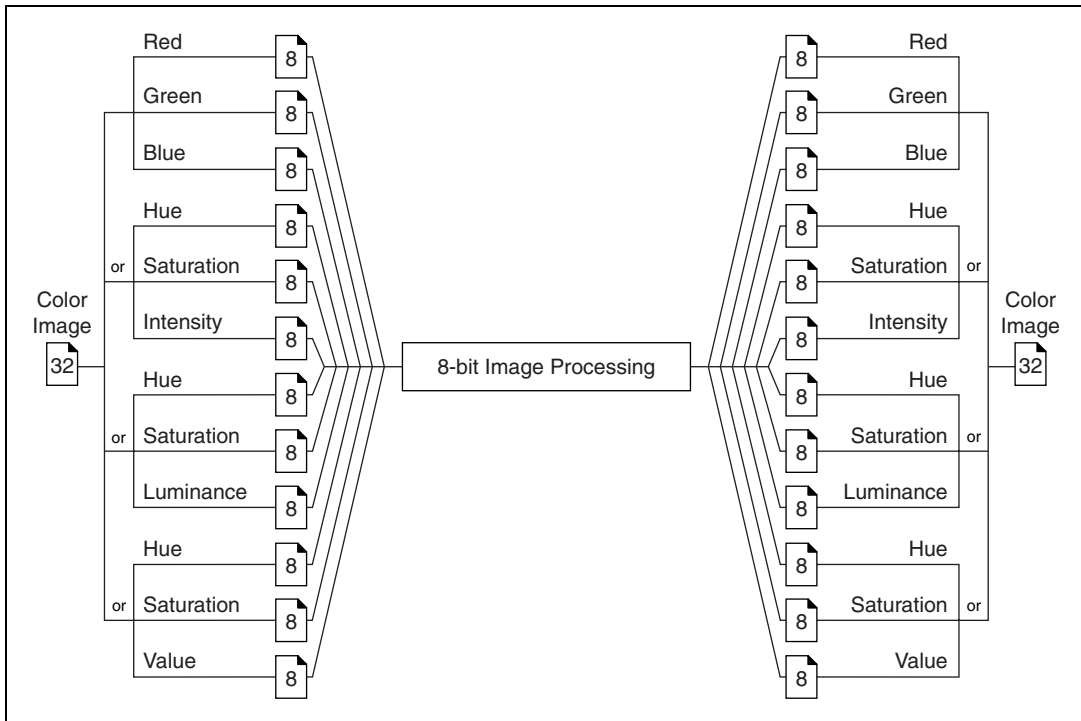
Use `CWIMAQVision.Centroid2` to compute the energy center of the image, or of a region within an image.

## Measure Color Statistics

---

Most image processing and analysis methods apply to 8-bit images. However, you can analyze and process individual components of a color image.

Using `CWIMAQVision.ExtractColorPlanes`, you can break down a color image into various sets of primary components, such as RGB (Red, Green, and Blue), HSI (Hue, Saturation, and Intensity), HSL (Hue, Saturation, and Luminance), or HSV (Hue, Saturation, and Value). Each component becomes an 8-bit image that you can process like any other grayscale image. Use `CWIMAQVision.ExtractSingleColorPlane` to extract a single color plane from an image. Use `CWIMAQVision.ReplaceColorPlanes` to reassemble a color image from a set of three 8-bit images, where each image becomes one of the three primary components. Figure 3-3 illustrates how a color image breaks down into its three components.



**Figure 3-3.** Primary Components of a Color Image

A color pixel encoded as a Long value can be decomposed into its individual components using `CWIMAQVision.IntegerToColorValue`. You can convert a pixel value represented in any color model into its components in any other color model using `CWIMAQVision.ColorValueConversion`.

## Comparing Colors

You can use the color matching capability of IMAQ Vision to compare or evaluate the color content of an image or regions in an image.

Follow these steps to compare colors using color matching:

1. Select an image containing the color information that you want to use as a reference. The color information can consist of a single color or multiple dissimilar colors, such as red and blue.
2. Use the entire image or regions in the image to learn the color information using `CWIMAQVision.LearnColor`, which stores the results of the operation in a `CWIMAQColorInformation` object that

you supply as a parameter. The color information object has a color spectrum that contains a compact description of the color information that you learned. See Chapter 14, *Color Inspection*, of the *IMAQ Vision Concepts Manual* for more information. Use the `CWIMAQColorInformation` object to represent the learned color information for all subsequent matching operations.

3. Define an entire image, a region, or multiple regions in an image as the inspection or comparison area.
4. Use `CWIMAQVision.MatchColor` to compare the learned color information to the color information in the inspection regions. This method returns an array of scores that indicates how close the matches are to the learned color information.
5. Use the color matching score as a measure of similarity between the reference color information and the color information in the image regions being compared.

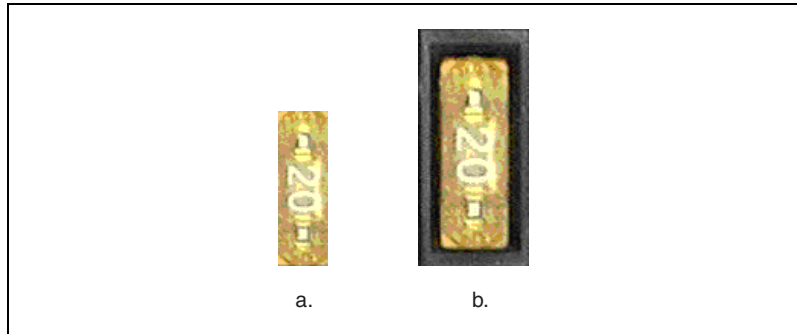
## Learning Color Information

When learning color information, you should choose the color information carefully:

- Specify an image or regions in an image that contain the color or color set that you want to learn.
- Select how detailed you want the color information to be learned.
- Choose colors that you want to ignore during matching.

## Choosing the Right Color Information

Because color matching only uses color information to measure similarity, the image or regions in the image representing the object should contain only the significant colors that represent the object, as shown in Figure 3-4a. Figure 3-4b illustrates an unacceptable region containing background colors.



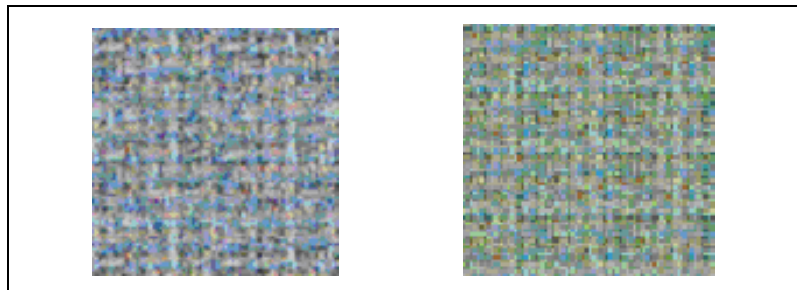
**Figure 3-4.** Template Color Information

## Specifying the Color Information to Learn

You can learn the color information associated with an entire image, a region in an image, or multiple regions in an image.

### Using the Entire Image

You can use an entire image to learn the color spectrum that represents the entire color distribution of the image. In a fabric identification application, for example, an entire image can specify the color information associated with a certain fabric type, as shown in Figure 3-5.



**Figure 3-5.** Using the Entire Image to Learn Color Distribution

### Using a Region in the Image

You can select a region in the image to provide the color information for comparison. A region is helpful for pulling out the useful color information in an image. Figure 3-6 shows an example of using a region that contains the color information that is important for the application.



**Figure 3-6.** Using a Single Region to Learn Color Distribution

## Using Multiple Regions in the Image

The interaction of light with an object's surface creates the observed color of that object. The color of a surface depends on the directions of illumination and the direction from which the surface is observed. Two identical objects may have different appearances because of a difference in positioning or a change in the lighting conditions. Figure 3-7 shows how light reflects differently off of the 3D surfaces of the fuses, resulting in slightly different colors for identical fuses. (Compare the 3 amp fuse in the upper row with the 3 amp fuse in the lower row.) This results in different color spectra for identical fuses.

If you learn the color spectrum by drawing a region of interest around the 3 amp fuse in the upper row, and then do a color matching for the 3 amp fuse in the upper row, you get a very high match score for it (close to 1000). But the match score for the 3 amp fuse in the lower row is quite low (around 500). This problem could cause a mismatch for the color matching in a fuse box inspection process.

IMAQ Vision's color learning software uses a clustering process to find the representative colors from the color information specified by one or multiple regions in the image. To create a representative color spectrum for all 3 amp fuses in the learning phase, draw a Region around the 3 amp fuse in the upper row, hold down <Ctrl>, and draw another Region around the 3 amp fuse in the lower row. The new color spectrum represents 3 amp fuses much better and results in high match scores (around 800) for both the fuses. Use as many samples as you want in an image to learn the representative color spectrum for a specified template.

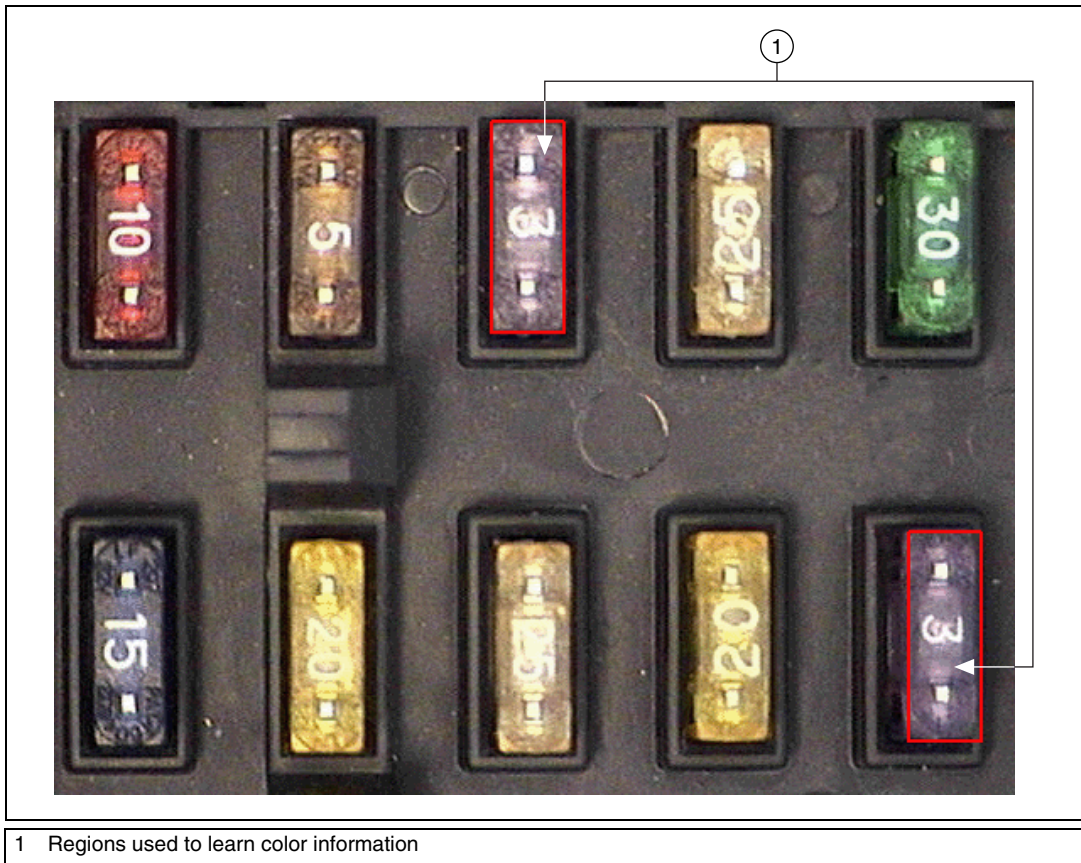


Figure 3-7. Using Multiple Regions to Learn Color Distribution

### Choosing a Color Representation Sensitivity

When you learn a color, you need to specify the granularity required to specify the color information. An image containing a few well-separated colors in the color space requires a lower granularity to describe the color than an image that contains colors that are close to one another in the color space. Use the `ColorSensitivity` parameter of `CWIMAQVision.LearnColor` to specify the granularity you want to use to represent the colors. For more information about color sensitivity, see the *Color Sensitivity* section of Chapter 5, *Machine Vision*.

## Ignoring Learned Colors

Ignore certain color components in color matching by setting the corresponding component in the input color spectrum array to  $-1$ . To set a particular color component, follow these steps:

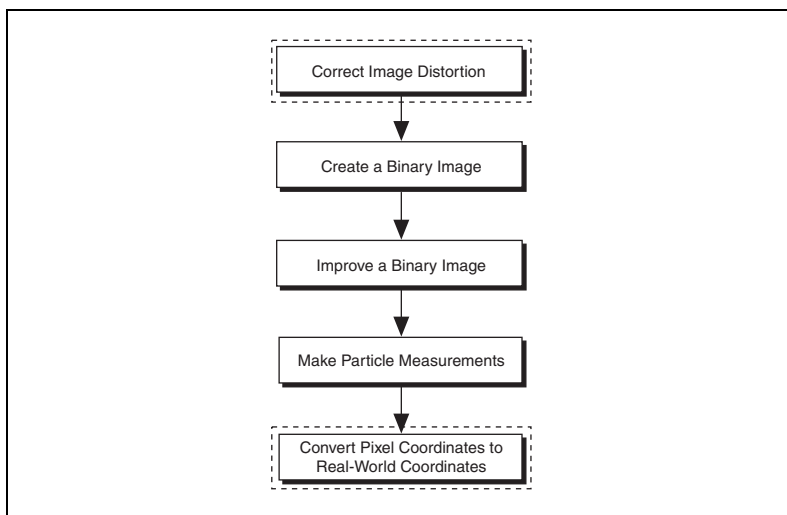
1. Copy `CWIMAQColorInformation.ColorSpectrum`, or create your own array.
2. Set the corresponding components of the array.
3. Assign this array to `CWIMAQColorInformation.ColorSpectrum` on the `CWIMAQColorInformation` object you want to use as input during the match phase.

For example, by setting the last component in the color spectrum to  $-1$ , color matching ignores the color white. By setting the second to last component in the color spectrum array to  $-1$ , color matching ignores the color black. To ignore other color components in color matching, determine the index to the color spectrum by locating the corresponding bins in the color wheel, where each bin corresponds to a component in the color spectrum array. Ignoring certain colors such as the background color results in a more accurate color matching score. Ignoring the background color also provides more flexibility when defining the regions of interest in the color matching process. Ignoring certain colors, such as the white color created by glare on a metallic surface, also improves the accuracy of the color matching. Experiment learning the color information on different parts of the images to determine which colors to ignore. For more information about the color wheel and color bins, see Chapter 14, *Color Inspection*, in the *IMAQ Vision Concepts Manual*.

# Blob Analysis

This chapter describes how to perform blob (Binary Large Object) analysis on your images. Use blob analysis to find statistical information about blobs, such as the presence, size, number, and location of blob regions. With this information, you can perform many machine vision inspection tasks, such as detecting flaws on silicon wafers or detecting soldering defects on electronic boards. Examples of how blob analysis can help you perform web inspection tasks include locating structural defects on wood planks or detecting cracks on plastic sheets.

Figure 4-1 illustrates the steps involved in performing blob analysis. Diagram items enclosed with dashed lines are optional steps.



**Figure 4-1.** Steps to Performing Blob Analysis



## Correct Image Distortion

---

If you need to make accurate shape measurements based on the blobs in an image containing perspective and nonlinear distortion errors, correct the distortion using the calibration information you attached to your image. Use `CWIMAQVision.CorrectCalibratedImage` to correct distortion in your grayscale image before thresholding it. See Chapter 6, [Calibration](#), for more information about correcting an image using calibration information.

## Create a Binary Image

---

Threshold your grayscale or color image to create a binary image. Creating a binary image separates the objects that you want to inspect from the background. The threshold operation sets the background pixels to 0 in the binary image, while setting the object pixels to a non-zero value. Object pixels have a value of 1 by default, but you can set the object pixels to any value or retain their original value.

You can use different techniques to threshold your image. If all the objects of interest in your grayscale image fall within a continuous range of intensities and you can specify this threshold range manually, use `CWIMAQVision.Threshold` to threshold your image.

If all the objects in your grayscale image are either brighter or darker than your background, you can use `CWIMAQVision.AutoThreshold` to automatically determine the optimal threshold range and threshold your image. Automatic thresholding techniques offer more flexibility than simple thresholds based on fixed ranges. Because automatic thresholding techniques determine the threshold level according to the image histogram, the operation is more independent of changes in the overall brightness and contrast of the image than a fixed threshold. These techniques are more resistant to changes in lighting, which makes them well suited for automated inspection tasks.

If your grayscale image contains objects that have multiple discontinuous grayscale values, use `CWIMAQVision.MultiThreshold2` to specify multiple threshold ranges.

If you need to threshold a color image, use `CWIMAQVision.ColorThreshold`. You must specify threshold ranges for each of the color planes (either Red, Green, and Blue or Hue, Saturation,

and Luminance). The binary image resulting from a color threshold is an 8-bit binary image.

## Improve the Binary Image

---

After you threshold your image, you may want to improve the resulting binary image with binary morphology. You can use primary binary morphology or advanced binary morphology to remove unwanted blobs, separate connected blobs, or improve the shape of blobs. Primary morphology methods work on the image as a whole by processing pixels individually. Advanced morphology operations are built upon the primary morphological operators and work on particles as opposed to pixels.



**Note** The terms *blob* and *particle* are used interchangeably in this chapter.

The advanced morphology methods that improve binary images require that you specify the type of connectivity to use. Connectivity specifies how IMAQ Vision determines whether two adjacent pixels belong to the same particle. If you have a blob that contains narrow areas, use connectivity-8 to ensure that the software recognizes the connected pixels as one blob. If you have two blobs that touch at one point, use connectivity-4 to ensure that the software recognizes the pixels as two separate blobs. For more information about connectivity, see Chapter 9, *Binary Morphology*, of the *IMAQ Vision Concepts Manual*.



**Note** Use the same type of connectivity throughout your application.

## Removing Unwanted Blobs

Use `CWIMAQVision.RejectBorder` to remove blobs that touch the border of the image. Reject blobs on the border of the image when you suspect that the information about those blobs is incomplete.

Use `CWIMAQVision.RemoveParticle` to remove large or small particles that do not interest you. You can also use the `Erode`, `Open`, and `POpen` methods in `CWIMAQVision.Morphology` to remove small particles. Unlike `CWIMAQVision.RemoveParticle`, these three methods alter the size and shape of the remaining blobs.

Use the hit-miss method of `CWIMAQVision.Morphology` to locate particular configurations of pixels, which you define with a structuring element. Depending on the configuration of the structuring element, the hit-miss method can locate single isolated pixels, cross-shape or

longitudinal patterns, right angles along the edges of particles, and other user-specified shapes. For more information about structuring elements, see Chapter 9, *Binary Morphology*, of the *IMAQ Vision Concepts Manual*.

If you know enough about the shape features of the blobs you want to keep, use `CWIMAQVision.ParticleFilter` to filter out particles that do not interest you. If you do not have enough information about the particles you want to keep at this point in your processing, use the particle measurement methods to obtain this information before applying a particle filter. See the *Make Particle Measurements* section for more information about the measurement methods.

## Separating Touching Blobs

Use `CWIMAQVision.Separation` or apply an erosion or an open operation with `CWIMAQVision.Morphology` to separate touching objects. `CWIMAQVision.Separation` is an advanced operation that separates blobs without modifying their shapes. However, erosion and open operations alter the shape of all the blobs.



**Note** A separation is a time-intensive operation compared to an erosion or open operation. Consider using an erosion if speed is an issue with your application.

## Improving Blob Shapes

Use `CWIMAQVision.FillHole` to fill holes in the blobs. Use `CWIMAQVision.Morphology` to perform a variety of operations on the blobs. You can use the open, close, proper open, proper close, and auto-median operations to smooth the boundaries of the blobs. Open and proper open smooth the boundaries of the blob by removing small isthmuses, while close widens the isthmuses. Close and proper close fill small holes in the blob. Auto-median removes isthmuses and fills holes. For more information about these operations, see Chapter 9, *Binary Morphology*, in the *IMAQ Vision Concepts Manual*.

## Make Particle Measurements

---

After you create a binary image and improve it, you can make particle measurements. With these measurements you can determine the location of blobs and their shape features. Use the following methods to perform particle measurements:

- `CWIMAQVision.BasicParticle`—This method returns the number of blobs in an image and a report containing the pixel area, real-world

area, and bounding rectangle of the blobs. You can use the bounding rectangle to determine the location of the blob in the image.

- `CWIMAVision.Particle`—This method returns the number of blobs in an image and a report containing 16 of the most commonly used measurements, including the area, bounding rectangle, and perimeter of blobs.
- `CWIMAVision.SelectParticles`—This method selects information about blobs from the reports generated by `CWIMAVision.BasicParticle` or `CWIMAVision.Particle`. Blobs that do not meet the criteria you set are filtered from the reports.
- `CWIMAVision.CalculateCoefficients`—This method uses the `CWIMAQFullParticleReport` objects from `CWIMAVision.Particle` or `CWIMAVision.SelectParticles` to calculate 50 particle measurements.

Table 4-1 lists all of the measurements that `CWIMAVision.CalculateCoefficients` returns.

**Table 4-1.** Particle Measurements

Measurement	Description
Area	area of the particle in pixels
AreaCalibrated	area of the particle in user-defined units
HoleNumber	number of holes within the particle
HoleArea	total area of the particle holes in pixels
AreaTotal	area of the particle and its holes
AreaScanned	area of the entire image in real-world units
RatioAreaScanned	ratio, expressed as a percent, of the surface area of a particle in relation to the image area
RatioAreaTotal	ratio, expressed as a percent, of the surface area of a particle in relation to the total area of the particle
CenterMassX	x-coordinate of the center of mass
CenterMassY	y-coordinate of the center of mass
RectLeft	left x-coordinate of the bounding rectangle
RectTop	top y-coordinate of the bounding rectangle
RectRight	right x-coordinate of the bounding rectangle

**Table 4-1.** Particle Measurements (Continued)

<b>Measurement</b>	<b>Description</b>
RectBottom	bottom y-coordinate of bounding rectangle
RectWidth	width of bounding rectangle in user-defined units
RectHeight	height of bounding rectangle in user-defined units
MaxSegment	length of longest horizontal line segment in a particle
MaxSegmentX	leftmost x-coordinate of longest horizontal line segment in a particle
MaxSegmentY	y-coordinate of longest horizontal line segment
Perimeter	length of the outer contour of the particle in user-defined units
HolePerimeter	perimeter of all holes in user-defined units
SigmaX	sum of the x-coordinates for each pixel of the particle
SigmaY	sum of the y-coordinates for each pixel of the particle
SigmaXX	sum of the squared x-coordinates for each pixel of the particle
SigmaYY	sum of the squared y-coordinates for each pixel of the particle
SigmaXY	sum of the product of the x-coordinate and y-coordinate for each pixel of the particle
ProjectionX	sum of the horizontal segments that do not superimpose any other vertical segment
ProjectionY	sum of the vertical segments that do not superimpose any other horizontal segment
IXX	inertia matrix coefficient in XX
IYY	inertia matrix coefficient in YY
IXY	inertia matrix coefficient in XY
MeanChordX	mean length of horizontal segments
MeanChordY	mean length of vertical segments
MaxIntercept	length of the longest segment in the convex hull of the particle
MeanIntercept	mean length of the chords in a particle perpendicular to its max intercept
Orientation	direction of the longest segment

**Table 4-1.** Particle Measurements (Continued)

Measurement	Description
EquEllipseMinor	total length of the minor axis of the ellipse having the same area as the particle and a major axis equal to half the max intercept
EllipseMajor	total length of the major axis of the ellipse having the same area and perimeter as the particle in user-defined units
EllipseMinor	total length of the minor axis of the ellipse having the same area and perimeter as the particle in user-defined units
RatioEquEllipse	fraction of the length of the major axis to the length of the minor axis of the ellipse having the same area and perimeter as the particle in user-defined units
RectBigSide	length of the long side of a rectangle having the same area and perimeter as the particle in user-defined units
RectSmallSide	length of the short side of a rectangle having the same area and perimeter as the particle in user-defined units
RatioRect	ratio of rectangle long side to rectangle short side
Elongation	max intercept/mean perpendicular intercept
Compactness	particle area/(height × width)
Heywood	particle perimeter/perimeter of circle having same area as the particle
TypeFactor	complex factor relating the surface area to the moment of inertia
HydraulicRadius	particle area/particle perimeter
WaddelDisk	diameter of the disk having the same area as the particle in user-defined units
Diagonal	diagonal of an equivalent rectangle in user-defined units

## Convert Pixel Coordinates to Real-World Coordinates

If you need to find the location of the center of mass or the bounding rectangle of the blobs in real-world units, use  
`CWIMAQVision.ConvertPixelToRealWorldCoordinates.`

---

# Machine Vision

This chapter describes how to perform many common machine vision inspection tasks. The most common inspection tasks are detecting the presence or absence of parts in an image and measuring the dimensions of parts to see if they meet specifications.

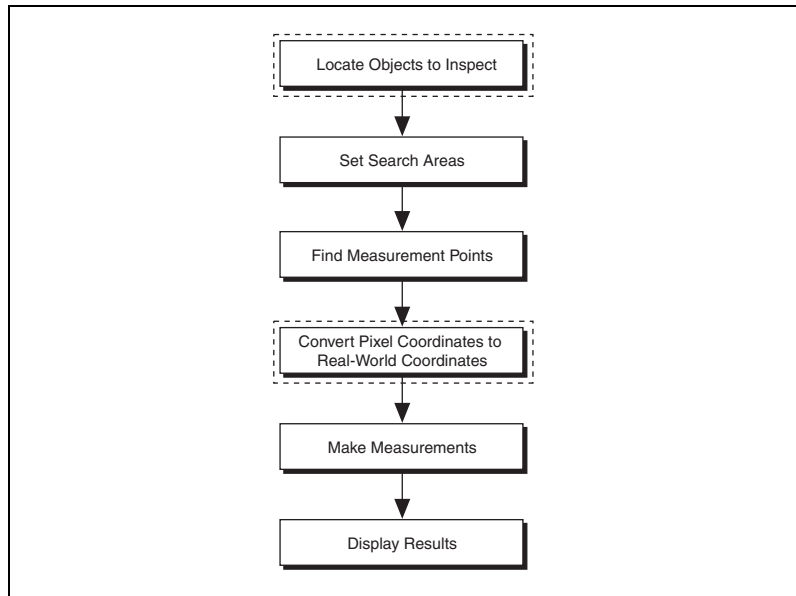
Measurements are based on characteristic features of the object represented in the image. Image processing algorithms traditionally classify the type of information contained in an image as edges, surfaces and textures, or patterns. Different types of machine vision algorithms leverage and extract one or more types of information.

Edge detectors and derivative techniques—such as rakes, concentric rakes, and spokes—use edges represented in the image. They locate with high accuracy the position of the edge of an object in the image. For example, you can use the edge location to measure the width of the part (a technique called clamping). You can combine multiple edge locations to compute intersection points, projections, circles, or ellipse fits.

Pattern matching algorithms use edges and patterns. Pattern matching can locate with very high accuracy the position of fiducials or characteristic features of the part under inspection. Those locations can then be combined to compute lengths, angles, and other object measurements.

The robustness of your measurement relies on the stability of your image acquisition conditions. Sensor resolution, lighting, optics, vibration control, part fixture, and general environment are key components of the imaging setup. All the elements of the image acquisition chain directly affect the accuracy of the measurements.

Figure 5-1 illustrates the basic steps involved in performing machine vision. Diagram items enclosed with dashed lines are optional steps.



**Figure 5-1.** Steps to Performing Machine Vision

## Locate Objects to Inspect

---

In a typical machine vision application, you extract measurements from regions of interest rather than the entire image. To use this technique, the parts of the object you are interested in must always appear inside the regions of interest you define.

If the object under inspection is always at the same location and orientation in the images you need to process, defining regions of interest is simple. See the [Set Search Areas](#) section of this chapter for information about selecting a region of interest.

Often, the object under inspection appears rotated or shifted in the image you need to process with respect to the reference image in which you located the object. When this occurs, the regions of interest (ROIs) need to shift and rotate with the parts of the object in which you are interested. In order for the ROIs to move with the object, you need to define a reference coordinate system relative to the object in the reference image. During the measurement process, the coordinate system moves with the object when it



appears shifted and rotated in the image you need to process. This coordinate system is referred to as the measurement coordinate system. The measurement methods automatically move the ROIs to the correct position using the position of the measurement coordinate system with respect to the reference coordinate system. For information about coordinate systems, see Chapter 13, *Dimensional Measurements*, of the *IMAQ Vision Concepts Manual*.

You can build a coordinate transformation using edge detection or pattern matching. The output of the edge detection and pattern matching methods that build a coordinate transformation is a `CWMVCoordinateTransformation` object, which contains a reference coordinate system and a measurement coordinate system. Some machine vision methods take this transformation and adjust the regions of inspection automatically. You can also use these outputs to move the regions of inspection relative to the object programmatically.

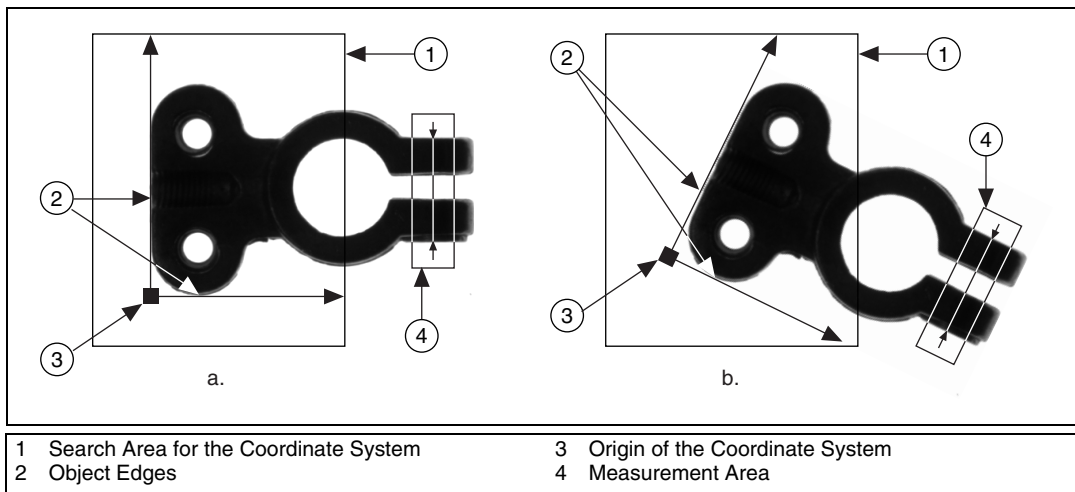
## Using Edge Detection to Build a Coordinate Transformation

You can build a coordinate transformation using two edge detection techniques. Use `CWMachineVision.FindCoordTransformUsingRect` to define a reference coordinate system using one rectangular region. Use `CWMachineVision.FindCoordTransformUsingTwoRects` to define a reference coordinate system using two independent rectangular regions. Follow the steps below to build a coordinate transformation using edge detection:



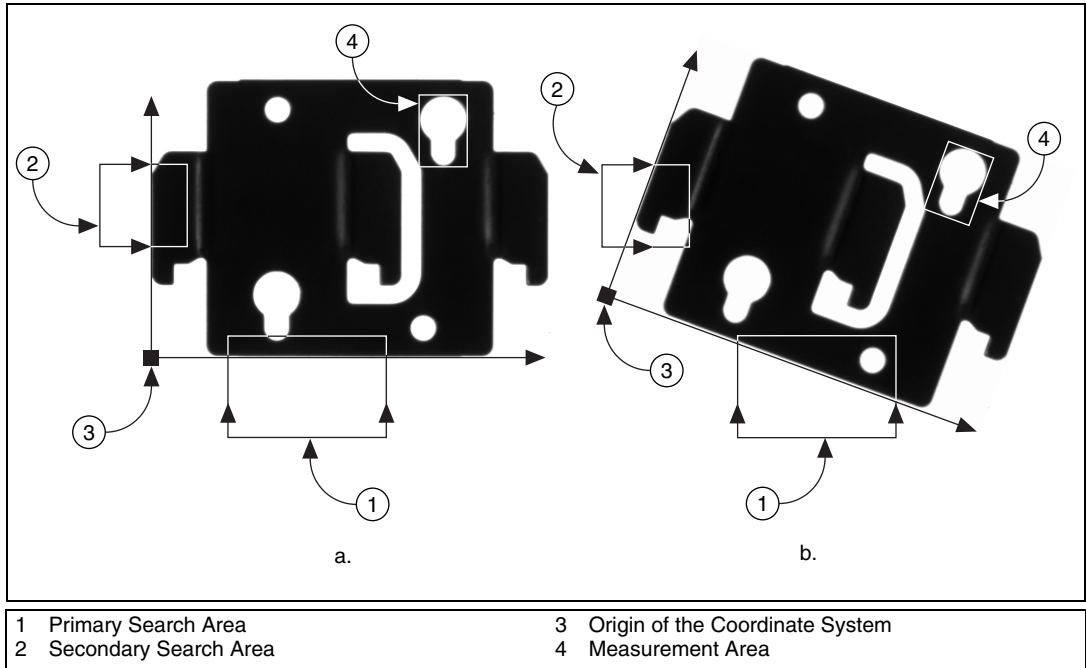
**Note** To use this technique, the object cannot rotate more than  $\pm 65^\circ$  in the image.

1. Specify one or two rectangular regions.
  - a. If you use `CWMachineVision.FindCoordTransformUsingRect`, specify one rectangular ROI that includes part of two straight, nonparallel boundaries of the object, as shown in Figure 5-2. This rectangular region must be large enough to include these boundaries in all the images you want to inspect.



**Figure 5-2.** Coordinate Systems of a Reference Image and Inspection Image

- b. If you use `CWMachineVision.FindCoordTransformUsingTwoRects`, specify two rectangular ROIs, each containing one separate, straight boundary of the object, as shown in Figure 5-3. The boundaries cannot be parallel. The regions must be large enough to include the boundaries in all the images you want to inspect.



**Figure 5-3.** Locating Coordinate System Axes with Two Search Areas

2. Choose the parameters you need to locate the edges on the object.
3. Choose the coordinate system axis direction.
4. Choose the results that you want to overlay onto the image.
5. Choose the mode for the method. To build a coordinate transformation for the first time, set the `FirstRun` parameter to `True`. To update the coordinate transformation in subsequent images, set this parameter to `False`.

## Using Pattern Matching to Build a Coordinate Transformation

You can build a coordinate transformation using pattern matching. Use `CWMachineVision.FindCoordTransformUsingPattern` to define a reference coordinate system based on the location of a reference feature. Use this technique when the object under inspection does not have straight, distinct edges. Follow the steps below to build a coordinate transformation using pattern matching:



**Note** The object may rotate  $\pm 360^\circ$  in the image using this technique if you use rotation-invariant pattern matching.

1. Define a template that represents the part of the object that you want to use as a reference feature. For more information about defining a template, see the [Find Measurement Points](#) section.
2. Define a rectangular search area in which you expect to find the template.
3. Set the `MatchMode` property of the `CWMVFindCTUsingPatternOptions` object to `cwimaqRotationInvariant` when you expect your template to appear rotated in the inspection images. Otherwise, set it to `cwimaqShiftInvariant`.
4. Choose the results that you want to overlay onto the image.
5. Choose the mode for the method. To build a transformation for the first time, set the `FirstRun` parameter to `True`. To update the transformation in subsequent images, set this parameter to `False`.

## Choosing a Method to Build the Coordinate Transformation

The following flowchart guides you through choosing the best method for building a coordinate transformation for your application.

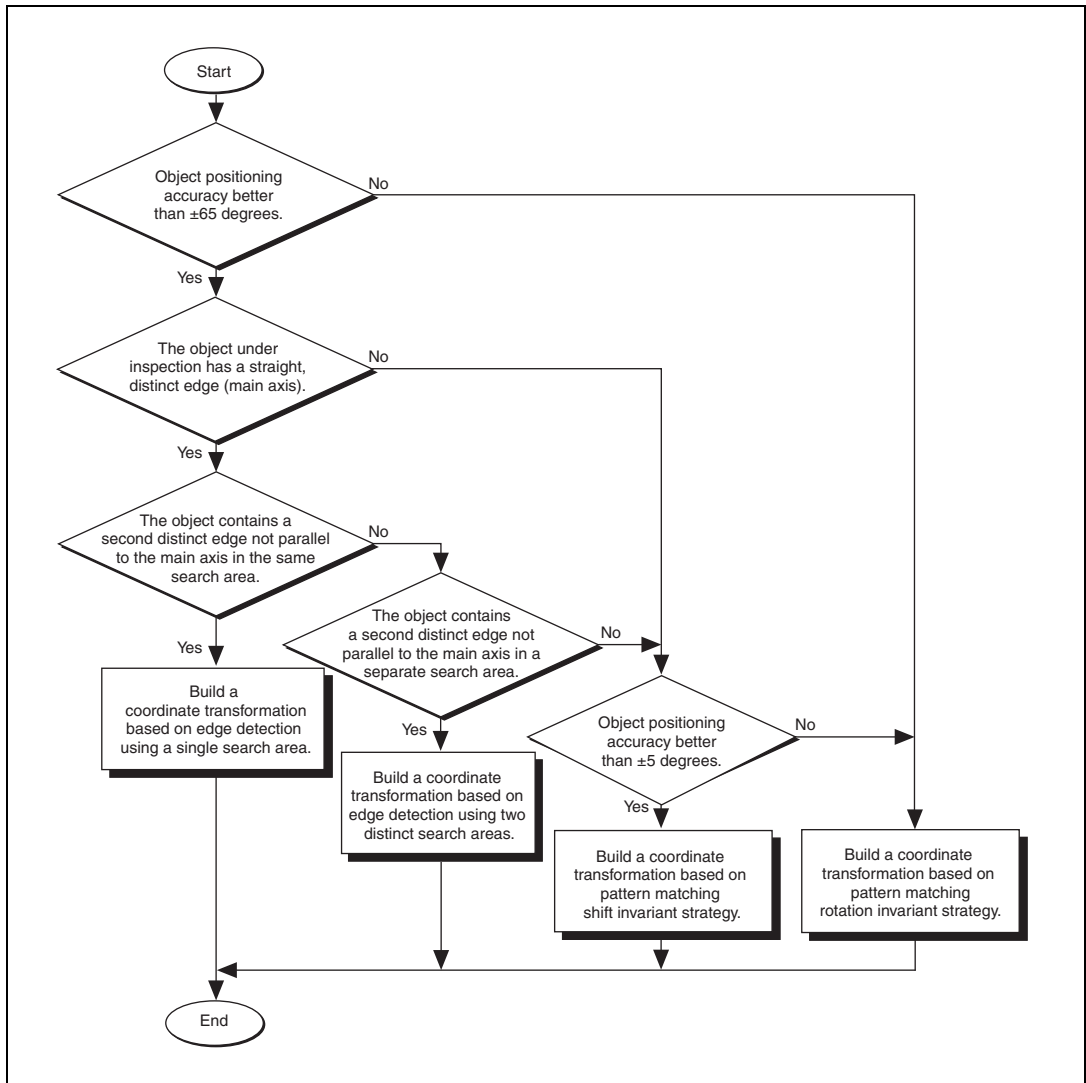


Figure 5-4. Building a Coordinate Transformation

# Set Search Areas

Select regions of interest (ROIs) in your images to limit the areas in which you perform your processing and inspection. You can define ROIs interactively or programmatically.

## Interactively Defining Regions

Follow these steps to interactively define an ROI:

1. Call `CWMachineVision.SetupViewerFor<shapename>Selection`. The following <shapename> values are available: `Annulus`, `Line`, `Point`, `Rectangle`, and `RotatedRect`. This method configures the viewer to display the appropriate tools for the shape you want to select.
2. Draw an ROI on your image. Resize or reposition the ROI until it defines the area of the image you want to process.
3. Use `CWMachineVision.GetSelected<shapename>FromViewer` to programmatically retrieve the shape from the viewer.

You can also use the techniques described in Chapter 3, *Grayscale and Color Measurements*, to select an ROI.

Table 5-1 indicates which ROI selection methods to use with a given `CWMachineVision` method:

**Table 5-1.** ROI Selection Methods to Use with `CWMachineVision` Methods

<b>CWMachineVision ROI Selection Methods</b>	<b>CWMachineVision Method</b>
<code>SetupViewerForRotatedRectSelection</code> <code>GetSelectedRotatedRectFromViewer</code>	<code>FindPattern</code> <code>MeasureMaximumDistance</code> <code>MeasureMinimumDistance</code> <code>FindStraightEdge</code> <code>LightMeterRectangle</code>
<code>SetupViewerForAnnulusSelection</code> <code>GetSelectedAnnulusFromViewer</code>	<code>FindCircularEdge</code> <code>FindConcentricEdge</code>
<code>SetupViewerForPointSelection</code> <code>GetSelectedPointFromViewer</code>	<code>LightMeterPoint</code>
<code>SetupViewerForLineSelection</code> <code>GetSelectedLineFromViewer</code>	<code>LightMeterLine</code>

## Programmatically Defining Regions

When you have an automated application, you need to define regions of interest programmatically. You can programmatically define regions by providing basic parameters that describe the region you want to define. You can specify a rotated rectangle by creating a `CWIMAQRotatedRectangle` object and setting the coordinates of the center, width, height, and rotation angle. You can specify an annulus by creating a `CWIMAQAnnulus` object and setting the coordinates of the center, inner radius, outer radius, start angle, and end angle. You can specify a point by setting its x-coordinates and y-coordinates. You can specify a line by setting the coordinates of the start and end points.

See Chapter 3, *Grayscale and Color Measurements*, for more information about defining regions of interest.

## Find Measurement Points

---

After you set regions of inspection, locate points within those regions on which you can base measurements. You can locate measurement points using edge detection, pattern matching, color pattern matching, and color location.

## Finding Features Using Edge Detection

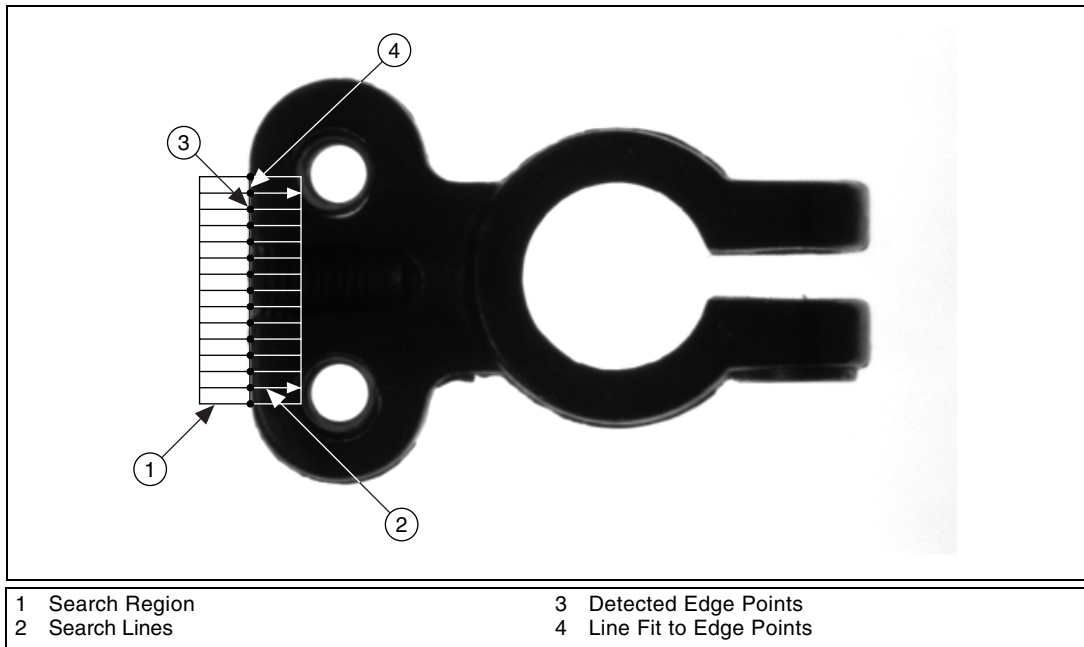
Use the edge detection tools to identify and locate sharp discontinuities in an image. Discontinuities typically represent abrupt changes in pixel intensity values, which characterize the boundaries of objects.

### Finding Lines or Circles

If you want to find points along the edge of an object and find a line describing the edge, use `CWMachineVision.FindStraightEdge` and `CWMachineVision.FindConcentricEdge`.

`CWMachineVision.FindStraightEdge` finds edges based on rectangular search areas, as shown in Figure 5-5.

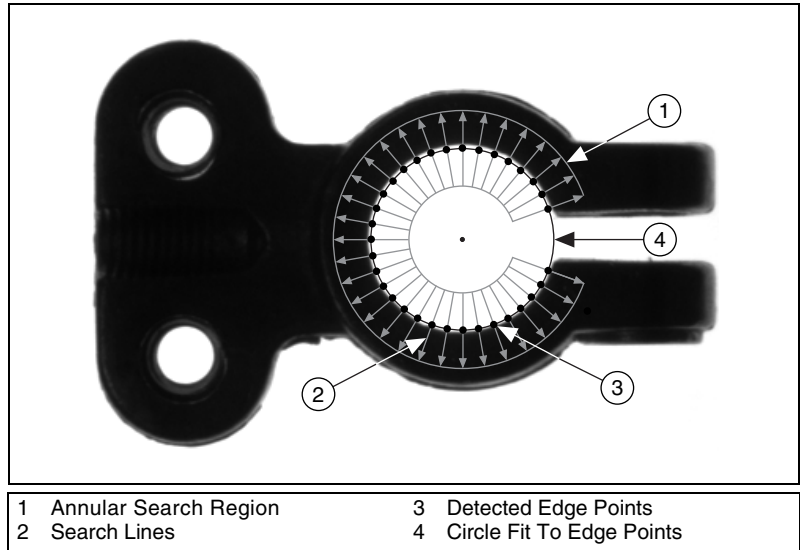
`CWMachineVision.FindConcentricEdge` finds edges based on annular search areas.



**Figure 5-5.** Finding a Straight Feature

If you want to find points along a circular edge and find the circle that best fits the edge, as shown in Figure 5-6, use `CWMachineVision.FindCircularEdge`.





**Figure 5-6.** Finding a Circular Feature

These methods locate the intersection points between a set of search lines within the search region and the edge of an object. Specify the separation between the lines that the methods use to detect edges. The methods determine the intersection points based on their contrast, width, and steepness. The software calculates a best-fit line with outliers rejected or a best-fit circle through the points it found. The methods return the coordinates of the edges found.

## Finding Edge Points Along One Search Contour

Use `CWIMAVision.SimpleEdge` and `CWIMAVision.FindEdges2` to find edge points along a contour. You can find the first edge, last edge, or all edges along the contour. Use `CWIMAVision.SimpleEdge` when your image contains little noise and the object and background are clearly differentiated. Otherwise, use `CWIMAVision.FindEdges2`.

These methods require you to input the coordinates of the points along the search contour. Use `CWIMAVision.RegionsProfile` to obtain the coordinates from a `CWIMAVision.Regions` object that describes the contour. If you have a straight line, use `CWIMAVision.GetPointsOnLine` to obtain the points along the line instead of using regions.

These methods determine the edge points based on their contrast and slope. You can specify whether you want to find the edge points using subpixel accuracy.

## Finding Edge Points Along Multiple Search Contours

Use the `CWIMAQVision.Rake`, `CWIMAQVision.Spoke`, and `CWIMAQVision.ConcentricRake` methods to find edge points along multiple search contours. These methods behave like `CWIMAQVision.FindEdges2`, but they find edges on multiple contours. These methods find only the first edge that meets the criteria along each contour. Pass in a `CWIMAQRegions` object to define the search region for these methods.

`CWIMAQVision.Rake` works on a rectangular search region. The search lines are drawn parallel to the orientation of the rectangle. Control the number of search lines in the region by specifying the distance, in pixels, between each line. Specify the search direction as left to right or right to left for a horizontally oriented rectangle. Specify the search direction as top to bottom or bottom to top for a vertically oriented rectangle.

`CWIMAQVision.Spoke` works on an annular search region, scanning the search lines that are drawn from the center of the region to the outer boundary and that fall within the search area. Control the number of lines in the region by specifying the angle, in degrees, between each line. Specify the search direction as either going from the center outward or from the outer boundary to the center.

`CWIMAQVision.ConcentricRake` works on an annular search region. The concentric rake is an adaptation of the rake to an annular region. Edge detection is performed along search lines that occur in the search region and that are concentric to the outer circular boundary. Control the number of concentric search lines that are used for the edge detection by specifying the radial distance between the concentric lines in pixels. Specify the direction of the search as either clockwise or counterclockwise.

## Finding Points Using Pattern Matching

The pattern matching algorithms in IMAQ Vision measure the similarity between an idealized representation of a feature, called a template, and the feature that may be present in an image. A feature is defined as a specific pattern of pixels in an image. Pattern matching returns the location of the center of the template and the template orientation. Follow these generalized steps to find features in an image using pattern matching:

1. Define a reference or fiducial pattern in the form of a template image.
2. Use the reference pattern to train the pattern matching algorithm with `CWIMAQVision.LearnPattern`.

3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.
4. Set the tolerances and parameters to specify how the algorithm operates at run time using `CWIMAQMatchPatternOptions`.
5. Test the search algorithm on test images using `CWIMAQVision.MatchPattern`.
6. Verify the results using a ranking method.

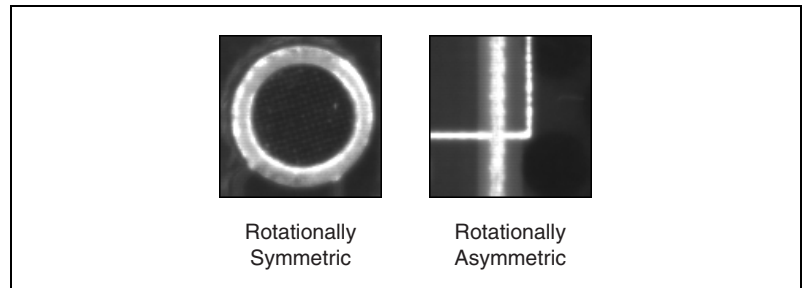
## Defining and Creating Good Template Images

The selection of a good template image plays a critical part in obtaining good results. Because the template image represents the pattern that you want to find, make sure that all the important and unique characteristics of the pattern are well defined in the image.

Several factors are critical in creating a template image. These critical factors include symmetry, feature detail, positional information, and background information.

### Symmetry

A rotationally symmetric template is less sensitive to changes in rotation than one that is rotationally asymmetric. A rotationally symmetric template provides good positioning information but no orientation information.



**Figure 5-7.** Symmetry

### Feature Detail

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

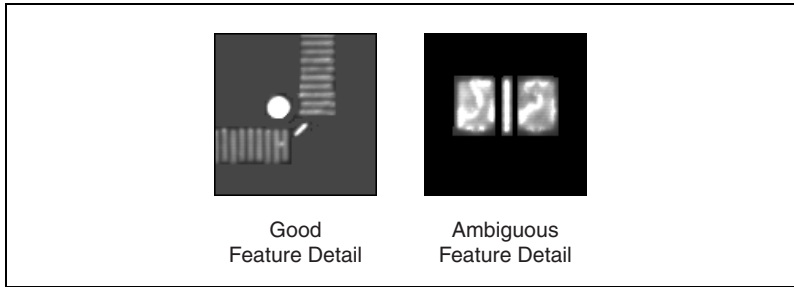


Figure 5-8. Feature Detail

### Positional Information

A template with strong edges in both the x and y directions is easier to locate.

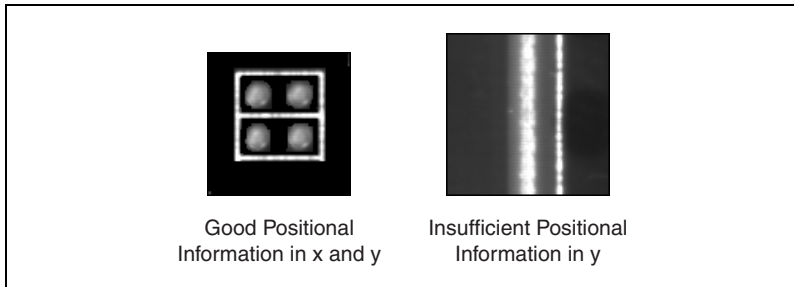


Figure 5-9. Positional Information

### Background Information

Unique background information in a template improves search performance and accuracy.

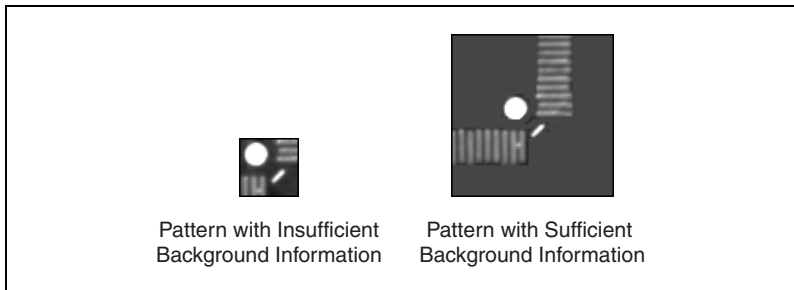


Figure 5-10. Background Information

## Training the Pattern Matching Algorithm

After you create a good template image, the pattern matching algorithm has to learn the important features of the template. Use `CWIMAQVision.LearnPattern` to learn the template. The learning process depends on the type of matching that you expect to perform. If you do not expect the instance of the template in the image to rotate or change its size, then the pattern matching algorithm has to learn only those features from the template that are necessary for shift-invariant matching. However, if you want to match the template at any orientation, the learning process must consider the possibility of arbitrary orientations. To specify which type of learning mode to use, pass the learn mode to the `LearnPatternOptions` parameter of `CWIMAQVision.LearnPattern`. You can also set the `LearnMode` property of a `CWIMAQLearnPatternOptions` object and pass this object for the `LearnPatternOptions` parameter of `CWIMAQVision.LearnPattern`.

The learning process is usually time intensive because the algorithm attempts to find the optimum features of the template for the particular matching process. You can train the pattern matching algorithm offline and then save the template image by using `CWIMAQVision.WriteImageAndVisionInfo`.

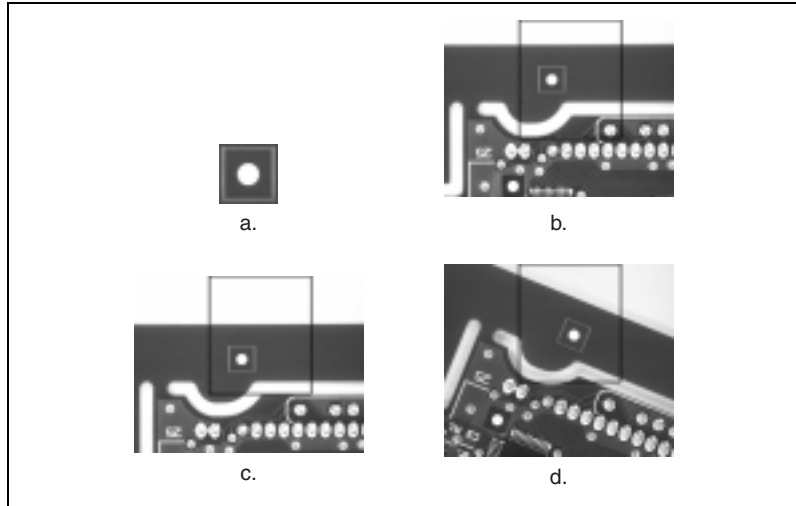
## Defining a Search Area

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

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

In many inspection applications, you have general information about the location of the fiducial. You should use this information to define a search area. For example, in a typical component placement application, each printed circuit board (PCB) being tested may not be placed in the same location with the same orientation. The location of the PCB in various images can move and rotate within a known range of values, as illustrated in Figure 5-11. Figure 5-11a shows the template used to locate the PCB in the image. Figure 5-11b shows an image containing a PCB with a fiducial

you want to locate. Notice the search area around the fiducial. If you know before the matching process begins that the PCB can shift or rotate in the image within a fixed range (as shown in Figure 5-11c and Figure 5-11d, respectively), then you can limit the search for the fiducial to a small region of the image.



**Figure 5-11.** Selecting a Search Area for Grayscale Pattern Matching

## Setting Matching Parameters and Tolerances

Every pattern matching algorithm makes assumptions about the images and pattern matching parameters used in machine vision applications. These assumptions work for a high percentage of the applications. However, there may be applications in which the assumptions used in the algorithm are not optimal. Knowing your particular application and the images you want to process is useful in selecting the pattern matching parameters. The following are parameters that influence the IMAQ Vision pattern matching algorithm.

### Minimum Contrast

You can set the minimum contrast to potentially increase the pattern matching algorithm's speed. Contrast is the difference between the smallest and largest pixel values in a region. The pattern matching algorithm ignores all image regions where contrast values fall beneath a set minimum contrast value. If the search image has high contrast but contains some low contrast regions, you can set a high minimum contrast value. By using a high minimum contrast value, you exclude all areas in the image with low

contrast, significantly reducing the region in which the pattern matching algorithm must search. If the search image has low contrast throughout, set a low minimum contrast parameter to ensure that the pattern matching algorithm looks for the template in all regions of the image. Use `CWIMAQMatchPatternOptions.MinimumContrast` to set the minimum contrast.

## Rotation Angle Ranges

For matching objects that may rotate in the image, the pattern matching algorithm, by default, allows any orientation between  $0^\circ$  to  $360^\circ$ . If the pattern rotation in your application is restricted to a certain range (for example, from  $-15^\circ$  to  $15^\circ$ ), you can provide this information to the pattern matching algorithm by using the `CWIMAQMatchPatternOptions.RotationAngleRanges` property. This information improves your search time because the pattern matching algorithm looks for the pattern at fewer angles.

## Testing the Search Algorithm on Test Images

To determine if your selected template or reference pattern is appropriate for your machine vision application, test the template on a few test images. These test images should reflect the images generated by your machine vision application during true operating conditions. If the pattern matching algorithm locates the reference pattern in all cases, you have selected a good template. Otherwise, refine the current template, or select a better template until both training and testing are successful.

## Using a Ranking Method to Verify Results

The manner in which you interpret the pattern matching results depends on your application. For typical alignment applications, such as finding a fiducial on a wafer, the most important information is the position and bounding rectangle of the best match. Use `CWIMAQPatternMatchReportItem.Position` and `CWIMAQPatternMatchReportItem.BoundingPoints` to get the position and location of a match.

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

Use `CWIMAQPatternMatchReportItem.Score` to get the score corresponding to a match.

## Finding Points Using Color Pattern Matching

Color pattern matching algorithms provide a quick way to locate objects when color is present. Use color pattern matching if:

- The object you want to locate has color information that is very different from the background, and you want to find a very precise location of the object in the image.
- The object to locate has grayscale properties that are very difficult to characterize or that are very similar to other objects in the search image. In such cases, grayscale pattern matching can give inaccurate results. If the object has color information that differentiates it from the other objects in the scene, color provides the machine vision software with the additional information to locate the object.

Color pattern matching returns the location of the center of the template and the template orientation. Follow these general steps to find features in an image using color pattern matching:

1. Define a reference or fiducial pattern in the form of a template image.
2. Use the reference pattern to train the color pattern matching algorithm with `CWIMAQVision.LearnColorPattern`.
3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.
4. Set `CWIMAQMatchColorPatternOptions.FeatureMode` to `cwimaqFeatureAll`.
5. Set the tolerances and parameters to specify how the algorithm operates at run time using `CWIMAQMatchColorPatternOptions`.
6. Test the search algorithm on test images using `CWIMAQVision.MatchColorPattern`.
7. Verify the results using a ranking method.

## Defining and Creating Good Color Template Images

The selection of a good template image plays a critical part in obtaining accurate results with the color pattern matching algorithm. Because the template image represents the color and the pattern that you want to find, make sure that all the important and unique characteristics of the pattern are well defined in the image.



Several factors are critical in creating a template image. These critical factors include color information, symmetry, feature detail, positional information, and background information.

## Color Information

A template with colors that are unique to the pattern provides better results than a template that contains many colors, especially colors found in the background or other objects in the image.

## Symmetry

A rotationally symmetric template in the luminance plane is less sensitive to changes in rotation than one that is rotationally asymmetric.

## Feature Detail

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

## Positional Information

A template with strong edges in both the x and y directions is easier to locate.

## Background Information

Unique background information in a template improves search performance and accuracy during the grayscale pattern matching phase. This requirement could conflict with the “color information” requirement because background colors may not be desirable during the color location phase. Avoid this problem by choosing a template with sufficient background information for grayscale pattern matching while specifying the exclusion of the background color during the color location phase. Refer to the [Training the Pattern Matching Algorithm](#) section of this chapter for more information on how to ignore colors.

## Training the Color Pattern Matching Algorithm

After you have created a good template image, the color pattern matching algorithm learns the important features of the template. Use `CWIMAVision.LearnColorPattern` to learn the template. The learning process depends on the type of matching that you expect to perform. By default, the color pattern matching algorithm learns only those features from the template that are necessary for shift-invariant matching.

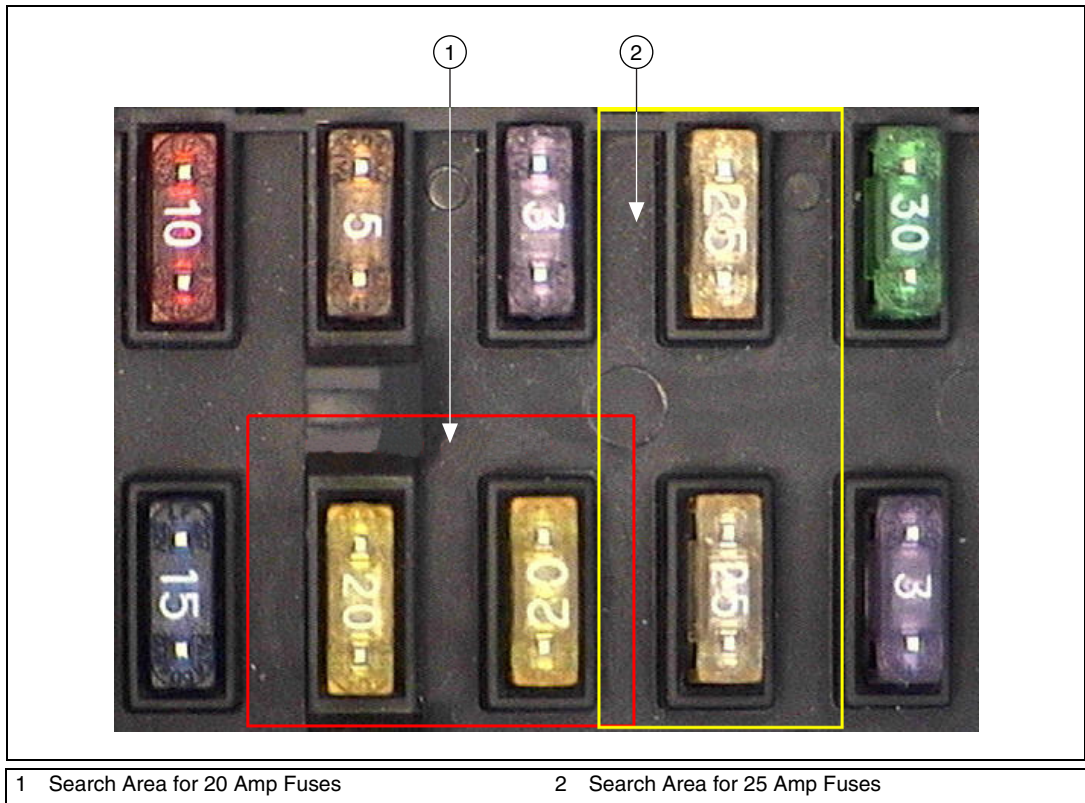
However, if you want to match the template at any orientation, the learning process must consider the possibility of arbitrary orientations. Use the `CWIMAQLearnColorPatternOptions.LearnMode` property to specify which type of learning mode to use.

Exclude colors in the template that you are not interested in using during the search phase. Typically, you should ignore colors that either belong to the background of the object or are not unique to the template, reducing the potential for incorrect matches during the color location phase. You can learn the colors to ignore using `CWIMAQVision.LearnColor`. Use the `CWIMAQLearnColorPatternOptions.IgnoreBlackAndWhite` or `CWIMAQLearnColorPatternOptions.IgnoreColorSpectra` properties to ignore background colors.

The training or learning process is time-intensive because the algorithm attempts to find optimal features of the template for the particular matching process. However, you can train the pattern matching algorithm offline, and save the template image using `CWIMAQVision.WriteImageAndVisionInfo`.

## Defining a Search Area

Two equally important factors define the success of a color pattern matching algorithm—accuracy and speed. You can define a search area to reduce ambiguity in the search process. For example, if your image has multiple instances of a pattern and only one instance is required for the inspection task, the presence of additional instances of the pattern can produce incorrect results. To avoid this, reduce the search area so that only the desired pattern lies within the search area. For example, in the fuse box inspection example use the location of the fuses to be inspected to define the search area. Because the inspected fuse box may not be in the exact location or have the same orientation in the image as the previous one, the search area you define should be large enough to accommodate these variations in the position of the box. Figure 5-12 shows how search areas can be selected for different objects.



**Figure 5-12.** Selecting a Search Area for Color Pattern Matching

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

## Setting Matching Parameters and Tolerances

Every color pattern matching algorithm makes assumptions about the images and color pattern matching parameters used in machine vision applications. These assumptions work for a high percentage of the applications.

In some applications, the assumptions used in the algorithm are not optimal. In such cases, you must modify the color pattern matching parameters. Knowing your particular application and the images you want to process is useful in selecting the pattern matching parameters.

The following are some parameters in the IMAQ Vision color pattern matching algorithm and how they influence color pattern matching: color sensitivity, search strategy, color score weight, minimum contrast, and rotation angle ranges.

## Color Sensitivity

Use the color sensitivity to control the granularity of the color information in the template image. If the background and objects in the image contain colors that are very close to colors in the template image, use a higher color sensitivity setting. A higher sensitivity setting distinguishes colors with very close hue values. Three color sensitivity settings are available in IMAQ Vision: low, medium, and high. Use the default low setting if the colors in the template are very different from the colors in the background or other objects that you are not interested in. Increase the color sensitivity settings as the color differences decrease. Use

`CWIMAQMatchColorPatternOptions.ColorSensitivity` to set the color sensitivity. For more information on color sensitivity, see Chapter 14, *Color Inspection*, of the *IMAQ Vision Concepts Manual*.

## Search Strategy

Use the search strategy to optimize the speed of the color pattern matching algorithm. The search strategy controls the step size, sub-sampling factor, percentage of color information used from the template.

Choose from four different strategies:

- **Very aggressive**—Uses the largest step size, the most sub-sampling and only the dominant color from the template to search for the template. Use this strategy when the color in the template is almost uniform, the template is well contrasted from the background and there is a good amount of separation between different occurrences of the template in the image. This strategy is the fastest way to find templates in an image.
- **Aggressive**—Uses a large step size, a lot of sub-sampling, and little of the color information from the template.
- **Balanced**—Uses values in between the aggressive and conservative strategies.
- **Conservative**—Uses a very small step size, a sub-sampling factor of two, and all the color information present in the template. The conservative strategy is the most reliable method to look for a template in any image at potentially reduced speed.



**Note** Use the conservative strategy if you have multiple targets located very close to each other in the image.

Decide on the best strategy by experimenting with the different options. Use `CWIMAQMatchColorPatternOptions.SearchStrategy` to select a search strategy.

## Color Score Weight

When you search for a template using both color and shape information, the color and shape scores generated during the match process are combined to generate the final color pattern matching score. The color score weight determines the contribution of the color score to the final color pattern matching score. If the template's color information is superior to its shape information, set the weight higher. For example, if you use a weight of 1000, the algorithm finds each match by using both color and shape information and then ranks the matches based entirely on their color scores. If the weight is 0, the matches are ranked based entirely on only their shape scores. Use the `CWIMAQMatchColorPatternOptions.ColorScoreWeight` property to set the color score weight.

## Minimum Contrast

Use the minimum contrast to increase the speed of the color pattern matching algorithm. The color pattern matching algorithm ignores all image regions where grayscale contrast values fall beneath a set minimum contrast value. Use `CWIMAQMatchColorPatternMatchingOptions.MinimumContrast` to set the minimum contrast. See the [Setting Matching Parameters and Tolerances](#) section of this chapter for more information about minimum contrast.

## Rotation Angle Ranges

If you know that the pattern rotation is restricted to a certain range (for example, between  $-15^\circ$  to  $15^\circ$ ), provide this restriction information to the pattern matching algorithm by using the `CWIMAQMatchPatternOptions.RotationAngleRanges` property. This information improves your search time because the color pattern matching algorithm looks for the pattern at fewer angles. See Chapter 12, *Pattern Matching*, in the *IMAQ Vision Concepts Manual* for more information on pattern matching.

## Testing the Search Algorithm on Test Images

To determine if your selected template or reference pattern is appropriate for your machine vision application, test the template on a few test images by using the `CWIMAVision.MatchColorPattern` method. These test images should reflect the images generated by your machine vision application during true operating conditions. If the color pattern matching algorithm locates the reference pattern in all cases, you have selected a good template. Otherwise, refine the current template, or select a better template until both training and testing are successful.

## Finding Points Using Color Location

Color location algorithms provide a quick way to locate regions in an image with specific colors.

Use color location when your application:

- Requires the location and the number of regions in an image with their specific color information
- Relies on the cumulative color information in the region, instead of the color arrangement in the region
- Does not require the orientation of the region
- Does not always require the location with sub-pixel accuracy
- Does not require shape information for the region

Follow these general steps to find features in an image using color location:

1. Define a reference pattern in the form of a template image.
2. Use the reference pattern to train the color location algorithm with `CWIMAVision.LearnColorPattern`.
3. Define an image or an area of an image as the search area. A small search area reduces the time to find the features.
4. Set `CWIMAVision.MatchColorPatternOptions.FeatureMode` to `cwimagFeatureColorInformation`.
5. Set the tolerances and parameters to specify how the method operates at run time using `CWIMAVision.MatchColorPatternOptions`.
6. Test the color location algorithm on test images using `CWIMAVision.MatchColorPattern`.
7. Verify the results using a ranking method.

You can save the template image using `CWIMAVision.WriteImageAndVisionInfo`.

# Convert Pixel Coordinates to Real-World Coordinates

---

The measurement points you located with edge detection and pattern matching are in pixel coordinates. If you need to make measurements using real-world units, use

`CWIMAQVision.ConvertPixelToRealWorldCoordinates` to convert the pixel coordinates into real-world units.

## Make Measurements

---

You can make different types of measurements either directly from the image or from points that you detect in the image.

### Distance Measurements

Use the following methods to make distance measurements for your inspection application.

Clamp methods measure the separation between two edges in a rectangular search region. First, clamp methods detect points along the two edges using the rake method. Then, they compute the distance between the points detected on the edges along each search line of the rake and return the largest or smallest distance in either the horizontal or vertical direction. The **Axis** parameter specifies the axis along which to measure. You also need to specify the parameters for edge detection and the separation between the search lines that you want to use within the search region to find the edges. These methods work directly on the image under inspection, and they output the coordinates of all the edge points that they find. The following list describes the available clamp methods:

- `CWMachineVision.MeasureMaximumDistance`—Measures the largest separation between two edges in a rectangular search region.
- `CWMachineVision.MeasureMinimumDistance`—Measures the smallest separation between two edges in a rectangular search region.

Use `CWIMAQVision.FindPointDistances` to compute the distances between consecutive pairs of points in an array of points. You can obtain these points from the image using any one of the feature detection methods described in the [Find Measurement Points](#) section of this chapter.

## Analytic Geometry Measurements

Use the following CWIMAVision methods to make geometrical measurements from the points you detect in the image:

- `FitLine`—Fits a line to a set of points and computes the equation of the line.
- `FitCircle`—Fits a circle to a set of at least three points and computes its area, perimeter and radius.
- `FitEllipse`—Fits an ellipse to a set of at least six points and computes its area, perimeter, and the lengths of its major and minor axis.
- `FindIntersectionPoint`—Finds the intersection point of two lines specified by their start and end points.
- `FindAngleBetweenLines`—Finds the smaller angle between two lines.
- `FindPerpendicularLine`—Finds the perpendicular line from a point to a line.
- `FindDistanceFromPointToLine`—Computes the perpendicular distance between the point and the line.
- `FindBisectingLine`—Finds the line that bisects the angle formed by two lines.
- `FindMidLine`—Finds the line that is midway between a point and a line and is parallel to the line.
- `FindPolygonArea`—Calculates the area of a polygon specified by its vertex points.

## Instrument Reader Measurements

You can make measurements based on the values obtained by meter, LCD, and barcode readers.

Use `CWIMAVision.MeterArc.CreateFromPoints` or `CWIMAVision.MeterArc.CreateFromLines` to calibrate a meter or gauge that you want to read. `CWIMAVision.MeterArc.CreateFromLines` calibrates the meter using the initial position and the full-scale position of the needle. `CWIMAVision.MeterArc.CreateFromPoints` calibrates the meter using three points on the meter: the base of the needle, the tip of the needle at its initial position, and the tip of the needle at its full-scale position. Use `CWIMAVision.ReadMeter` to read the position of the needle using the `CWIMAVision.MeterArc` object.



Use `CWIMAQVision.FindLCDSegments` to calculate the regions of interest around each digit in an LCD or LED. To find the area of each digit, all the segments of the indicator must be activated. Use `CWIMAQVision.ReadLCD` to read the digits of an LCD or LED.

Use `CWIMAQVision.ReadBarcode` to read values encoded in 1D barcodes. First, specify a region of interest that encloses the barcode information, and specify the type of barcode. Then, read the barcode.

## Display Results

---

You can display the results obtained at various stages of your inspection process on the window that displays your inspection image. You can do this by overlaying information on an image. The software attaches the information that you want to overlay to the image, but it does not modify the image. Access overlays using the `CWIMAQImage.Overlays` property. The `CWIMAQOverlays` collection contains a single `CWIMAQOverlay` object that you can access using `CWIMAQImage.Overlay(1)`.



**Note** The `CWIMAQImage.Overlays` collection does not support usual collection methods—such as `Add`, `Remove`, and `RemoveAll`—because they are reserved for future use.

Use the following methods on the `CWIMAQOverlay` object to overlay search regions, inspection results, and other information, such as text and pictures. Overlays onto a viewer image are automatically updated when you call one of these methods.

- `DrawLine`—Overlays a `CWIMAQLine` object on an image.
- `DrawConnectedPoints`—Overlays a `CWIMAQPoints` collection and draws a line between sequential points.
- `DrawRectangle`—Overlays a `CWIMAQRectangle` object on an image.
- `DrawOval`—Overlays a `CWIMAQOval` object on an image.
- `DrawArc`—Overlays a `CWIMAQArc` object on an image.
- `DrawPicture`—Overlays a picture object onto the image.
- `DrawText`—Overlays text on an image.
- `DrawRegions`—Overlays an ROI described by the `CWIMAQRegions` object on an image.

You can select the color of overlays by using the above methods. If you do not supply a color to an overlay method, the `CWIMAQOverlay.DefaultColor` property is used.

You can configure the following `CWMachineVision` methods to overlay different types of information on the inspection image:

- `FindStraightEdge`
- `FindCircularEdge`
- `FindConcentricEdge`
- `MeasureMaximumDistance`
- `MeasureMinimumDistance`
- `FindPattern`
- `CountAndMeasureObjects`
- `FindCoordTransformUsingRect`
- `FindCoordTransformUsingTwoRects`
- `FindCoordTransformUsingPattern`

You can overlay the following information with all the above methods except `CWMachineVision.FindPattern`:

- The search area input into the method
- The search lines used for edge detection
- The edges detected along the search lines
- The result of the method

Each of the above `CWMachineVision` methods has a settings object input that allows you to select the information you want to overlay. Set the boolean property that corresponds to the information you want to overlay to `True`. With `CWMachineVision.FindPattern`, you can overlay the search area and the result.

Use `CWIMAQOverlay.Clear` to clear any previous overlay information from the image. Use `CWIMAQVision.WriteImageAndVisionInfo` to save an image with its overlay information to a file. You can read the information from the file into an image using the `CWIMAQVision.ReadImageAndVisionInfo`. As with calibration information, overlay information is removed from an image when the image size or orientation changes.

---

# Calibration

This chapter describes how to calibrate your imaging system, save calibration information, and attach calibration information to an image.

After you set up your imaging system, you may want to calibrate your system. If your imaging setup is such that the camera axis is perpendicular or nearly perpendicular to the object under inspection and your lens has no distortion, use simple calibration. With simple calibration, you do not need to learn a template. Instead, you define the distance between pixels in the horizontal and vertical directions using real-world units.

If your camera axis is not perpendicular to the object under inspection or your lens is distorted, use perspective and nonlinear distortion calibration to calibrate your system.

---

## Perspective and Nonlinear Distortion Calibration

Perspective errors and lens aberrations cause images to appear distorted. This distortion misplaces information in an image, but it does not necessarily destroy the information in the image. Calibrate your imaging system if you need to compensate for perspective errors or nonlinear lens distortion.

Follow these general steps to calibrate your imaging system:

1. Define a calibration template.
2. Define a reference coordinate system.
3. Learn the calibration information.

After you calibrate your imaging setup, you can attach the calibration information to an image. See the [Attach Calibration Information](#) section of this chapter for more information. Depending on your needs, you can either apply the calibration information to convert pixel coordinates to real-world coordinates without correcting the image, or you can create a distortion-free image by correcting the image for perspective errors and lens aberrations. See Chapter 4, [Blob Analysis](#), and Chapter 5, [Machine Vision](#), for more information about applying calibration information before making measurements.

## Defining a Calibration Template

You can define a calibration template by supplying an image of a grid or providing a list of pixel coordinates and their corresponding real-world coordinates. This section discusses the grid method in detail.

A calibration template is a user-defined grid of circular dots. As shown in Figure 6-1, the grid has constant spacings in the  $x$  and  $y$  directions. You can use any grid, but follow these guidelines for best results:

- The displacement in the  $x$  and  $y$  directions should be equal ( $dx = dy$ ).
- The dots should cover the entire desired working area.
- The radius of the dots should be 6–10 pixels.
- The center-to-center distance between dots should range from 18 to 32 pixels, as shown in Figure 6-1.
- The minimum distance between the edges of the dots should be 6 pixels, as shown in Figure 6-1.

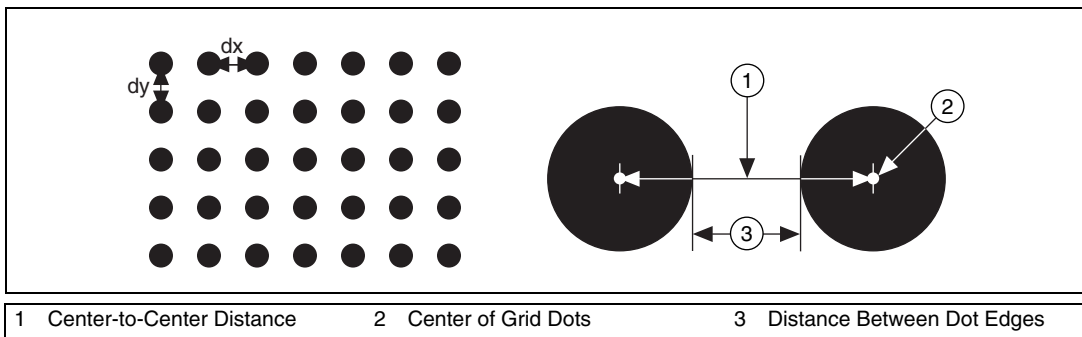


Figure 6-1. Defining a Calibration Grid

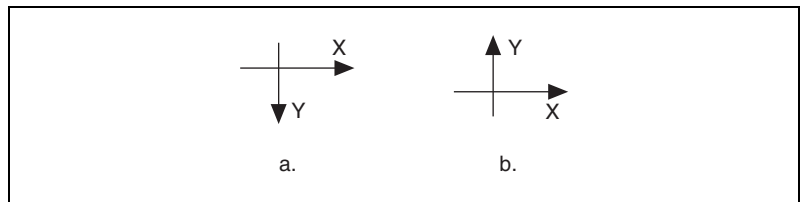


**Note** You can use the calibration grid installed with IMAQ Vision at **Start»Programs»National Instruments»Vision»Documentation»Calibration Grid**. The dots have radii of 2 mm and center-to-center distances of 1 cm. Depending on your printer, these measurements may change by a fraction of a millimeter. You can purchase highly accurate calibration grids from optics suppliers, such as Edmund Industrial Optics.

## Defining a Reference Coordinate System

To express measurements in real-world units, you need to define a coordinate system in the image of the grid. Use `CWIMAQLearnCalibrationOptions.CalibrationAxisInfo` to define a coordinate system by its origin, angle, and axis direction.

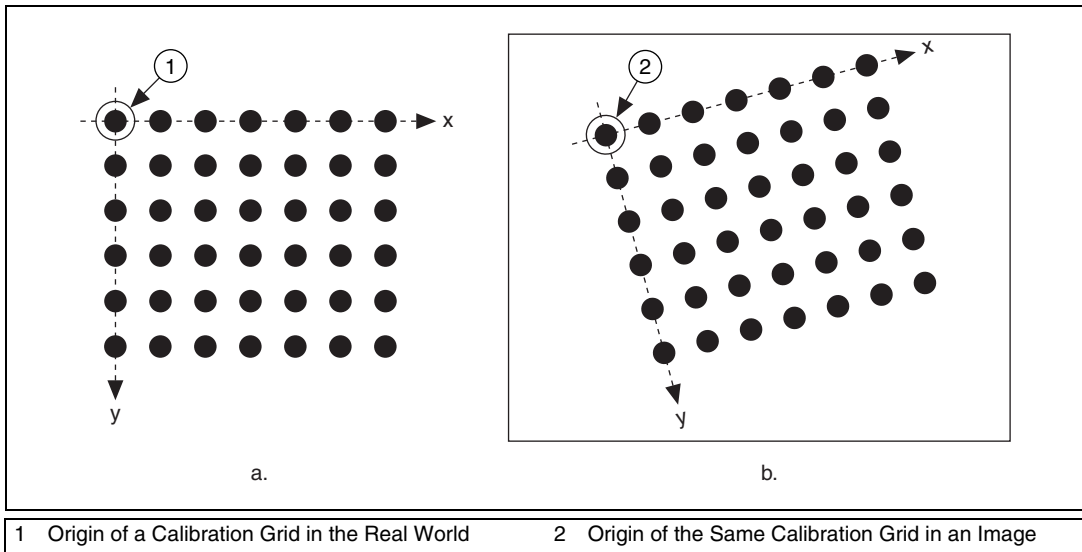
The origin, expressed in pixels, defines the center of your coordinate system. The angle specifies the orientation of your coordinate system with respect to the angle of the topmost row of dots in the grid image. The calibration procedure automatically determines the direction of the horizontal axis in the real world. The vertical axis direction can either be indirect, as shown in Figure 6-2a, or direct, as shown in Figure 6-2b.



**Figure 6-2.** Axis Direction in the Image Plane

If you do not specify a coordinate system, the calibration process defines a default coordinate system. If you specify a grid for the calibration process, the software defines the following default coordinate system, as shown in Figure 6-3:

1. The origin is placed at the center of the left, topmost dot in the calibration grid.
2. The angle is set to  $0^\circ$ . This aligns the x-axis with the first row of dots in the grid, as shown in Figure 6-3b.
3. The axis direction is set to *indirect*. This aligns the y-axis to the first column of the dots in the grid, as shown in Figure 6-3b.



1 Origin of a Calibration Grid in the Real World

2 Origin of the Same Calibration Grid in an Image

**Figure 6-3.** A Calibration Grid and an Image of the Grid



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

1. The origin is placed at the point in the list with the lowest x-coordinate value and then the lowest y-coordinate value.
2. The angle is set to  $0^\circ$ .
3. The axis direction is set to *indirect*.

If you define a coordinate system yourself, carefully consider the needs of your application. Remember the following:

- Express the origin in pixels. Always choose an origin location that lies within the calibration grid so that you can convert the location to real-world units.
- Specify the angle as the angle between the x-axis of the new coordinate system ( $x'$ ) and the top row of dots ( $x$ ), as shown in Figure 6-4. If your imaging system exhibits nonlinear distortion, you cannot visualize the angle as you can in Figure 6-4 because the dots do not appear in straight lines.

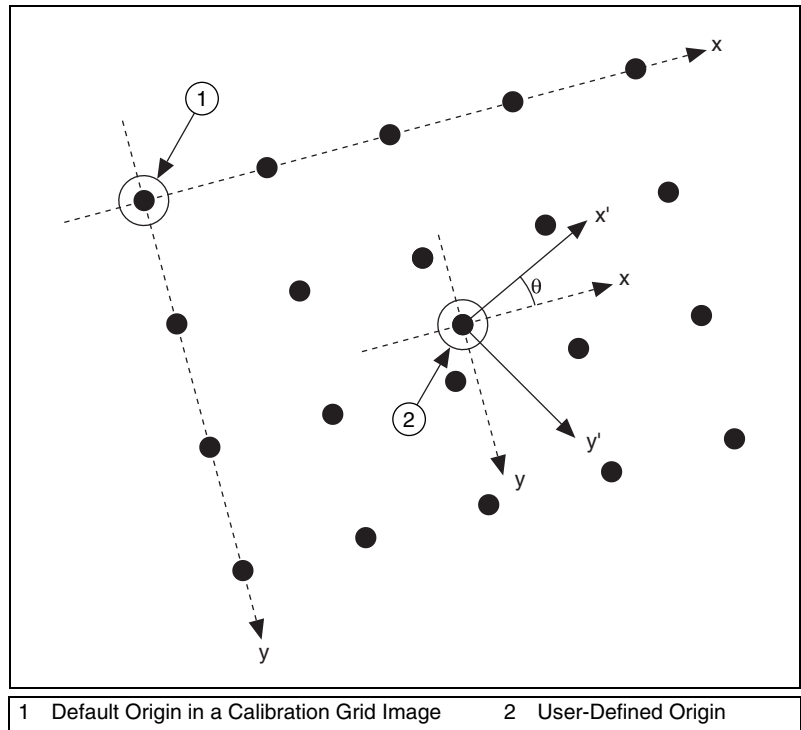


Figure 6-4. Defining a Coordinate System

## Learning Calibration Information

After you define a calibration grid and reference axis, acquire an image of the grid using the current imaging setup. For information about acquiring images, see the [Acquire or Read an Image](#) section of Chapter 2, [Getting Measurement-Ready Images](#). The grid does not need to occupy the entire image. You can choose a region within the image that contains the grid. After you acquire an image of the grid, learn the calibration information by inputting the image of the grid into `CWIMAQVision.LearnCalibrationGrid`.



**Note** If you want to specify a list of points instead of a grid, use `CWIMAQVision.LearnCalibrationPoints` to learn the calibration information. Use the `CWIMAQCalibrationPoints` object to specify the pixel to real-world mapping.

## Specifying Scaling Factors

Scaling factors are the real-world distances between the dots in the calibration grid in the x and y directions and the units in which the distances are measured. Use

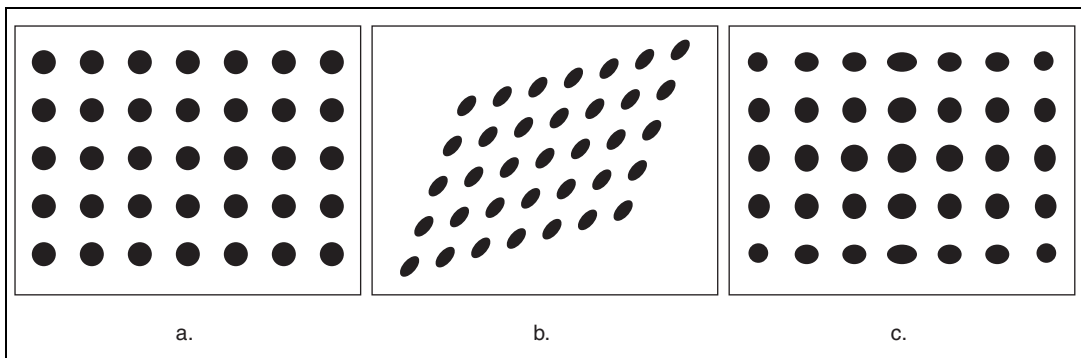
`CWIMAQCalibrationGridOptions.GridDescriptor` to specify the scaling factors.

## Choosing a Region of Interest

Define a learning region of interest (ROI) during the learning process to define a region of the calibration grid you want to learn. The software ignores dot centers outside this region when it estimates the transformation. Creating a user-defined ROI is an effective way to increase correction speeds depending on the other calibration options selected. Pass a `CWIMAQRegions` collection representing the region you want to learn to the `Regions` parameter of `CWIMAQVision.LearnCalibrationGrid` or `CWIMAQVision.LearnCalibrationPoints`.

## Choosing a Learning Algorithm

Select a method in which to learn the calibration information: perspective projection or nonlinear. Figure 6-5 illustrates the types of errors your image can exhibit. Figure 6-5a shows an image of a calibration grid with no errors. Figure 6-5b shows an image of a calibration grid with perspective projection. Figure 6-5c shows an image of a calibration grid with nonlinear distortion.



**Figure 6-5.** Types of Image Distortion

Choose the perspective projection algorithm when your system exhibits perspective errors only. A perspective projection calibration has an accurate transformation even in areas not covered by the calibration grid, as shown

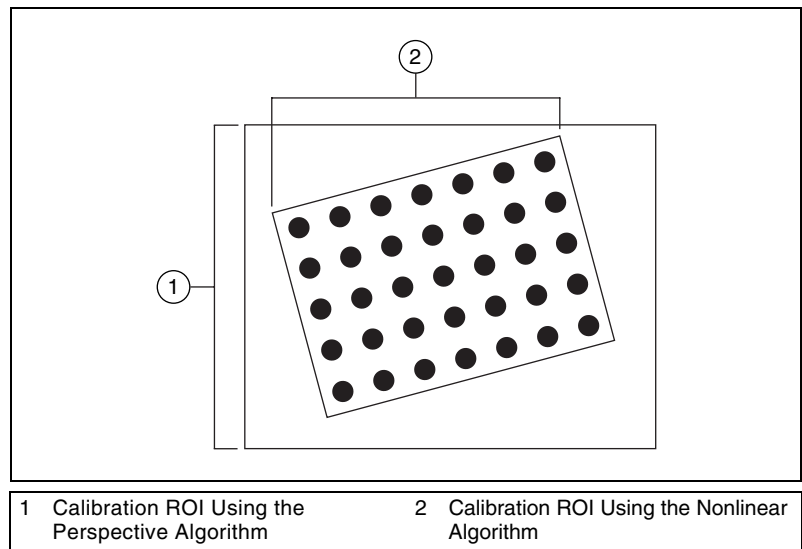


in Figure 6-6. Set

`CWIMAQLearnCalibrationOptions.CalibrationMethod` to `cwimaqPerspectiveCalibration` to choose the perspective calibration algorithm. Learning and applying perspective projection is less computationally intensive than the nonlinear method. However, perspective projection cannot handle nonlinear distortions.

If your imaging setup exhibits nonlinear distortion, use the nonlinear method. The nonlinear method guarantees accurate results only in the area that the calibration grid covers, as shown in Figure 6-6. If your system exhibits both perspective and nonlinear distortion, use the nonlinear method to correct for both. Set

`CWIMAQLearnCalibrationOptions.CalibrationMethod` to `cwimaqNonLinearCalibration` to choose the nonlinear calibration algorithm.



**Figure 6-6.** Calibration ROIs

## Using the Learning Score

The learning process returns a score that reflects how well the software learned the input image. A high learning score indicates that you chose the the appropriate learning algorithm, that the grid image complies with the guideline, and that your vision system setup is adequate.

If the learning process returns a low score, try the following:

1. Make sure your grid complies with the guidelines listed in the [Defining a Calibration Template](#) section.
2. Check the lighting conditions. If you have too much or too little lighting, the software may estimate the center of the dots incorrectly. Also, adjust your threshold range to distinguish the dots from the background.
3. Select another learning algorithm. When nonlinear lens distortion is present, using perspective projection sometimes results in a low learning score.



**Note** A high score does *not* reflect the accuracy of your system.

## Learning the Error Map

An error map helps you gauge the quality of your complete system. The error map returns an estimated error range to expect when a pixel coordinate is transformed into a real-world coordinate. The transformation accuracy may be higher than the value the error range indicates. Set `CWIMAQLearnCalibrationOptions.LearnErrorMap` to `True` to learn the error map.

## Learning the Correction Table

If the speed of image correction is a critical factor for your application, use a correction table. The correction table is a lookup table stored in memory that contains the real-world location information of all the pixels in the image. The extra memory requirements for this option are based on the size of the image. Use this option when you want to correct several images at a time in your vision application. Set `CWIMAQLearnCalibrationOptions.LearnCorrectionTable` to `True` to learn the correction table.

## Setting the Scaling Mode

Use the scaling mode option to choose the appearance of the corrected image. Set `CWIMAQLearnCalibrationOptions.CorrectionScalingMode` to `cwimaqScaleToFit` or `cwimaqScaleToPreserveArea`. For more information about the scaling mode, see Chapter 3, *System Setup and Calibration*, in the *IMAQ Vision Concepts Manual*.

## Calibration Invalidation

Any image processing operation that changes the image size or orientation voids the calibration information in a calibrated image. Examples of methods that void calibration information include

`CWIMAQVision.Resample2`, `CWIMAQVision.Extract2`, `CWIMAQVision.Unwrap`, and `CWIMAQImage.ArrayToImage`.

## Simple Calibration

---

When the axis of your camera is perpendicular to the image plane and lens distortion is negligible, use simple calibration. In simple calibration, a pixel coordinate is transformed to a real-world coordinate through scaling in the horizontal and vertical directions.

Use simple calibration to map pixel coordinates to real-world coordinates directly without a calibration grid. The software rotates and scales a pixel coordinate according to predefined coordinate reference and scaling factors. You can assign the calibration to an arbitrary image using `CWIMAQVision.SetSimpleCalibration`.

To perform a simple calibration, set a coordinate system (angle, center, and axis direction) and scaling factors on the defined axis, as shown in Figure 6-7. Express the angle between the x-axis and the horizontal axis of the image in degrees. Express the center as the position, in pixels, where you want the coordinate system origin. Set the axis direction to direct or indirect. Simple calibration also offers a correction table option and a scaling mode option.

Use `CWIMAQSimpleCalibrationOptions.CalibrationAxisInfo` to define the coordinate reference. Use `CWIMAQSimpleCalibrationOptions.GridDescriptor` to specify the scaling factors. Use `CWIMAQSimpleCalibrationOptions.CorrectionScalingMode` to set the scaling mode. Set `CWIMAQSimpleCalibrationOptions.LearnCorrectionTable` to `True` to learn the correction table.

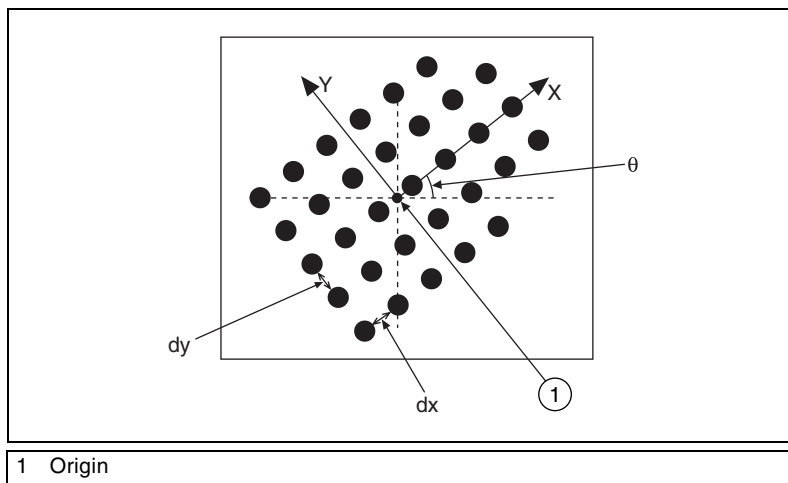


Figure 6-7. Defining a Simple Calibration

## Save Calibration Information

After you learn the calibration information, you can save it so that you do not have to relearn the information for subsequent processing. Use `CWIMAQVision.WriteImageAndVisionInfo` to save the image of the grid and its associated calibration information to a file. To read the file containing the calibration information use `CWIMAQVision.ReadImageAndVisionInfo`. For more information about attaching the calibration information you read from another image, see the *Attach Calibration Information* section.

## Attach Calibration Information

Now that you have calibrated your setup correctly, you can apply the calibration settings to images that you acquire. Use `CWIMAQVision.SetCalibrationInformation` to attach the calibration information of the current setup to each image you acquire. This method takes in a source image containing the calibration information and a destination image that you want to calibrate. The output image is your inspection image with the calibration information attached to it.

Using the calibration information attached to the image, you can accurately convert pixel coordinates to real-world coordinates to make any of the analytic geometry measurements with `CWIMAQVision.ConvertPixelToRealWorldCoordinates`. If your application requires you to make shape measurements, correct the image by removing distortion with `CWIMAQVision.CorrectCalibratedImage`.



**Note** Because calibration information is part of the image, it is propagated throughout the processing and analysis of the image. Methods that modify the image size (such as an image rotation method) void the calibration information. Use `CWIMAQVision.WriteImageAndVisionInfo` to save the image and all of the attached calibration information to a file.



---

# Technical Support Resources

## Web Support

---

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of [ni.com](http://ni.com)

## NI Developer Zone

---

The NI Developer Zone at [ni.com/zone](http://ni.com/zone) is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

## Customer Education

---

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of [ni.com](http://ni.com) for online course schedules, syllabi, training centers, and class registration.

## System Integration

---

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of [ni.com](http://ni.com)

## Worldwide Support

---

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of [ni.com](http://ni.com). Branch office Web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.

# Glossary

---

## Numbers

1D One-dimensional.

2D Two-dimensional.

## A

ActiveX control A standard software tool that adds additional functionality to any compatible ActiveX container. An ActiveX control has properties, methods, objects, and events.

AIPD National Instruments proprietary image file format used for saving complex images and calibration information pertaining to step and spatial units (extension APD).

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

alpha channel Channel used to code extra information, such as gamma correction, about a color image. The alpha channel is stored as the first byte in the four-byte representation of an RGB pixel.

annulus A region of interest that resembles a ring or partial ring.

area (1) A rectangular portion of an acquisition window or frame that is controlled and defined by software. (2) The size of an object in pixels or user-defined units.

area threshold Detects objects based on their size, which can fall within a user-specified range.

arithmetic operators The image operations multiply, divide, add, subtract, and remainder.

array Ordered, indexed set of data elements of the same type.

auto-median function A function that uses dual combinations of opening and closing operations to smooth the boundaries of objects.



## B

b	Bit. One binary digit, either 0 or 1.
B	Byte. Eight related bits of data, an eight-bit binary number. Also denotes the amount of memory required to store one byte of data.
barycenter	The grayscale value representing the centroid of the range of an image's grayscale values in the image histogram.
binary image	An image in which the objects usually have a pixel intensity of 1 (or 255) and the background has a pixel intensity of 0.
binary morphology	Functions that perform morphological operations on a binary image.
binary threshold	Separation of an image into objects of interest (assigned a non-zero pixel value) and background (assigned pixel values of 0) based on the intensities of the image pixels.
bit depth	The number of bits ( $n$ ) used to encode the value of a pixel. For a given $n$ , a pixel can take $2^n$ different values. For example, if $n$ equals 8-bits, a pixel can take 256 different values ranging from 0 to 255. If $n$ equals 16 bits, a pixel can take 65,536 different values ranging from 0 to 65,535 or $-32,768$ to 32,767.
black reference level	The level that represents the darkest an image can get. <i>See also</i> <a href="#">white reference level</a> .
blob	Binary large object. A connected region or grouping of pixels in an image in which all pixels have the same intensity level. Blobs are also referred to as objects or particles.
blob analysis	A series of processing operations and analysis functions that produce some information about the blobs in an image.
blurring	Reduces the amount of detail in an image. Blurring commonly occurs because the camera is out of focus. You can blur an image intentionally by applying a lowpass frequency filter.
BMP	Bitmap. Image file format commonly used for 8-bit and color images (extension BMP).
brightness	(1) A constant added to the red, green, and blue components of a color pixel during the color decoding process. (2) The perception by which white objects are distinguished from gray and light objects from dark objects.

buffer Temporary storage for acquired data.

## C

caliper (1) A function in IMAQ Vision Builder that calculates distances, angles, circular fits, and the center of mass based on positions given by edge detection, particle analysis, centroid, and search functions.  
(2) A measurement function that finds edge pairs along a specified path in the image. This function performs an edge extraction and then finds edge pairs based on specified criteria such as the distance between the leading and trailing edges, edge contrasts, and so forth.

center of mass The point on an object where all the mass of the object could be concentrated without changing the first moment of the object about any axis

centroid (1) The average of the x-coordinates and y-coordinates of a binary image or a blob in the image. The centroid of a blob may lie outside the blob.  
(2) The weighted average of the x-coordinates and y-coordinates in a grayscale image, where the weights are determined by the pixel values in the image.

character recognition The ability of a machine to read human-readable text.

chroma The color information in a video signal.

chromaticity The combination of hue and saturation. The relationship between chromaticity and brightness characterizes a color.

circle function Detects circular objects in a binary image.

closing A dilation followed by an erosion. A closing fills small holes in objects and smooths the boundaries of objects.

clustering A technique where the image is sorted into a discrete number of classes. Each class contains pixels that fall within a distinct range of grayscale values. The 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.

CLUT Color lookup table. Table for converting the value of a pixel in an image into a red, green, and blue (RGB) intensity.

color images Images containing color information, usually encoded in the RGB form.

color location	The technique that locates a color template in a color image based on only the color information.
color matching	The technique that compares the color information in an image or region of an image to the color information in another image or region of an image.
color pattern matching	The technique that locates a color template in a color image based on both the color information and grayscale pattern in the template.
color space	The mathematical representation for a color. For example, color can be described in terms of red, green, and blue; hue, saturation, and luminance; or hue, saturation, and intensity.
complex image	Stores information obtained from the FFT of an image. The complex numbers that compose the FFT plane are encoded in 64-bit floating-point values: 32bits for the real part and 32bits for the imaginary part.
connectivity	Defines which of the surrounding pixels of a given pixel constitute its neighborhood.
connectivity-4	Only pixels adjacent in the horizontal and vertical directions are considered neighbors.
connectivity-8	All adjacent pixels are considered neighbors.
contrast	(1) A constant multiplication factor applied to the luma and chroma components of a color pixel in the color decoding process. (2) The difference between light and dark intensity values in an image.
convex function	Computes the convex regions of objects in a binary image.
convex hull	The smallest convex polygon that can encapsulate a particle.
convolution	See <a href="#">linear filter</a> .
convolution kernel	2D matrices (or templates) used to represent the filter in the filtering process. The contents of these kernels are a discrete two-dimensional representation of the impulse response of the filter that they represent.
cross correlation	A technique that compares the similarity of two images or parts of an image. You can use cross correlation to find the optimal position where similarity exists.

**D**

Danielsson function	Similar to the distance functions, but with more accurate results.
dB	Decibel. The unit for expressing a logarithmic measure of the ratio of two signal levels: $dB = 20\log_{10} V_1/V_2$ , for signals in volts.
default setting	A default parameter value recorded in the driver. In many cases, the default input of a control is a certain value (often 0).
definition	The number of values a pixel can take on, which is the number of colors or shades that you can see in the image.
dendrite	Branches of the skeleton of an object.
densitometry	Intensity information about an image or regions of an image. Typical measurements include minimum, maximum, and mean intensity values as well as the standard deviation of the intensity values.
density function	For each gray level in a linear histogram, the function gives the number of pixels in the image that have the same gray level.
differentiation filter	Extracts the contours (edge detection) in gray level.
digital image	An image $f(x, y)$ that has been converted into a discrete number of pixels. Both spatial coordinates and brightness are specified.
dilation	Increases the size of an object along its boundary and removes tiny holes in the object.
distance calibration	Determination of the physical dimensions of a pixel by defining the physical dimensions of a line in the image.
distance function	Assigns to each pixel in an object a gray-level value equal to its shortest Euclidean distance from the border of the object.
driver	Software that controls a specific hardware device, such as an IMAQ or DAQ device.

## E

edge	Defined by a sharp change (transition) in the pixel intensities in an image or along an array of pixels.
edge contrast	The difference between the average pixel intensity before and the average pixel intensity after the edge.
edge detection	Any of several techniques to identify the edges of objects in an image.
edge hysteresis	The difference in threshold level between a rising and a falling edge.
edge steepness	The number of pixels that corresponds to the slope or transition area of an edge.
energy center	The center of mass of a grayscale image. <i>See also</i> <a href="#">center of mass</a> .
entropy	A measure of the randomness in an image. An image with high entropy contains more pixel value variation than an image with low entropy.
equalize function	<i>See</i> <a href="#">histogram equalization</a> .
erosion	Reduces the size of an object along its boundary and eliminates isolated points in the image.
Euclidean distance	The shortest distance between two points in a Cartesian system.
event	Object-generated response to some action or change in state, such as a mouse click. The event calls an event handler (callback function), which processes the event. Events are defined as part of an ActiveX control object.
exponential and gamma corrections	Expand the high gray-level information in an image while suppressing low gray-level information.
exponential function	Decreases brightness and increases contrast in bright regions of an image, and decreases contrast in dark regions of an image.

## F

feature	A specific pattern of pixels in an image.
FFT	Fast Fourier Transform. A method used to compute the Fourier transform of an image to get frequency information.

fiducial	A reference pattern on a part that helps a machine vision application find the part's location and orientation in an image.
form	Window or area on the screen on which you place controls and indicators to create the user interface for your program.
Fourier spectrum	The magnitude information of the Fourier transform of an image.
Fourier transform	Transforms an image from the spatial domain to the frequency domain.
frequency filters	Counterparts of spatial filters in the frequency domain. For images, frequency information is in the form of spatial frequency.

## G

gamma	The nonlinear change in the difference between the video signal's brightness level and the voltage level needed to produce that brightness.
gauging	Measurement of an object or distances between objects.
Gaussian filter	A filter similar to the smoothing filter, but using a Gaussian kernel in the filter operation. The blurring in a Gaussian filter is more gentle than a smoothing filter.
gradient convolution filter	<i>See</i> gradient filter.
gradient filter	Extracts the contours (edge detection) in gray-level values. Gradient filters include the Prewitt and Sobel filters.
gray level	The brightness of a pixel in an image.
gray-level dilation	Increases the brightness of pixels in an image that are surrounded by other pixels with a higher intensity.
gray-level erosion	Reduces the brightness of pixels in an image that are surrounded by other pixels with a lower intensity.
grayscale image	An image with monochrome information.
grayscale morphology	Functions that perform morphological operations on a gray-level image.

## H

highpass attenuation	Applies a linear attenuation to the frequencies in an image, with no attenuation at the highest frequency and full attenuation at the lowest frequency.
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	Removes all frequency information below a certain frequency.
histogram	Indicates the quantitative distribution of the pixels of an image per gray-level value.
histogram equalization	Transforms the gray-level values of the pixels of an image to occupy the entire range (0 to 255 in an 8-bit image) of the histogram, increasing the contrast of the image.
histogram inversion	Finds the inverse of an image. The histogram of a reversed image is equal to the original histogram flipped horizontally around the center of the histogram.
hit-miss function	Locates objects in the image similar to the pattern defined in the structuring element.
hole filling function	Fills all holes in objects that are present in a binary image.
HSI	Color encoding scheme in Hue, Saturation, and Intensity.
HSL	Color encoding scheme using Hue, Saturation, and Luma information where each image in the pixel is encoded using 32 bits: 8 bits for hue, 8 bits for saturation, 8 bits for luma, and 8 bits for the alpha channel.
HSV	Color encoding scheme in Hue, Saturation, and Value.
hue	Represents the dominant color of a pixel. The hue function is a continuous function that covers all the possible colors generated using the R, G, and B primaries. <i>See also</i> <a href="#">RGB</a> .

hue offset angle      The value added to all hue values so that the discontinuity occurs outside the values of interest during analysis.

## 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 border      A user-defined region of pixels surrounding an image. Functions that process pixels based on the value of the pixel neighbors require image borders.

Image Browser      An image that contains thumbnails of images to analyze or process in a vision application.

image buffer      Memory location used to store images.

image definition      *See* [pixel depth](#).

image enhancement      The process of improving the quality of an image that you acquire from a sensor in terms of signal-to-noise ratio, image contrast, edge definition, and so on.

image file      A file containing pixel data and additional information about the image.

image format      Defines how an image is stored in a file. Usually composed of a header followed by the pixel data.

image mask      A binary image that isolates parts of a source image for further processing. A pixel in the source image is processed if its corresponding mask pixel has a non-zero value. A source pixel whose corresponding mask pixel has a value of 0 is left unchanged.

image palette      The gradation of colors used to display an image on screen, usually defined by a color lookup table.

image processing      Encompasses various processes and analysis functions that you can apply to an image.

image source      Original input image.

image understanding      A technique that interprets the content of the image at a symbolic level rather than a pixel level.



image visualization	The presentation (display) of an image (image data) to the user.
imaging	Any process of acquiring and displaying images and analyzing image data.
IMAQ	Image Acquisition.
inner gradient	Finds the inner boundary of objects.
inspection	The process by which parts are tested for simple defects such as missing parts or cracks on part surfaces.
inspection function	Analyzes groups of pixels within an image and returns information about the size, shape, position, and pixel connectivity. Typical applications include quality of parts, analyzing defects, locating objects, and sorting objects.
instrument driver	A set of high-level software functions, such as NI-IMAQ, that control specific plug-in computer boards. Instrument drivers are available in several forms, ranging from a function callable from a programming language to a virtual instrument (VI) in LabVIEW.
intensity	(1) The sum of the Red, Green, and Blue primary colors divided by three, $(Red + Green + Blue)/3$ in a color image. (2) The gray-level value of a pixel in a grayscale image.
intensity calibration	Assigning user-defined quantities, such as optical densities or concentrations, to the gray-level values in an image.
intensity profile	The gray-level distribution of the pixels in an ROI of an image.
intensity range	Defines the range of gray-level values in an object of an image.
intensity threshold	Characterizes an object based on the range of gray-level values in the object. If the intensity range of the object falls within the user-specified range, it is considered an object. Otherwise it is considered part of the background.
interpolation	The technique used to find values in between known values when resampling an image or array of pixels.
IRE	A relative unit of measure (named for the Institute of Radio Engineers). 0 IRE corresponds to the blanking level of a video signal, 100 IRE to the white level. Note that for CCIR/PAL video, the black level is equal to the blanking level or 0 IRE, while for RS-170/NTSC video, the black level is at 7.5 IRE.

**J**

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

**K**

**kernel** Structure that represents a pixel and its relationship to its neighbors. The relationship is specified by weighted coefficients of each neighbor.

**L**

**labeling** The process by which each object in a binary image is assigned a unique value. This process is useful for identifying the number of objects in the image and giving each object a unique identity.

**LabVIEW** Laboratory Virtual Instrument Engineering Workbench. Program development environment application based on the programming language G used commonly for test and measurement applications.

**Laplacian filter** Extracts the contours of objects in the image by highlighting the variation of light intensity surrounding a pixel.

**line gauge** Measures the distance between selected edges with high-precision subpixel accuracy along a line in an image. For example, this function can be used to measure distances between points and edges. This function also can step and repeat its measurements across the image.

**line profile** Represents the gray-level distribution along a line of pixels in an image.

**linear filter** A special algorithm that calculates the value of a pixel based on its own pixel value as well as the pixel values of its neighbors. The sum of this calculation is divided by the sum of the elements in the matrix to obtain a new pixel value.

**logarithmic and inverse gamma corrections** Expand low gray-level information in an image while compressing information from the high gray-level ranges.

**logarithmic function** Increases the brightness and contrast in dark regions of an image and decreases the contrast in bright regions of the image.

logic operators	The image operations AND, NAND, OR, XOR, NOR, XNOR, difference, mask, mean, max, and min.
lossless compression	Compression in which the decompressed image is identical to the original image.
lossy compression	Compression in which the decompressed image is visually similar but not identical to the original image.
lowpass attenuation	Applies a linear attenuation to the frequencies in an image, with no attenuation at the lowest frequency and full attenuation at the highest frequency.
lowpass FFT filter	Removes or attenuates high frequencies present in the FFT domain of an image.
lowpass filter	Attenuates intensity variations in an image. You can use these filters to smooth an image by eliminating fine details and blurring edges.
lowpass frequency filter	Attenuates high frequencies present in the frequency domain of the image. A lowpass frequency filter suppresses information related to fast variations of light intensities in the spatial image.
lowpass truncation	Removes all frequency information above a certain frequency.
LSB	Least significant bit.
L-skeleton function	Uses an L-shaped structuring element in the skeleton function.
luma	The brightness information in the video picture. The luma signal amplitude varies in proportion to the brightness of the video signal and corresponds exactly to the monochrome picture.
luminance	<i>See</i> luma.
LUT	Lookup table. Table containing values used to transform the gray-level values of an image. For each gray-level value in the image, the corresponding new value is obtained from the lookup table.

## **M**

machine vision	An automated application that performs a set of visual inspection tasks.
mask FFT filter	Removes frequencies contained in a mask (range) specified by the user.

match score	A number ranging from 0 to 1000 that indicates how closely an acquired image matches the template image. A match score of 1000 indicates a perfect match. A match score of 0 indicates no match.
median filter	A lowpass filter that assigns to each pixel the median value of its neighbors. This filter effectively removes isolated pixels without blurring the contours of objects.
memory buffer	See <a href="#">buffer</a> .
method	In Visual Basic, a set of software instructions executed by a single line of code that may have input and/or output parameters and returns a value when executed.
morphological transformations	Extract and alter the structure of objects in an image. You can use these transformations for expanding (dilating) or reducing (eroding) objects, filling holes, closing inclusions, or smoothing borders. They are used primarily to delineate objects and prepare them for quantitative inspection analysis.
MSB	Most significant bit.
M-skeleton function	Uses an M-shaped structuring element in the skeleton function.
<b>N</b>	
neighbor	A pixel whose value affects the value of a nearby pixel when an image is processed. The neighbors of a pixel are usually defined by a kernel or a structuring element.
neighborhood operations	Operations on a point in an image that take into consideration the values of the pixels neighboring that point.
NI-IMAQ	Driver software for National Instruments IMAQ hardware.
nonlinear filter	Replaces each pixel value with a nonlinear function of its surrounding pixels.
nonlinear gradient filter	A highpass edge-extraction filter that favors vertical edges.
nonlinear Prewitt filter	A highpass, edge-extraction filter based on two-dimensional gradient information.

nonlinear Sobel filter	A highpass, edge-extraction filter based on two-dimensional gradient information. The filter has a smoothing effect that reduces noise enhancements caused by gradient operators.
Nth order filter	Filters an image using a nonlinear filter. This filter orders (or classifies) the pixel values surrounding the pixel being processed. The pixel being processed is set to the Nth pixel value, where N is the order of the filter.
number of planes (in an image)	The number of arrays of pixels that compose the image. A gray-level or pseudo-color image is composed of one plane, while an RGB image is composed of three planes (one for the red component, one for the blue, and one for the green).

## **O**

object	A connected region or grouping of pixels in an image in which all pixels have the same intensity level. Objects are also referred to as blobs or particles.
offset	The coordinate position in an image where you want to place the origin of another image. Setting an offset is useful when performing mask operations.
opening	An erosion followed by a dilation. An opening removes small objects and smooths boundaries of objects in the image.
operators	Allow masking, combination, and comparison of images. You can use arithmetic and logic operators in IMAQ Vision.
optical character verification	A machine vision application that inspects the quality of printed characters.
optical representation	Contains the low-frequency information at the center and the high-frequency information at the corners of an FFT-transformed image.
outer gradient	Finds the outer boundary of objects.
overlay	ROIs, text, and bitmaps that you can place on top of a displayed image to annotate it without modifying it.

## P

palette	The gradation of colors used to display an image on screen, usually defined by a color lookup table.
particle	A connected region or grouping of pixels in an image in which all pixels have the same intensity level. Particles are also referred to as blobs or objects.
pattern matching	The technique used to quickly locate a grayscale template within a grayscale image
picture aspect ratio	The ratio of the active pixel region to the active line region. For standard video signals like RS-170 or CCIR, the full-size picture aspect ratio normally is 4/3 (1.33).
picture element	An element of a digital image. Also called pixel.
pixel	Picture element. The smallest division that makes up the video scan line. For display on a computer monitor, a pixel's optimum dimension is square (aspect ratio of 1:1, or the width equal to the height).
pixel aspect ratio	The ratio between the physical horizontal size and the vertical size of the region covered by the pixel. An acquired pixel should optimally be square, thus the optimal value is 1.0, but typically it falls between 0.95 and 1.05, depending on camera quality.
pixel calibration	Directly calibrating the physical dimensions of a pixel in an image.
pixel depth	The number of bits used to represent the gray level of a pixel.
PNG	Portable Network Graphic. Image file format for storing 8-bit, 16-bit, and color images with lossless compression (extension PNG).
power 1/Y function	Similar to a logarithmic function but with a weaker effect.
power Y function	<i>See</i> <a href="#">exponential function</a> .
Prewitt filter	Extracts the contours (edge detection) in gray-level values using a $3 \times 3$ filter kernel. <i>See also</i> <a href="#">gradient filter</a> .
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.
pyramidal matching	A technique used to increase the speed of a pattern matching algorithm by matching subsampled versions of the image and the reference pattern.

## Q

quantitative analysis	Obtaining various measurements of objects in an image.
-----------------------	--

## R

real time	A property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time.
relative accuracy	A measure in LSB of the accuracy of an ADC; it includes all nonlinearity and quantization errors but does not include offset and gain errors of the circuitry feeding the ADC.
remove border objects function	Removes blobs in a binary image that touch the image border.
resolution	The number of rows and columns of pixels. An image composed of $m$ rows and $n$ columns has a resolution of $m \times n$ .
reverse function	Inverts the pixel values in an image.
RGB	Color encoding scheme using red, green, and blue (RGB) color information where each pixel in the color image is encoded using 32 bits: 8 bits for red, 8 bits for green, 8 bits for blue, and 8 bits for the alpha value (unused).
Roberts filter	Extracts the contours (edge detection) in gray level, favoring diagonal edges.

ROI	Region of interest. (1) An area of the image that is graphically selected from a window displaying the image. This area can be used to focus further processing. (2) A hardware-programmable rectangular portion of the acquisition window.
ROI tools	Collection of tools that enable you to select a region of interest from an image. These tools let you select points, lines, annuli, polygons, rectangles, rotated rectangles, ovals, and freehand open and closed contours.
rotational shift	The amount by which one image is rotated with respect to a reference image. This rotation is computed with respect to the center of the image.
rotation-invariant matching	A pattern matching technique in which the reference pattern can be located at any orientation in the test image as well as rotated at any degree.
<b>S</b>	
saturation	The amount of white added to a pure color. Saturation relates to the richness of a color. A saturation of zero corresponds to a pure color with no white added. Pink is a red with low saturation.
scale-invariant matching	A pattern matching technique in which the reference pattern can be any size in the test image. <i>See also</i> shape matching.
segmentation function	Fully partitions a labeled binary image into non-overlapping segments, with each segment containing a unique object.
separation function	Separates objects that touch each other by narrow isthmuses.
shape matching	Finds objects in an image whose shape matches the shape of the object specified by a shape template. The matching process is invariant to rotation and can be set to be invariant to the scale of the objects.
shift-invariant matching	A pattern matching technique in which the reference pattern can be located anywhere in the test image but cannot be rotated or scaled.
Sigma filter	A highpass filter that outlines edges.
skeleton function	Applies a succession of thinning operations to an object until its width becomes one pixel.
skiz function	Obtains lines in an image that separate each object from the others and are equidistant from the objects that they separate.



smoothing filter	Blurs an image by attenuating variations of light intensity in the neighborhood of a pixel.
Sobel filter	Extracts the contours (edge detection) in gray-level values using a $3 \times 3$ filter kernel. <i>See also</i> <a href="#">gradient filter</a> .
spatial calibration	Assigning physical dimensions to the area of a pixel in an image.
spatial filters	Alter the intensity of a pixel with respect to variations in intensities of its neighboring pixels. You can use these filters for edge detection, image enhancement, noise reduction, smoothing, and so forth.
spatial resolution	The number of pixels in an image, in terms of the number of rows and columns in the image.
square function	<i>See</i> <a href="#">exponential function</a> .
square root function	<i>See</i> <a href="#">logarithmic function</a> .
standard representation	Contains the low-frequency information at the corners and high-frequency information at the center of an FFT-transformed image.
structuring element	A binary mask used in most morphological operations. A structuring element is used to determine which neighboring pixels contribute in the operation.
sub-pixel analysis	Finds the location of the edge coordinates in terms of fractions of a pixel.

## T

template	Color, shape, or pattern that you are trying to match in an image using the color matching, shape matching, or pattern matching functions. A template can be a region selected from an image or it can be an entire image.
thickening	Alters the shape of objects by adding parts to the object that match the pattern specified in the structuring element.
thinning	Alters the shape of objects by eliminating parts of the object that match the pattern specified in the structuring element.
threshold	Separates objects from the background by assigning all pixels with intensities within a specified range to the object and the rest of the pixels to the background. In the resulting binary image, objects are represented with a pixel intensity of 255 and the background is set to 0.

threshold interval	Two parameters, the lower threshold gray-level value and the upper threshold gray-level value.
TIFF	Tagged Image File Format. Image format commonly used for encoding 8-bit, 16-bit, and color images (extension TIF).
Tools palette	Collection of tools that enable you to select regions of interest, zoom in and out, and change the image palette.

## V

value	The grayscale intensity of a color pixel computed as the average of the maximum and minimum red, green, and blue values of that pixel.
-------	--

## W

watershed	A technique used to segment an image into multiple regions.
web inspection	The process of detecting defects in a continuous sheet of materials at production speeds. Example materials include plastic film, cloth, paper and pulp products, metal, and glass.
white reference level	The level that defines what is white for a particular video system. <i>See also</i> <a href="#">black reference level</a> .

# Index

---

## A

acquiring measurement-ready images. *See* measurement-ready images, acquiring.  
Acquisition type combo box, 2-4  
ActiveX objects, 1-4  
analytic geometry measurements, 5-26  
analyzing images, 2-7 to 2-8  
Annulus tool (table), 3-2  
application development  
    general steps (figure), 1-5  
    inspection steps (figure), 1-6  
array, converting to image, 2-6  
attaching calibration information to images, 2-7, 6-10 to 6-11  
attenuation  
    highpass, 2-12  
    lowpass, 2-12

## B

binary images. *See* blob analysis.  
blob analysis, 4-1 to 4-7  
    converting pixel coordinates to real-world coordinates, 4-7  
    correcting image distortion, 4-2  
    creating binary image, 4-2 to 4-3  
    improving binary image, 4-3 to 4-4  
        improving blob shapes, 4-4  
        removing unwanted blobs, 4-3 to 4-4  
        separating touching blobs, 4-4  
    particle measurements, 4-4 to 4-7  
    steps (figure), 4-1  
Broken Line tool (table), 3-2

## C

calibration, 6-1 to 6-11  
    attaching calibration information to images, 2-7, 6-10 to 6-11  
    defining reference coordinate system, 6-3 to 6-5  
    defining template, 6-2  
    learning calibration information, 6-5 to 6-9  
        choosing learning algorithm, 6-6 to 6-7  
        choosing region of interest, 6-6  
        correction table, 6-8  
        error map, 6-8  
        invalidation of calibration, 6-9  
        learning score, 6-7 to 6-8  
        setting scaling mode, 6-8  
        specifying scaling factors, 6-6  
    overview, 2-2  
    for perspective and nonlinear distortion, 6-1 to 6-9  
    saving calibration information, 6-10  
    simple calibration, 6-9 to 6-10  
circles, finding points along edge, 5-10 to 5-11  
color location algorithms, 5-24  
color measurements, 3-1 to 3-13  
    color statistics, 3-7 to 3-13  
        comparing colors, 3-8 to 3-9  
        learning color information, 3-9 to 3-13  
        primary components of color image (figure), 3-8  
    defining regions of interest, 3-1 to 3-6  
        interactively, 3-1 to 3-5  
        programmatically, 3-5 to 3-6  
        using masks, 3-6  
color pattern matching, 5-18 to 5-24. *See also* pattern matching.  
    defining search area, 5-20 to 5-21  
    defining template images, 5-18 to 5-19

- ignoring background colors, 5-20
- setting matching parameters and tolerances, 5-21 to 5-23
  - color score weight, 5-23
  - color sensitivity, 5-22
  - minimum contrast, 5-23
  - rotation angle ranges, 5-23
  - search strategy, 5-22 to 5-23
- testing search algorithm on test images, 5-24
- training pattern matching algorithm, 5-19 to 5-20
- color statistics, 3-7 to 3-13
  - comparing colors, 3-8 to 3-9
  - learning color information, 3-9 to 3-13
    - choosing color representation sensitivity, 3-12
    - choosing right color information, 3-9 to 3-10
    - ignoring learned colors, 3-13
    - specifying information to learn, 3-10 to 3-12
    - using entire image, 3-10
    - using multiple regions in image, 3-11 to 3-12
    - using region in image, 3-10 to 3-11
  - primary components of color image (figure), 3-8
- complex image operations, 2-12
- contour, finding points along edge, 5-11 to 5-12
- conventions used in manual, *iv*
- convolution filters, 2-10
- coordinate reference
  - building for machine vision
    - choosing method (figure), 5-7
    - edge detection, 5-3 to 5-5
    - pattern matching, 5-6
  - defining for calibration, 6-3 to 6-5
- coordinates, converting pixel to real-world, 4-7, 5-25
- correction table, for calibration, 6-8
- creating applications. *See* application development.
- creating images. *See* images.
- customer education, A-1
- CWIMAQ control, 1-3
- CWIMAQ.AcquireImage method, 2-4
- CWIMAQ.ArrayToImage method, 2-12
- CWIMAQ.CalibrationGridOptions.GridDescriptor method, 6-6
- CWIMAQ.ColorInformation.ColorSpectrum method, 3-13
- CWIMAQImage object, 2-2. *See also* images.
- CWIMAQImage.ArrayToImage method, 2-6
- CWIMAQImage.Overlay object, 5-27
- CWIMAQImage.Overlays collection, 5-27
- CWIMAQImage.Overlays property, 5-27
- CWIMAQImage.ToArray method, 2-12
- CWIMAQKernel.Element method, 2-10
- CWIMAQKernel.LoadKernel method, 2-10
- CWIMAQKernel.SetSize method, 2-10
- CWIMAQLearnCalibrationOptions.
  - CalibrationAxisInfo method, 6-3
- CWIMAQLearnCalibrationOptions.
  - CalibrationMethod, 6-7
- CWIMAQLearnCalibrationOptions.
  - CorrectionScalingMode, 6-8
- CWIMAQLearnCalibrationOptions.LearnCorrectionTable, 6-8, 6-9
- CWIMAQLearnCalibrationOptions.LearnErrorMap, 6-8
- CWIMAQLearnColorPatternOptions.LearnMode property, 5-20
- CWIMAQMatchColorPatternOptions
  - method, 5-24
- CWIMAQMatchColorPatternOptions.ColorScoreWeight property, 5-23
- CWIMAQMatchColorPatternOptions.ColorSensitivity method, 5-22
- CWIMAQMatchColorPatternOptions.FeatureMode, 5-18, 5-24

- CWIMAQMatchColorPatternOptions.
  - MinimumContrast method, 5-23
- CWIMAQMatchColorPatternOptions.Search
  - Strategy method, 5-23
- CWIMAQMatchPatternOptions.Minimum
  - Contrast, 5-17
- CWIMAQMatchPatternOptions.Rotation
  - AngleRanges, 5-17, 5-23
- CWIMAQMeterArc.CreateFromLines
  - method, 5-26
- CWIMAQMeterArc.CreateFromPoints
  - method, 5-26
- cwimaq.ocx, 1-3
- CWIMAQOverlay.Clear method, 5-28
- CWIMAQOverlay.DefaultColor
  - property, 5-28
- CWIMAQPalette.Type property, 2-7
- CWIMAQPatternMatchReportItem.Bounding
  - Points, 5-17
- CWIMAQPatternMatchReportItem.Position,
  - 5-17
- CWIMAQPatternMatchReportItem.Score,
  - 5-18
- CWIMAQRegions collection, 3-5
- CWIMAQRegions.CopyTo method, 3-5
- CWIMAQRegion.Shape property, 3-5
- CWIMAQSimpleCalibrationOptions.
  - CalibrationAxisInfo method, 6-9
- CWIMAQSimpleCalibrationOptions.Grid
  - Descriptor method, 6-9
- CWIMAQ.Start method, 2-4, 2-5
- CWIMAQ.Stop method, 2-5
- CWIMAQViewer.Attach method, 2-6
- CWIMAQViewercontrol, 1-3, 2-6
- CWIMAQViewer.Image property, 2-6
- CWIMAQViewer.Palette property, 2-7
- CWIMAQViewer.Regions property, 3-5
- CWIMAQViewer.ShowToolInfo
  - property, 3-3
- CWIMAQVision control, 1-3
- CWIMAQVision.AutoThreshold method, 4-2
- CWIMAQVision.BasicParticle method, 4-4
- CWIMAQVision.CalculateCoefficients
  - method
- CWIMAQVision.CannyEdgeFilter
  - method, 2-10
- CWIMAQVision.Centroid2 method, 3-7
- CWIMAQVision.ColorThreshold
  - method, 4-2
- CWIMAQVision.ColorValueConversion
  - method, 3-8
- CWIMAQVision.ConcentricRake
  - method, 5-12
- CWIMAQVision.ConvertPixelToRealWorld
  - Coordinates method, 4-7, 5-25, 6-11
- CWIMAQVision.Convolute method, 2-9,
  - 2-10
- CWIMAQVision.CorrectCalibratedImage
  - method, 4-2, 6-11
- CWIMAQVision.CxAttenuate method, 2-11
- CWIMAQVision.CxTruncate method, 2-11
- CWIMAQVision.EdgeFilter method, 2-10
- CWIMAQVision.Equalize2 method, 2-9
- CWIMAQVision.ExtractColorPlanes
  - method, 3-7
- CWIMAQVision.ExtractComplexPlane
  - method, 2-12
- CWIMAQVision.ExtractSingleColorPlane
  - method, 3-7
- CWIMAQVision.FFT method, 2-11
- CWIMAQVision.FillHole method, 4-4
- CWIMAQVision.FindEdges2 method, 2-7,
  - 5-11, 5-12
- CWIMAQVision.FindLCDSegments
  - method, 5-27
- CWIMAQVision.FindPointDistances
  - method, 5-25
- CWIMAQVision.GetFileInformation
  - method, 2-6
- CWIMAQVision.GetPointsOnLine
  - method, 5-11
- CWIMAQVision.GrayMorphology
  - method, 2-10

- CWIMAQVision.Histogram2 method, 2-7
- CWIMAQVision.IntegerToColorValue method, 3-8
- CWIMAQVision.Inverse method, 2-9
- CWIMAQVision.Label method, 3-7
- CWIMAQVision.LearnCalibrationGrid method, 6-5, 6-6
- CWIMAQVision.LearnCalibrationPoints method, 6-5, 6-6
- CWIMAQVision.LearnColor method, 3-8, 5-20
- CWIMAQVision.LearnColorPattern method, 5-18, 5-19, 5-24
- CWIMAQVision.LearnColorPatternOptions.IgnoreBlackAndWhite method, 5-20
- CWIMAQVision.LearnColorPatternOptions.IgnoreColorSpectra method, 5-20
- CWIMAQVision.LearnPattern method, 5-12, 5-15
- CWIMAQVision.LineProfile2 method, 2-7
- CWIMAQVision.LowPass method, 2-9
- CWIMAQVision.MastToRegions method, 3-6
- CWIMAQVision.MatchColor method, 3-9
- CWIMAQVision.MatchColorPattern method, 5-18, 5-24
- CWIMAQVision.MathLookup method, 2-9
- CWIMAQVision.Morphology method, 4-3, 4-4
- CWIMAQVision.MultiThreshold2 method, 4-2
- CWIMAQVision.NthOrder method, 2-9, 2-10
- CWIMAQVision.Particle method, 4-5
- CWIMAQVision.ParticleFilter method, 4-4
- CWIMAQVision.Quantify method, 3-7
- CWIMAQVision.Rake method, 5-12
- CWIMAQVision.ReadBarcode method, 5-27
- CWIMAQVision.ReadImage method, 2-6
- CWIMAQVision.ReadImageAndVisionInfo method, 2-6, 5-28, 6-10
- CWIMAQVision.ReadLCD method, 5-27
- CWIMAQVision.ReadMeter method, 5-26
- CWIMAQVision.RegionsProfile method, 2-7, 5-11
- CWIMAQVision.RejectBorder method, 4-3
- CWIMAQVision.RemoveParticle method, 4-3
- CWIMAQVision.ReplaceColorPlanes method, 3-7
- CWIMAQVision.ReplaceComplexPlane method, 2-12
- CWIMAQVision.SelectParticles method, 4-5
- CWIMAQVision.Separation method, 4-4
- CWIMAQVision.SetCalibrationInformation method, 2-7, 6-10
- CWIMAQVision.SetSimpleCalibration method, 6-9
- CWIMAQVision.SimpleEdge method, 5-11
- CWIMAQVision.Spoke method, 5-12
- CWIMAQVision.Threshold method, 4-2
- CWIMAQVision.UserLookup method, 2-9
- CWIMAQVision.WriteImageAndVisionInfo method
  - saving calibration information, 2-7, 6-10
  - saving image with overlay information, 5-28
  - saving template image, 5-15, 5-20, 5-24
- CWMachineVision control, 1-4
- CWMachineVision.FindCircularEdge method, 5-10
- CWMachineVision.FindConcentricEdge method, 5-9
- CWMachineVision.FindCoordTransformUsingPattern method, 5-6
- CWMachineVision.FindCoordTransformUsingRect method, 5-3
- CWMachineVision.FindCoordTransformUsingTwoRects method, 5-4
- CWMachineVision.FindStraightEdge method, 5-9
- CWMachineVision.GetSelected<shapename>FromViewer method, 5-8

CWMachineVision.LightMeterLine  
method, 3-7

CWMachineVision.LightMeterPoint  
method, 3-7

CWMachineVision.LightMeterRectangle  
method, 3-7

CWMachineVision.MeasureMaximum  
Distance method, 5-25

CWMachineVision.MeasureMinimum  
Distance method, 5-25

CWMachineVision.SetupViewerFor  
<shapename>Selection method, 5-8

CWMOVFindCTUsingPatternOptions  
object, 5-6

cwmv.ocx, 1-4

## D

displaying  
images, 2-6 to 2-7  
results of inspection process, 5-27 to 5-29

distance measurements, 5-25

distortion, correcting. *See* calibration.

documentation  
conventions used in manual, *iv*  
documentation resources and  
examples, 1-1 to 1-2

## E

edge detection  
building coordinate reference, 5-3 to 5-5  
finding measurement points, 5-9 to 5-12  
along multiple search contours, 5-12  
along one search contour,  
5-11 to 5-12  
lines or circles, 5-10 to 5-11

error map, for calibration, 6-8

## F

Fast Fourier Transform (FFT), 2-11 to 2-12

filters  
convolution, 2-10  
highpass, 2-10  
highpass frequency filters, 2-12  
improving images, 2-9 to 2-10  
lowpass, 2-9  
lowpass frequency filters, 2-12  
Nth order, 2-10

finding measurement points. *See* measurement  
points, finding.

Free Region tool (table), 3-3

Freeline tool (table), 3-3

frequency domain, 2-11

## G

geometrical measurements, 5-26

grab  
definition, 2-5  
example, 2-5

grayscale and color measurements, 3-1 to 3-13  
color statistics, 3-7 to 3-13  
comparing colors, 3-8 to 3-9  
learning color information,  
3-9 to 3-13  
primary components of color image  
(figure), 3-8

defining regions of interest, 3-1 to 3-6  
interactively, 3-1 to 3-5  
programmatically, 3-5 to 3-6  
using masks, 3-6

grayscale statistics, 3-8

grayscale morphology, 2-10 to 2-11

grayscale statistics, 3-6

**H**

- highpass filters, 2-10
- highpass frequency filters
  - attenuation, 2-12
  - truncation, 2-12

**I**

- ignoring colors
  - learned colors, 3-13
  - pattern matching, 5-20
- images. *See also* blob analysis.
  - acquiring or reading, 2-4 to 2-6
  - analyzing, 2-7 to 2-8
  - attaching calibration information, 2-7, 6-10 to 6-11
  - complex image operations, 2-12
  - converting array to image, 2-6
  - creating, 2-2 to 2-3
    - overview, 2-2 to 2-3
    - valid image types, 2-2 to 2-3
  - CWIMAQImage object, 2-2
  - displaying, 2-6 to 2-7
  - improving, 2-9 to 2-12
    - complex image operations, 2-12
    - FFT (Fast Fourier Transform), 2-11 to 2-12
    - filters, 2-9 to 2-10
    - grayscale morphology, 2-10 to 2-11
    - lookup tables, 2-9
- imaging system
  - calibrating, 2-2
  - setting up, 2-1 to 2-2
- IMAQ Vision for Visual Basic
  - creating applications, 1-5 to 1-6
    - general steps (figure), 1-5
    - inspection steps (figure), 1-6
  - organization, 1-3 to 1-4
    - ActiveX objects, 1-4
    - controls, 1-3 to 1-4

- IMAQ Vision products, 1-1
- instrument reader measurements, 5-26 to 5-27
- invalidation of calibration, 6-9

**L**

- learning calibration information, 6-5 to 6-9
  - choosing learning algorithm, 6-6 to 6-7
  - choosing region of interest, 6-6
  - correction table, 6-8
  - error map, 6-8
  - invalidation of calibration, 6-9
  - learning score, 6-7 to 6-8
  - scaling mode, 6-8
  - specifying scaling factors, 6-6
- learning color information, 3-9 to 3-13
  - choosing color representation
    - sensitivity, 3-12
  - choosing right color information, 3-9 to 3-10
  - entire image, 3-10
  - ignoring learned colors, 3-13
  - multiple regions in image, 3-11 to 3-12
  - region in image, 3-10 to 3-11
  - specifying information to learn, 3-10 to 3-12
- Line tool (table), 3-2
- lines, finding points along edge, 5-9 to 5-11
- locating objects to inspect. *See* machine vision.
- lookup table transformations, 2-9
- lowpass filters, 2-9
- lowpass frequency filters
  - attenuation, 2-12
  - truncation, 2-12



**M**

machine vision, 5-1 to 5-28

- converting pixels coordinates to real-world coordinates, 5-25
- defining region of interest for search area, 5-8 to 5-9
  - interactively, 5-8
  - programmatically, 5-9
- displaying results, 5-27 to 5-28
- finding measurement points, 5-9 to 5-24
  - color location, 5-24
  - color pattern matching, 5-18 to 5-24
  - edge detection, 5-9 to 5-12
  - pattern matching, 5-12 to 5-18
- locating objects to inspect, 5-2 to 5-7
  - choosing method for building coordinate reference (figure), 5-7
  - edge detection for building coordinate reference, 5-3 to 5-5
  - pattern matching for building coordinate reference, 5-6
- making measurements, 5-25 to 5-27
  - analytic geometry measurements, 5-26
  - distance measurements, 5-25
  - instrument reader measurements, 5-26 to 5-27
- overview, 5-1 to 5-2
- steps for performing (figure), 5-2

masks, for defining regions of interest, 3-6

measurement points, finding, 5-9 to 5-24

- color location, 5-24
- color pattern matching, 5-18 to 5-24
- edge detection, 5-9 to 5-12
- pattern matching, 5-12 to 5-18

measurement-ready images, acquiring, 2-1 to 2-12

- acquiring or reading images, 2-4 to 2-6
  - continuous acquisition, 2-5 to 2-6
  - converting array to image, 2-6

- one-shot acquisition, 2-4 to 2-5
- reading files, 2-6

analyzing images, 2-7 to 2-8

attaching calibration information, 2-7

calibrating imaging system, 2-2

creating images, 2-2 to 2-3

displaying images, 2-6 to 2-7

improving images, 2-9 to 2-12

- complex image operations, 2-12
- FFT (Fast Fourier Transform), 2-11 to 2-12
- filters, 2-9 to 2-10
- grayscale morphology, 2-10 to 2-11
- lookup tables, 2-9

setting up imaging system, 2-1 to 2-2

**N**

NI Developer Zone, A-1

Nth order filters, 2-10

**O**

one-shot acquisition, 2-4 to 2-5

Oval tool (table), 3-2

**P**

Pan tool (table), 3-3

particle measurements, 4-4 to 4-7

pattern matching. *See also* color pattern matching.
 

- building coordinate reference, 5-6
- finding measurement points, 5-12 to 5-18
  - defining and creating template images, 5-13 to 5-14
- defining search area, 5-15 to 5-16
- general steps, 5-12 to 5-13
- setting matching parameters and tolerances, 5-16 to 5-17

- testing search algorithm on test images, 5-17
- training the algorithm, 5-15
- verifying results with ranking method, 5-17 to 5-18

perspective errors, calibrating. *See* calibration.

pixel coordinates, converting to real-world coordinates, 4-7, 5-25

Point tool (table), 3-2

points, finding. *See* measurement points, finding.

Polygon tool (table), 3-3

## R

ranking method for verifying pattern matching, 5-17 to 5-18

reading

- files, 2-6
- images, 2-4 to 2-6

Rectangle tool (table), 3-2

regions of interest, defining, 3-1 to 3-6

- calibration, 6-6
- interactively, 3-1 to 3-5
  - for machine vision inspection, 5-8
  - tools palette functions (table), 3-2 to 3-3
- methods for adding shapes to regions (table), 3-5
- programmatically, 3-5 to 3-6
  - for machine vision inspection, 5-9
- using masks, 3-6

ring operation

- definition, 2-5
- example, 2-5 to 2-6

ROI. *See* regions of interest, defining.

Rotated Rectangle tool (table), 3-2

## S

scaling factors, specifying for calibration, 6-6

scaling mode, for calibration, 6-8

search contour, finding points along edge, 5-11 to 5-12

Selection tool (table), 3-2

sequence

- asynchronous sequence (example), 2-5
- definition, 2-4

snap

- definition, 2-4
- synchronous snap (example), 2-4

statistics, measuring

- color statistics, 3-7 to 3-13
- grayscale statistics, 3-8

system integration, by National Instruments, A-1

## T

technical support resources, A-1 to A-2

template for calibration, defining, 6-2

template images

- defining
  - color pattern matching, 5-18 to 5-19
  - pattern matching, 5-13 to 5-14
- training
  - color pattern matching, 5-19 to 5-20
  - pattern matching, 5-15

tools palette functions (table), 3-2 to 3-3

truncation

- highpass, 2-12
- lowpass, 2-12

## V

verifying pattern matching, 5-17 to 5-18

**W**

Web support from National Instruments, A-1  
Worldwide technical support, A-2

**Z**

Zoom tool (table), 3-3