EPSON

EPSON RC+ 7.0 Option

Vision Guide 7.0 Ver. 7.4

Software

Rev.1   EM186S3684F

EPSON RC+ 7.0 Option   Vision Guide 7.0 (Ver.7.4) Software Rev.1

EPSON RC+ 7.0 Option

Vision Guide 7.0 *(Ver.7.4) Software*

Rev.1

FOREWORD

Thank you for purchasing our robot products. This manual contains the information necessary for the correct use of the EPSON RC+ Option Vision Guide.

Please carefully read this manual and other related manuals when using this software.

Keep this manual in a handy location for easy access at all times.

WARRANTY

The robot and its optional parts are shipped to our customers only after being subjected to the strictest quality controls, tests and inspections to certify its compliance with our high performance standards.

Product malfunctions resulting from normal handling or operation will be repaired free of charge during the normal warranty period. (Please ask your Regional Sales Office for warranty period information.)

However, customers will be charged for repairs in the following cases (even if they occur during the warranty period):

1. Damage or malfunction caused by improper use which is not described in the manual, or careless use.

2. Malfunctions caused by customers' unauthorized disassembly.

3. Damage due to improper adjustments or unauthorized repair attempts.

4. Damage caused by natural disasters such as earthquake, flood, etc.

Warnings, Cautions, Usage:

1. If the robot or associated equipment is used outside of the usage conditions and product specifications described in the manuals, this warranty is void.

2. If you do not follow the WARNINGS and CAUTIONS in this manual, we cannot be responsible for any malfunction or accident, even if the result is injury or death.

3. We cannot foresee all possible dangers and consequences. Therefore, this manual cannot warn the user of all possible hazards.

SOFTWARE LICENSE

For Compact Vision users, please read this software license agreement carefully before using this option.

Hardware & Setup manual:
   Appendix A: End User License Agreement for Compact Vision
   Appendix B: Open Source Software License for Compact Vision

TRADEMARKS

Microsoft, Windows, Windows logo, Visual Basic, and Visual C++ are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Other brand and product names are trademarks or registered trademarks of the respective holders.

TRADEMARK NOTIFICATION IN THIS MANUAL

Microsoft® Windows® XP Operating system

Microsoft® Windows® Vista Operating system

Microsoft® Windows® 7 Operating system

Microsoft® Windows® 8 Operating system

Microsoft® Windows® 10 Operating system

Throughout this manual, Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10 refer to above respective operating systems. In some cases, Windows refers generically to Windows XP, Windows Vista, Windows 7, Windows 8, and Windows 10.

NOTICE

No part of this manual may be copied or reproduced without authorization.

The contents of this manual are subject to change without notice.

Please notify us if you should find any errors in this manual or if you have any comments regarding its contents.

MANUFACTURER

SEIKO EPSON CORPORATION

SAFETY PRECAUTIONS

Installation of robots and robotic equipment should only be performed by qualified personnel in accordance with national and local codes. Please carefully read this manual and other related manuals when using this software.

Keep this manual in a handy location for easy access at all times.

⚠ WARNING	This symbol indicates that a danger of possible serious injury or death exists if the associated instructions are not followed properly.
⚠ CAUTION	This symbol indicates that a danger of possible harm to people or physical damage to equipment and facilities exists if the associated instructions are not followed properly.

6. Vision Objects   47

7. Vision Calibration   266

1. Manual Help

1.1 Vision Guide Manual Construction

Overview

This section provides general information on this manual. Online help, safety features, and reference cases are also described for understanding of the basic features of the EPSON RC+ 7.0

How to Install:

This section describes the necessary system, product configuration of Vision Guide 7.0, and how to install the hardware and software.

Quick Start: First Vision Guide 7.0 Application

This section describes for the users first using the Vision Guide 7.0 how to use it using the sample applications. It thoroughly explains the use of the Vision Guide 7.0, from the creation of a new vision object, calibration of the Vision Guide 7.0 mobile camera, and actual robot motion to the parts detected by Vision Guide 7.0.

Vision Guide Window

This section shows the layout and gives a usage explanation for the Vision Guide window. It also includes information on the Vision Guide 7.0 toolbar, Image Display, Run Panel, the Object, Sequence, and Calibration tabs.

Vision Objects

This section describes the different types of vision tools available with Vision Guide 7.0 and how to use them.

Histogram and Statistics Tools

This section describes the usage of Histogram for various vision object types including Blob, Correlation, and Polar objects.
It also describes the Vision Guide statistics tools from the Vision Guide window with the Statistics dialog and from the SPEL+ Language through accessing statistics properties.

Vision Sequences

This section describes what vision sequences are, how to use and apply them, and also explains about debugging techniques for Vision Guide Sequences.

Calibration

This section explains the usage for the various calibration types.

Using Vision Guide 7.0 with SPEL+

This section shows how to run vision sequences from the SPEL+ language and how to access vision properties and results. It also explains how to use Vision Guide 7.0 results for robot guidance.

1.2 Related Manuals

Refer to the following related manuals along with the Vision Guide 7.0 manuals for using the Vision Guide 7.0.

Vision Guide 7.0 Hardware & Setup

   This manual contains description on proper usage, operating precautions, and warnings for setting up the hardware of the Vision Guide.

Vision Guide 7.0 Properties & Result Reference

   This manual contains a complete reference of all the properties and results available for vision sequences and vision objects. It contains detailed information relating to the proper usage, cautions, and warnings for each property and result

EPSON RC+7.0 User's Guide

   This manual contains information on using the EPSON RC+ Robot Control System.

SPEL⁺ Language Reference Manual

   This manual contains a complete description of all commands for the SPEL⁺ language.

1.3 Using On-Line Help

EPSON RC+ 7.0 supports the On-Line Help system. The help system makes it easy to find information than the conventional method using manuals.

There are several ways to refer to the on-line help in EPSON RC+ 7.0:

- Press the F1 function key at any time for context sensitive help. Help will be displayed for the current item you are working with. This is very useful when you need information for a certain item in a screen or dialog. If you are editing a program, the help information for the SPEL⁺ keyword at the cursor position will be displayed. You can use the on-line help for referring syntax information to use the SPEL⁺ language.

- Click on the <Help> button in the dialog box, if available.

- To view the table of contents and select topics, select Contents from the Help menu. Topics can be selected by clicking on the underlined text that is highlighted in green. (This causes a jump to the topic of interest.)

- Select Contents from the Help menu, then press <S> or click the <Search> button to search for information on a specific topic.

Once you are in the on-line help you will notice that some items are highlighted in green and underlined. These are hypertext links and when you click on this highlighted text, the system will jump to the area in the Help System that is related to the highlighted text. You will also notice that some text is highlighted in green with dotted underlines. Clicking on this type of text will cause a small popup window to appear with a more detailed description of the highlighted text and possibly related information that you can jump to.

Most of the information found in this manual is also available in the Vision Guide 7.0 Help System although it may be arranged a little differently to provide the proper hypertext links and ease of use.

2. Safety

Please read this manual before using the Vision Guide.

Keep this manual handy for easy access at all times and reread it when you find anything unclear.

2.1 Conventions

Important safety considerations are indicated throughout the manual by the following symbols. Be sure to read the descriptions shown with each symbol.

⚠ WARNING	This symbol indicates that a danger of possible serious injury or death exists if the associated instructions are not followed properly.
⚡ WARNING	This symbol indicates that a danger of possible harm to people caused by electric shock exists if the associated instructions are not followed properly.
⚠ CAUTION	This symbol indicates that a danger of possible harm to people or physical damage to equipment and facilities exists if the associated instructions are not followed properly.

2.2 Safety Precautions

⚡ WARNING	■ Do not use the Compact Vision for the purpose to ensure safety. ■ The product must be used within the conditions described in this manual. Using the product in an environment that exceeds the specified environmental conditions may not only shorten the life cycle of the product but may also cause serious safety problems.

⚠ CAUTION	■ Purchase cameras and camera cables from our suppliers. Note that cameras and camera cables of other manufacturers are not included in the warranty.

⚠ CAUTION	■ The result of the image processing may be affected by the change of light source, ambient light or peripheral electromagnetic noise. If the manipulator is operated based on the result of the image processing, the robot system should be designed considering that the robot may operate in reaction to the whole field of camera view.

2.3 Robot Safety

Whenever you are working with robots or other automation equipment, safety must be the top priority. The EPSON RC+ 7.0 system has many safety features built in, such as E-Stop, Robot Pause Input, and a Safety Guard Input. These safety features should be used when designing the robot cell.

Refer to the *Safety* chapter in *the EPSON RC+ 7.0 User's Guide* for safety information and guidelines.

3. Introduction

3.1 Vision Guide 7.0 Overview

3.1.1 Purpose of Vision Guide 7.0

Having its primary focus to provide the vision tools necessary to solve robot guidance applications, Vision Guide 7.0 is capable for many different types of machine vision applications. Some of the typical applications for Vision Guide 7.0 are locating and moving parts, part inspections, and gauging applications. In fact, almost any application that requires motion and vision integrated together is a good candidate application for Vision Guide 7.0 and EPSON robots.

One of the primary differences between Vision Guide 7.0 and other vision systems is that Vision Guide 7.0 was made as an integrated part of the EPSON robot systems. As a result you will find many features such robot to camera calibration procedures built into the Vision Guide 7.0 system. The net result is a great reduction in time to complete your vision-based application.

3.1.2 Features of Vision Guide

Some of the primary features/tools provided with Vision Guide 7.0 include the following:

- Integrated tools that support several camera orientations and calibrations. Calibration is made much simpler than with other robot vision systems and more accurate.

- GUI operation for quick prototyping where the prototype vision sequences can actually be used in the final application.

- Complete integration into the EPSON RC+ 7.0 programming and development environment.

- Blob analysis tools that measure the size, shape, and position of objects with variations. There are also tools that count the number of holes within and tell the roundness of a given blob.

- Geometric pattern search that searches for a model based on geometric part features.

- Normalized correlation search tool that locates objects under varying light conditions using an advanced template matching technique.

- Edge detection tool that locates a specific edge with sub-pixel accuracy.

- LineFinder tool and ArcFinder tool that detect the edges of linear and circular objects easily and that enable accurate positioning.

- Polar Search is a high-speed angular search tool that can quickly measure the rotation of complex objects. This tool is very useful for vision guided robot pick and place applications.

- DefectFinder tool that performs a differential test using template images to detect image defects.

- Line and point tools that provide a mechanism for creating lines between points and measurement capabilities as well.

- Frame tool that allows all vision tools to be dynamically located based on a frame of reference.

- An object reference mechanism that allows one vision tool's position to be based on another vision tool's result thus saving hours of development time.

- Histogram charts provide a powerful mechanism for looking more closely at pixel data as well as setting proper thresholds for tools that require it.

- Statistics calculations are built in to provide mean, standard deviation, range, min and max values and a variety of other statistics that can be referenced for each vision tool at design-time or runtime.

- Automatic compensation for ngular discrepancies in camera lens, camera, and robot peripherals.

- Support for color cameras and images by a ColorMatch tool and a ColorFilter operation.

3.2 Assumptions about the User's Knowledge of EPSON RC+ 7.0

Vision Guide 7.0 is an optional addition to the core EPSON RC+ 7.0 Environment. In order to use Vision Guide 7.0, you should be familiar with the EPSON RC+ 7.0 Development and EPSON Robots. For the purposes of this manual, it is assumed that the user is already familiar with the following items:

- EPSON RC+ 7.0 Project Management concepts and usage.

- Creating and editing SPEL+ programs with the EPSON RC+ 7.0 program editor.

- Running SPEL programs from the Run window.

- The basic SPEL+ language constructs such as functions, variable usage, etc.

Those users not familiar with SPEL+ should attend an EPSON RC+ 7.0 training class.

4. The Vision Guide Environment

4.1 Overview

In this chapter we will focus on some concepts and definitions so you can gain a complete understanding of Vision Guide 7.0 and its components. We will cover the following topics:

- Basic definitions which you should understand to use Vision Guide 7.0.
- How to open the Vision Guide 7.0 window.
- An explanation of the Vision Guide 7.0 video display.
- An introduction to all the vision objects and the buttons on the toolbar to manipulate them.
- An introduction to properties and results.
- An explanation of the Sequence, Object, and Calibration tabs
- How to execute a vision object or a vision sequence and an explanation about when to use each.

4.2 Basic Concepts Required to Understand Vision Guide 7.0

Following is a quick explanation of some of the basic concepts which helps you to understand the contents of this chapter better.

4.2.1 Vision Sequence

A vision sequence is a grouping of vision objects in a specific order that can be executed from the Vision Guide window or from the SPEL+ language.

A vision sequence has specific properties that are used to set up vision sequence execution. For example, the Camera property defines which camera will be used to capture an image for this vision sequence and the RuntimeAcquire property defines how the image will be acquired for this vision sequence.

A vision sequence can be thought of as a type of container that holds all the necessary vision objects required to solve a specific Vision process or part of a process. In general a vision sequence is the starting point for all Vision Processing.

4.2.2 Vision Object

A vision object is a vision tool that can be applied to an image that is captured by the camera. Some of the vision objects supported are: Geometric Search, Correlation Search, Blob Analysis, Polar Search, Edge Detection, Line Search, Arch Search, Differential Test, Line Creation, Points, and Frames.

All vision objects can be executed (applied to the current Image) and all return results such as how much time the vision object took to execute, position information, angle information, whether or not the vision object was even found, and whether or not the object was accepted. Vision objects have properties that are used to define the characteristics of how the vision object will perform. They also have results that are the values that are returned after a vision object is executed.

4.2.3 Property

Properties can be thought of as parameters that are set for each vision object or vision sequence.

The setting of vision properties can be done in a point and click fashion from the Vision Guide window that provides for a quick method to build and test vision applications. Vision properties can also be set and checked from the SPEL+ language. This provides the flexibility required for dynamic modifications of vision objects at runtime.

Vision sequence and vision object properties are very powerful because they are easily modified and help make vision objects easier to understand and use while at the same time they don't limit the flexibility required for more complex applications.

4.2.4 Result

Results are the values that are returned from vision objects or vision sequences after they are executed. Examples of commonly used results include:

Time   returns how much time the vision object or sequence took to execute

RobotXYU   returns the X, Y, and U position of the found feature in robot coordinates.

Found   returns whether or not the vision object was found

Passed   returns whether or not the result of the vision object was accepted.

Vision results can be seen from within the Vision Guide window on the Sequence and Object tabs. SPEL+ programs can also use vision results.

4.2.5 Runtime Vision Commands

A series of vision commands have been added to the SPEL+ robot language to provide a seamless integration of robot motion and vision guidance.

Commands such as VRun allow a user to initiate a vision sequence from the SPEL+ language with just one function call.

VGet allows a user to get the results that are returned from vision objects, sequences, and calibrations. This is very powerful because vision sequences are created, modified and maintained from within the Vision Guide 7.0 point and click development environment but all things created within Vision Guide 7.0 are also accessible from the SPEL+ language.

Required Vision Hardware

To use the Vision Guide 7.0, either of the following hardware is required.

 - Compact Vision CV1 (with the firmware ver. 2.1.0.0 or later)

 - Compact Vision CV2 (with the firmware ver. 2.3.0.0 or later)

 - PC Vision PV1

For details on hardware, refer to *Vision Guide 7.0 Hardware & Setup* manual.

4.2.6 Vision Guide 7.0 in EPSON RC+ 7.0 Projects

EPSON RC+ 7.0 is based on Projects that are containers of all the necessary programs, teach points, and robot settings required for a specific robot application.

This Project configuration makes it easy to use one robot for multiple projects or test environments to check out new ideas without destroying your old working applications.

When you create an EPSON RC+ 7.0 application which includes Vision Guide 7.0, all the associated vision sequences and vision objects required for that application are kept within the project along with the other items normally contained in projects. This ensures that when you open an existing project, everything relating to this project is available to you.

4.3 Coordinate Systems

The section describes the coordinate systems used in Vision Guide 7.0.

4.3.1 Image Coordinate System

The image coordinate system is used for the camera video buffer and video display. Units are in pixels.
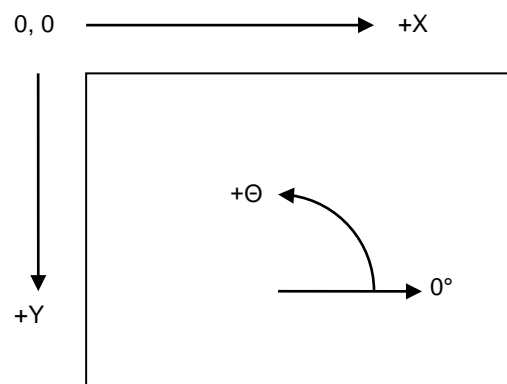
4.3.2 Camera Coordinate System

The camera coordinate system is physical coordinates in the camera field of view. Units are in millimeters. To obtain coordinates in this system, a camera must be calibrated using any of the camera orientations.
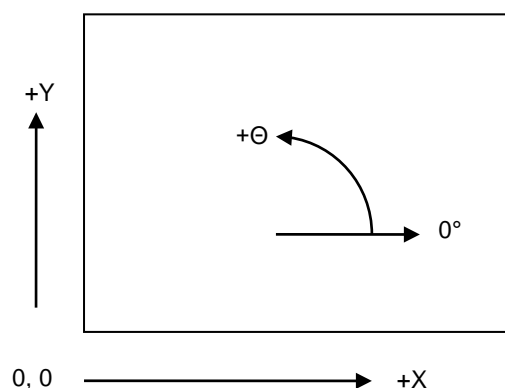
4.3.3 Robot Coordinate System

The robot coordinate system is physical coordinates which the robot has. Units are in millimeters. To obtain coordinates in this system, a camera must be calibrated using camera orientation in the robot world. For details on the robot coordinate system, refer to the *EPSON RC+ 7.0 User's Guide* manual.

Robot Coordinate System of a SCARA Robot

Robot coordinate system Z axis

Robot coordinate system U axis
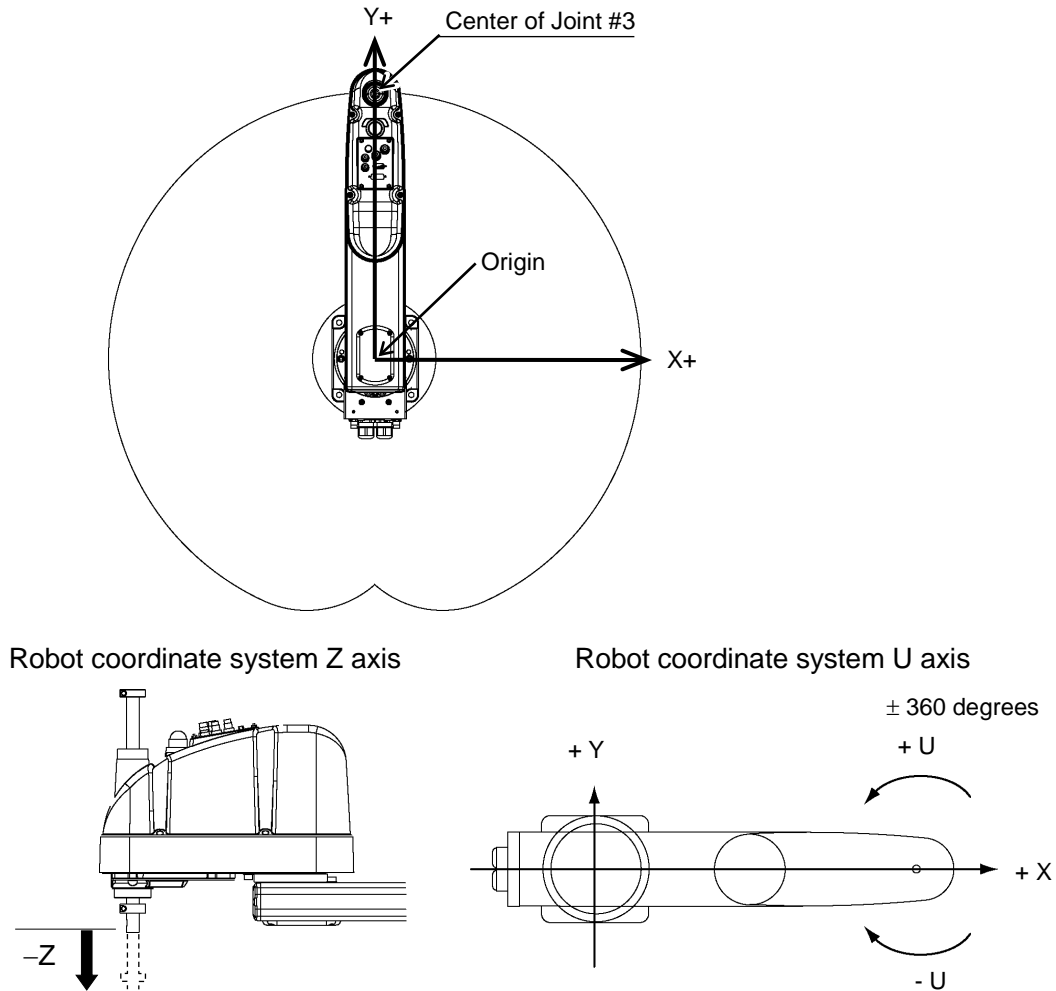
4.4 Open the Vision Guide Window

The Vision Guide window is opened from within the EPSON RC+ 7.0 development environment.

After EPSON RC+ 7.0 has been started, the Vision Guide window can be opened in 2 different ways:

From the main Toolbar: From the main toolbar in EPSON RC+ 7.0 you should see the Vision Guide icon 📷. Clicking on the Vision Guide icon will open the Vision Guide window.

From the Tools menu: Selecting Vision from the Tools menu will open the Vision Guide window.

Once the Vision Guide window is open we can now begin using Vision Guide 7.0. The next few pages describe the basic parts of the Vision Guide window.

4.5 The Parts of the Vision Guide Window

The Vision Guide window is the point and click environment where most of your vision development will take place.

An understanding of this window and its major parts is the first thing you will need to grasp in order to use Vision Guide 7.0. The Vision Guide window can be broken into the following major sections:

- Title Bar

- Toolbar

- Flow Chart

- Sequence and calibration trees

- Image Display

- Sequence, Object, Calibration, Jog, and Robot Tabs

- Jog and Robot tabs

- Run Panel
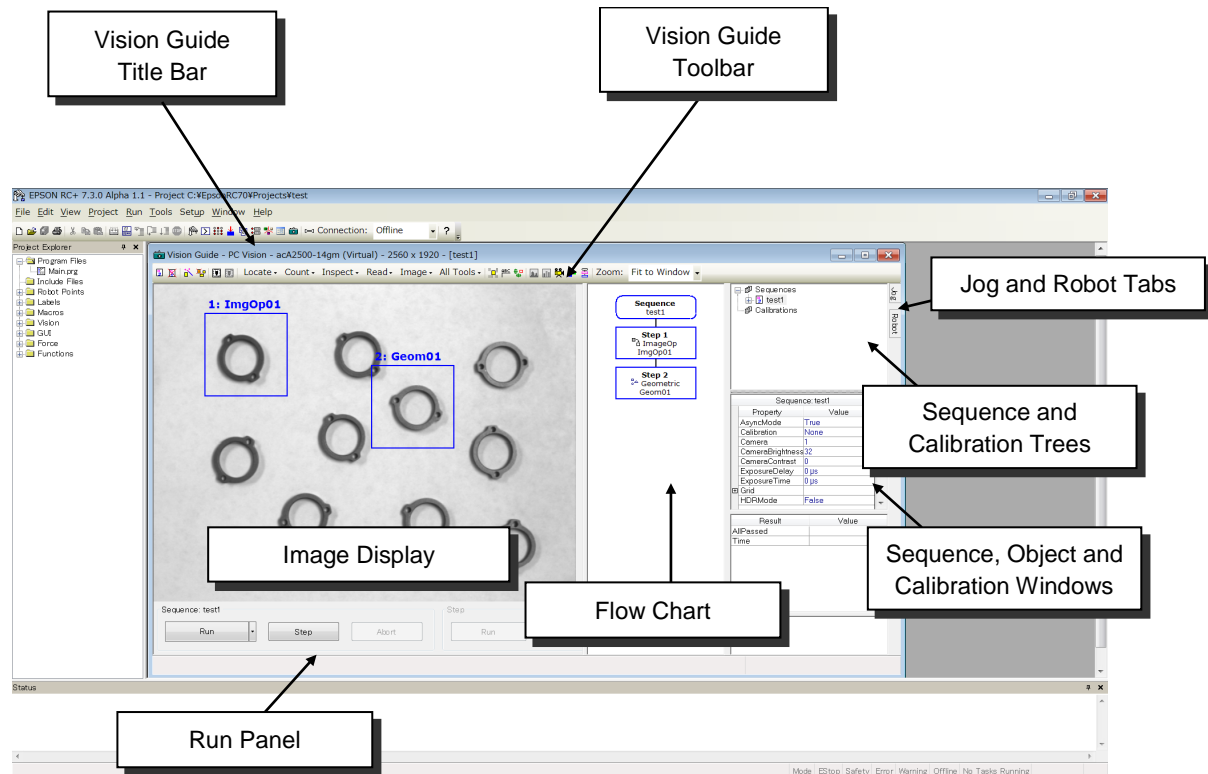
4.5.1 The Title Bar

The Vision Guide window has a title bar that includes the title of the window, the camera type and resolution, and the name of the current sequence. The vision sequence name is put in brackets. A sample title bar is shown below. It shows that we are running Vision Guide 7.0 with a fixed camera and currently using the "blobtut" vision sequence.
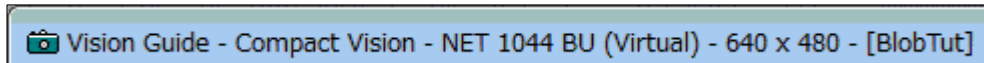
It is important to understand the difference between the main title bar for the EPSON RC+ 7.0 environment and the Vision Guide title bar. The EPSON RC+ 7.0 title bar contains the name of the current project while the Vision Guide window title bar contains the name of the current sequence.

4.5.2 The Toolbar

Toolbars are typically included with Microsoft Windows^{TM} applications because they provide quick access to many of the more common features of the product.

Both EPSON RC+ 7.0 and Vision Guide 7.0 make use of toolbars.

The Vision Guide 7.0 Toolbar is located at the top of the Vision Guide window just below the Title Bar and appears as follows:
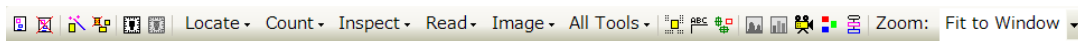
Notice that the toolbar buttons for the Vision Guide 7.0 window are separated into small groups. This is done to make them easier to find and use. A general description for each of the groups of toolbar buttons is as follows:

- The first two toolbar buttons are grouped together because they are used for creating and deleting vision sequences.

- The next two toolbar buttons are used to run the step wizard and delete the objects.

- The next two toolbar buttons are used to create or delete a calibration.

- The next toolbar button group is used to select the vision object category (detection, count, inspection, read, image, or All Tools).

- The next toolbar buttons that are grouped together can be thought of as utilities for the Vision Guide 7.0 Environment. They include Show Only Current Object, Delete Object, Force All Labels Off, and Force All Graphics On.

- The next two toolbar buttons are for opening the Histogram and Statistics dialogs.

- The last two buttons are for switching between live and frozen video, and for switching between color and grayscale images.

- The last toolbar button is used to enable or disable display of the flow chart.

The Vision Guide 7.0 toolbar includes tool tips for each toolbar button like the one shown here.
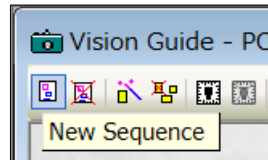
To see the description for a specific toolbar button, move the mouse pointer over the toolbar button and after about 2 seconds you will see the tool tip for that specific toolbar button.

Followings are general descriptions for each of the Vision Guide 7.0 toolbar buttons.

Button	Description
	New sequence: Creates a vision sequence. A dialog pops up and the user is asked to enter the name for the new sequence.
	Delete sequence: Deletes a vision sequence in the current project. This button is dimmed if there are no vision sequences for the current project.
	Show only current object: When pressed in, only the current selected object is displayed. This is useful when you want to work with one object without interference from other objects.
	Delete object: Deletes the current active vision object. To delete a vision object select the desired vision object and then click on this button. This button is dimmed if there are no vision sequences for the current project or if there are no vision objects defined for the current vision sequence.
	Force all labels off: When pressed in, this button causes the labels on the vision objects to be removed. This feature is useful when you have many vision objects close together and it's difficult to distinguish between them. This button is dimmed if there are no vision sequences for the current project.
	New step: When pressed, the step wizard appears. By using the step wizard, the objects can be added. Unlike the menu bar, object's name and its order in the sequence can be configured in the step wizard before adding the object.

Button	Description
	Force all graphics on: When pressed, this button causes all graphics (search window, model origin, model window, Lines, and Labels) for the vision objects to be displayed. This button overrides the setting of the Graphics property for each individual vision object making it easy to quickly see all vision objects rather than modifying the Graphics property for each vision object individually.
	Histogram: Clicking on this button opens the Histogram dialog. This button is dimmed if there are no vision sequences for the current project or if there is no vision object for the current vision sequence.
	Statistics: Clicking on this button opens the Statistics dialog. This button is dimmed if there are no vision sequences for the current project or if there is no vision object for the current vision sequence.
	New calibration: Opens the Calibration dialog to create a new calibration. This button is disabled if there are no vision sequences for the current project.
	Delete calibration: Opens the Delete Calibration dialog to delete a calibration. This button is disabled if there are no vision calibrations for the current project.
	Freeze image: Toggles between live and frozen image.
	Color / Grayscale: Toggles between color and grayscale image displays.
	Flow Chart: Enables or disables display of the flow chart.
Locate ▾	**Locate:** The detection category object can be selected. (Geometric, Correlation, Blob, Edge, Polar, Arc Finder, Line Finder, Frame, Line, Point)
Count ▾	**Count:** The count category object can be selected. (Blob, Correlation, Geometric)
Inspect ▾	**Inspect:** The inspection category object can be selected. (Blob, Defect Finder, Line, Line Inspector, Arc Inspector, Color Match)
Read ▾	**Read:** The read category object can be selected (Code Reader, OCR)
Image ▾	**Image:** The image category object can be selected. (ImageOp)
All Tools ▾	**All Tools:** All objects can be selected.

Buttons called by selecting categories on the toolbar

Button	Tool tip: Simple description
	ImageOp (Image operation): Creates an ImageOp object. This button is dimmed if there are no vision sequences for the current project.
	Geometric (Geometric shape): Creates a Geometric object. This button is dimmed if there are no vision sequences for the current project.
	Correlation (Correlation search): Creates a Correlation object. This button is dimmed if there are no vision sequences for the current project.
	Blob (Blob analysis): Creates a Blob object. This button is dimmed if there are no vision sequences for the current project.
	Edge (Edge detection): Creates an Edge object. This button is dimmed if there are no vision sequences for the current project.
	Polar (Polar search): Creates a Polar object. This button is dimmed if there are no vision sequences for the current project.
	New OCR: Creates an OCR object. This button is dimmed if there are no vision sequences for the current project. This button is dimmed if there is no license for OCR.
	New code reader: Creates a CodeReader object. This button is dimmed if there are no vision sequences for the current project.
	New color match: Creates a ColorMatch object. This button is dimmed if there are no vision sequences for the current project.
	New LineFinder: Creates a LineFinder object. This button is dimmed if there are no vision sequences for the current project.
	New LineInspector: Creates a LineInspector object. This button is dimmed if there are no vision sequences for the current project.
	New ArcFinder: Creates an ArcFinder object. This button is dimmed if there are no vision sequences for the current project.
	New ArcInspector: Creates an ArcInspector object. This button is dimmed if there are no vision sequences for the current project.
	New DefectFinder: Creates a DefectFinder object. This button is dimmed if there are no vision sequences for the current project.
	Point: Creates a new Point object. This button is dimmed if there are no vision sequences for the current project.
	Line: Creates a new Line object. This button is dimmed if there are no vision sequences for the current project.
	Frame: Creates a new Frame object. This button is dimmed if there are no vision sequences for the current project.

4.5.3 The Image Display

The image display is located directly beneath the Vision Guide 7.0 Toolbar. This is the area of the Vision Guide window where the image brought in through the camera (or from disk) is displayed.

This image is overlaid on the Vision Window to fit within the area we call the image display. The image display is also used to display graphics for each of the vision objects that are used to process the images. For example, you may see boxes and cross hairs displayed at various positions in the image display. These graphics help developers use the vision object Tools.

A picture of the Vision Guide window with the image display labeled is shown in *4.5 The Parts of the Vision Guide Window*.

The image display expands as the Vision Guide GUI window is expanded. The size of the Run dialog display is also changed as the window size changes. There is a splitter bar between the image display and the tab group that allows you to adjust the image display size. The aspect ratio for the image is maintained as you change the size of the image display.

4.5.4 The Vision Guide Window Tabs

The primary purpose of tabs is to provide quick access to data that is related or grouped together in some way. This proves much easier to use than additional menu items or multiple windows.

The Vision Guide window uses tabs to provide a single window vision development environment that makes the system easy to learn and remember.

Located on the right side of the Vision Guide window is a set of two tabs that are labeled "Jog" and "Robot". These tabs are positioned in a fixed location next to the sequence tree and can be used outside the image display.

The two tabs are used in the Vision Guide 7.0 development environment and provide easy access to knowledge and help for usage of Vision Guide 7.0. The details for each of the tabs are described later in this chapter.

4.5.5 The Run Panel

The Run Panel is located just below the image display. The purpose of the Run Panel is to execute and debug sequences from the Vision Guide 7.0 development environment. The Run Panel is shown as below:

Execution of Sequence

Once a vision sequence has been created and vision objects have been added, a vision sequence can be run by clicking on the <Run> button located on the left side of the Run Panel.

Cycles

The number of cycles can be specified in <Cycles> that can be displayed by clicking the ▼ button at the right side of the <Run> button. The vision sequence is executed as many times as specified in this box.

Abort

If at any time you want to stop multiple vision sequence cycles, click the <Abort> button on the Run Panel. The <Abort> button is only enabled when a sequence is actually running. The <Abort> button is also used to abort a sequence that is waiting for a strobe trigger.

Step

The <Step> button allows for single stepping of a vision sequence, where each step will execute one vision object. Click on the <Step> button each time you want to execute the next step.

The first time you click on the <Step> button, the vision objects are put in an inactive step mode where they are drawn with blue dashed lines.

The next time you click the <Step> button, the first object in the sequence is executed, and then the 2nd and so on until the last vision object in the sequence is executed. To reset the step execution back to the beginning, click on the <Run> button on the left side of the Run Panel and run the entire vision sequence.

During step execution, you can see which vision object will execute next from the Step list in the Sequence Tab. The vision object that will execute on the next step is highlighted in blue.

To examine a specific object's result values during execution of a vision sequence, use the flow chart or sequence tree to select the vision object you are interested in. Once a vision object is selected, click the <Run> button on the left side of the Run Panel and the vision sequence will begin execution.

When the object is selected from the flow chart or sequence tree, you can see the results for the specific vision object that you selected.

NOTE
☞

The <Run> button on the left side of the Run Panel provides a quick method to test and debug entire vision sequences without having to run them from SPEL+. You can make changes to a vision sequence, test it and if you don't like the results, you can revert back to the saved version of the vision sequence. For this reason, the <Run> button on the left side of the Run Panel does not automatically save your vision sequences each time you run a vision sequence. However, when you run the vision sequence from the Run window, the vision sequence is automatically saved along with the rest of the project if Auto Save is enabled.

Execute object

After creating a vision sequence and adding a vision object, click the <Run> button which is the second button from the right side on the Run Panel to execute the vision object.

Teach

If the created vision object requires teaching, click the <Teach> button to perform teaching at the teach window part of the object.

If the object does not require teaching, the <Teach> button cannot be clicked.

Calibration

When the calibration project is selected from the sequence tree, only the < Teach Points> and < Calibrate> buttons are shown on the Run Panel.

Clicking the <Teach Points> button displays the point teach window for executing teaching.

The calibration can be executed by clicking the < Calibrate> button after completeing the point teaching. If the point teaching has not been performed, the < Calibrate> button cannot be clicked.

4.5.6 Flow Chart

The flow chart is displayed on the right side of the image display. The flow chart shows a processing flow of the objects in the selected sequence. The first flow indicates the currently selected sequence, and the second and following flows are the objects included in the sequence. The objects are placed in the order of execution.

The frames of flows are usually shown in blue. When the flow for the object you want to select is clicked, the frame color changes to pink. At this point, the object window in the image display also turns to be pink. However, the sequence flow does not turn to be pink even when it is clicked.

When the sequence or object is selected, properties and results for the selected sequence or object are displayed in the property and result lists on the right side of the flow chart. The background color of corresponding nodes turns to be gray.

The frame color of the sequence flow changes to green when all the objects included in the sequence are executed successfully, or red if any of them is failed.

The frame color of the object flow changes to green when it is executed successfully, or red if failed.

Right-clicking the flow calls various operations.

The following operations are available when right-clicking the sequence flow.

   New sequence, delete sequence, change step order…,

   Execute sequence, add new step

The following operations are available when right clicking the object flow.

   Copy, cut, paste, delete, teach the model, and display the model

4.5.7 Sequence and Calibration Trees

The sequence and calibration trees are displayed at the upper right part of the flow chart. All the sequences and calibration projects are displayed in the trees. The sequence and object nodes in the sequence tree can be operated in the same way as the flows in the flow chart.

Clicking the calibration node changes the Run Panel to display the panel for calibration.

4.6 Vision Guide Window Tabs

4.6.1 The Sequence Window

The sequence window can be displayed at the right side of the window by clicking the vision sequence in the flow chart or selecting the sequence in the sequence tree.
The sequence window is used to:
- Set properties for the vision sequence.
- Examine the results for the entire vision sequence.
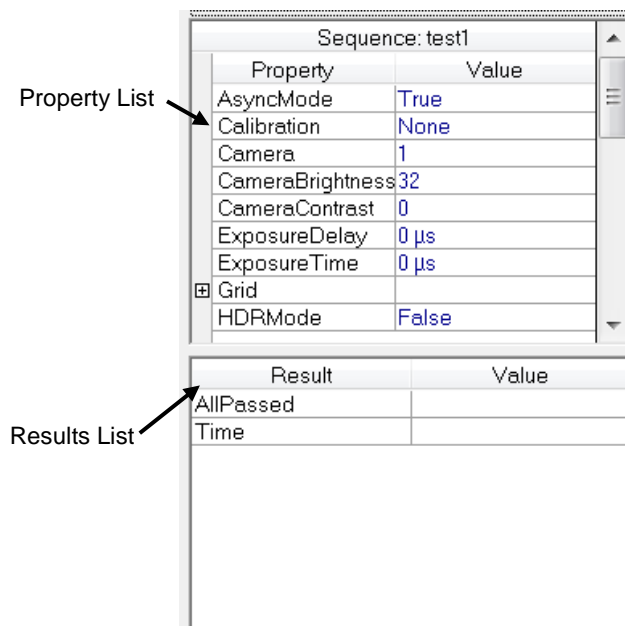
Sequence Window: Properties List

The properties for the vision sequence are shown in the Properties list that looks similar to a small spreadsheet. The name of the property is on the left side and the value is shown on the right.

Vision properties are set by first clicking on the value field for a specific property and then either entering a value or selecting from a list which is displayed.

Some of the vision sequence properties include:

Camera   Specifies which camera to use for this vision sequence.

RuntimeAcquire   Defines the acquisition method for this sequence
                  (i.e. strobed, none, stationary)

Calibration   Defines which calibration to use for this vision sequence

ImageBuffer   Specifies the buffer number to be used in the vision sequence

                  0: Unique buffer numbers that each camera has (Default)

                  1 to 10: Buffers shared in the vision sequence

ImageSize   Specifies the resolution of the camera to be used in the vision sequence

                  Default: Camera resolution

For details on vision sequence properties, refer to *5. Vision Sequences* or the *Vision Guide 7.0 Property and Results Reference Manual*.

Sequence Window: Results List

The results for the vision sequence are shown in the Results list that is located just below the Properties list. The Results list will only display values in the value field after a vision sequence is executed. Prior to execution the result field will not display anything.

The following results are displayed in the result column:

AllPassed   Displays whether or not all vision objects were accepted.

Time   the amount of time it took to process the vision sequence

4.6.2 The Object Window

The Object window can be displayed at the right side of the window by clicking the vision object in the flow chart or selecting the object in the sequence tree.

The Object window is used to:

- Set property values for vision objects.

- View results after execution of vision objects or vision sequences.
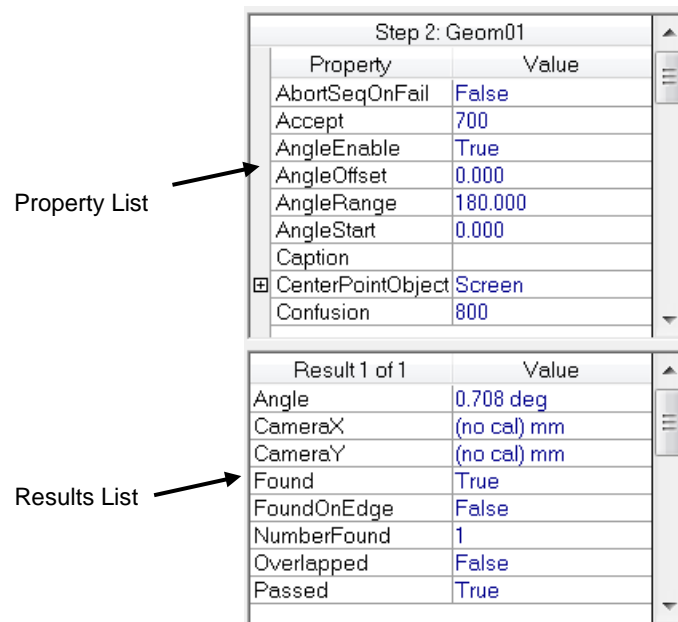
Object Window: Properties List

The properties for vision objects are shown in the Properties list that is similar to a small spreadsheet. The name of the property is on the left side and the value is shown on the right.

Vision object properties can be set by clicking on the Value field for a specific property and then either entering a value or selecting from a list which is displayed.

Vision object properties vary depending upon vision object type but some of the more common vision object properties are:

AbortSeqOnFail   If set to TRUE, the entire vision sequence will abort if this object is not passed during execution

Accept   The threshold value for comparing with the Score result.
If the Score is greater than the Accept property value, this object is considered found.

Frame   Defines which vision frame to apply to the positioning of this object.

PassType   Defines the acceptance condition of the object detection result.

For details on vision object properties, refer to *6. Vision Objects* of this manual and the *Vision Guide 7.0Properties and Results Reference Manual.*

Object Window: Results List

The results for the vision object are shown in the Results list that is located just below the Properties list. The Results list will display values in the Value field after either this vision object or the entire vision sequence is executed. Prior to execution the Results list of a specific vision object may not display anything or it may display the results from the last time the vision object was run.

It is sometimes useful to display the Object tab when running an entire vision sequence. This allows you to see the results for a particular object each time you run the sequence.

Vision Guide 7.0 allows you to run a vision sequence and then switch between vision objects to see the results for each individual vision object. You must select between the vision objects by using the vision object drop down list. If one of the vision objects is selected by clicking on the object in the image display, the object's results are cleared.

Vision Guide 7.0 keeps the results available even which switching between vision objects.

The results vary depending upon vision object type. Common vision object results are as follows:

Found   shows whether or not the vision object was found.

Passed   displays whether or not the object detection result was accepted.

Time   the amount of time it took to process this vision object.

For details on the teaching vision objects, refer to *6.Vision Objects*.

The <Run> button for the object in the Run Panel is used to run the currently selected vision object. This is useful for testing the current vision object by itself to fine-tune the property values for that object.

If a vision object is associated with other vision objects (e.g. a Polar object whose center position is defined by the results of a Correlation object), then the required vision objects will execute first, and then the current vision object will be executed. For example, assume that you want to run a Polar object when the Polar object's CenterPointObject property is defined by the results of a Correlation object. In this case, you can select the Polar object as the current object for use in the Object tab. Then, click the <Run Object> button and the Correlation object will execute first, followed immediately by the Polar object.

NOTE

☞

It is important to understand the difference between running a vision object and running a vision sequence. To run a vision object simply click on the <Run Object> button on the Object tab. Make sure that the desired vision object is selected in the flow chart or sequence tree before running. Running a vision object will execute only that vision object and any vision objects that may be required by that object. When executing a vision sequence, all vision objects within that sequence will execute.

4.6.3 The Calibration Window

The Calibration window can be displayed at the right side of the window by selecting the calibration in the sequence tree.
The Calibration window is used for setting a calibration and viewing calibration results.
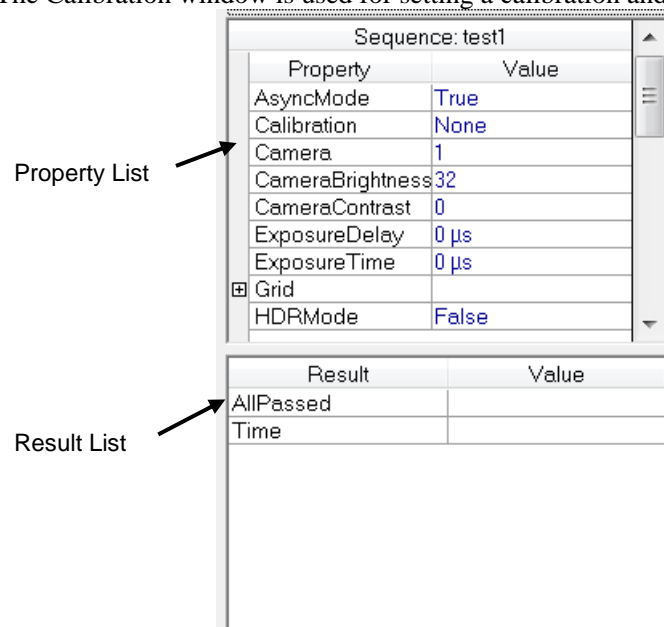
Calibration Window: Properties List

The properties for the vision Calibration are shown in the Properties list, which looks similar to a small spreadsheet. The name of the property is on the left side and the value is shown on the right. Properties are set by first clicking on the Value field for a specific property and then either entering a value or selecting from a list which is displayed.

For information on Calibration properties, refer to *4.9 Calibrations* or *the Vision Guide 7.0 Property and Results Reference Manua*l.

Calibration Window: Results list

The results for the Vision Calibration are shown in the Results list that is located just below the Properties list.

For details on camera calibration, refer to *4.9 Calibration*.

4.6.4 The Jog Tab

The Jog tab is used for jogging the robot during calibration or while viewing live video.
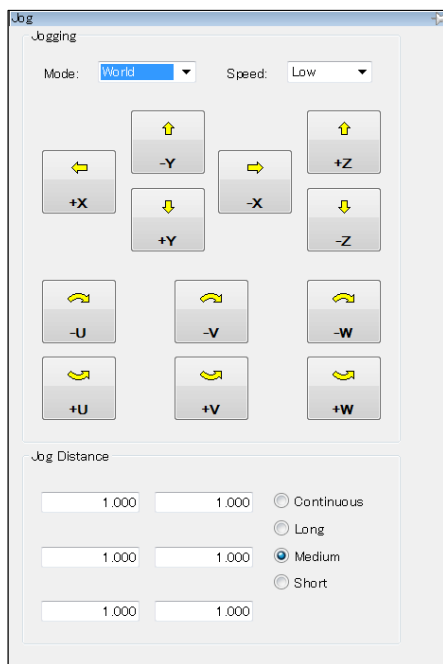
The Jog tab can be displayed by clicking the <Jog> button at the right side of the sequence and calibration trees. The Jog tab is also a fly-out panel that can be placed at arbitrary position.

Slecting the [Jog] tab starts the communication with the controller. If the communication failed, the display returns to the previously-selected tab.

Before jogging, the robot motors must be turned on. You can turn on the motors from the Robot Manager or the Command Window.

Jog Tab: Selecting the Jog Mode

Select the jog mode from the [Mode] box. The choices available are World, Tool, Local, Joint, and ECP (if ECP is enabled). You can select the current Tool, Local, and ECP from the Robot Manager.

Jog Tab: Selecting the Jog Speed

Select the jog speed from the [Speed] box.

Jog Tab: Selecting the Jog Distance

Select the jog distance by clicking on the option button in the [Jog Distance] group. When Long, Medium, and Short distances are selected, you can change the distance by typing in the jog distance text boxes.

Jog Tab: Jogging the Robot

After selecting the jog mode, jog speed, and jog distance, click on the jogging buttons to jog the robot. When the jog distance is set to Continuous, the robot will move until you release the button. For the other distances, the robot will move the amount specified in the jog distance. If a jog button is held down, the robot will continue to move.

4.6.5 The Robot Tab

The Robot tab can be displayed by clicking the <Robot> button below the Jog tab at the right side of the sequence and calibration trees. The Robot tab is also a fly-out panel that can be placed at arbitrary position.

The Robot tab is used for setting various functions that may be necessary for jogging the robot. You can do the following tasks:

> Select the current robot
>
> Turn motor power on and off
>
> Change power
>
> Free joints
>
> Execute Home or MCal
>
> Execute Reset
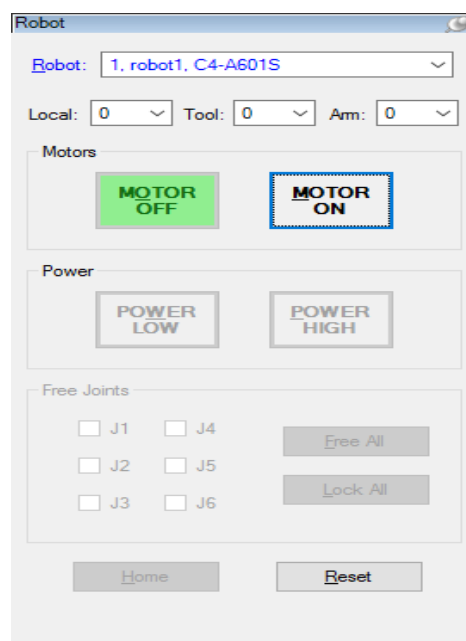
4.7 Vision Sequences and Objects

4.7.1 Overview of Vision Sequences

This chapter describes following items related to the basic knowledge and usage of vision sequences.

- Vision sequence definition and general information.

- The Sequence window.

- Creating and deleting vision sequences.

- Vision sequence properties and results.

- Running vision sequences from the Vision Guide window.

- Testing and Debugging vision sequences.

- How to run vision sequences from the SPEL$^+$ language.

- Image Acquisition.

4.7.2 Vision Sequence Definition & General Information

What is a vision sequence?

A vision sequence is a group of vision objects in a specific order that can be executed from the Vision Guide Development Window or from the SPEL$^+$ language.

Like vision objects, a vision sequence has specific properties that are used to set up execution. For example, the Camera property defines which camera will be used to acquire an image for this vision sequence and the RuntimeAcquire property defines how the image will be acquired for this vision sequence.

While vision objects can be run and tested individually, most vision applications require a series of vision objects to be executed sequentially to compute the final result. This is where vision sequences come in.

Vision sequences are used to run the vision objects in a specific order. After creating and testing individual vision objects, you need to test the vision sequence.

How does a vision sequence work?

Vision sequences contain one or more vision objects and the fundamental purpose of the vision sequence is to execute these vision objects in sequential order (as previously defined by the user.)

The following steps show what happens when a vision sequence executes:

1. The vision sequence uses its property settings to set up the execution of the sequence. These include things like which camera to use to acquire an image, determining the method to use to acquire an image (Strobed, Stationary, or No acquire at all).

2. The image is acquired and placed in the frame buffer for use by the vision objects that will execute next. (If the RuntimeAcquire property is set to "None", then the image that was already in the frame buffer is used for the sequence. This allows multiple sequences to operate on the same image.)

3. Vision objects are now executed in sequential order using the image acquired in step 2 above. Once all the vision objects have been executed, sequence and object results are available for display on the Sequence tab and Object tab and are also available from the SPEL$^+$ Language.

Vision sequences and EPSON RC+ 7.0 projects

EPSON RC+ 7.0 is based on the Project concept where each EPSON RC+ 7.0 Project contains programs, teach points, and all other robot configuration parameters required for a specific application.

When Vision Guide 7.0 is used within an EPSON RC+ 7.0 project, all vision sequences which were created within that project are also saved with the project.

This allows the use and modification of all vision sequences previously created in the project at the time the user opens a particular project next time.

4.7.3 Overview of Vision Objects

Vision objects make up the heart of Vision Guide 7.0 as they are the foundation upon which everything else is based.

This chapter provides the fundamental knowledge required to apply Vision Guide 7.0 towards solving your vision applications.

In this chapter the following items are covered:

- Definition of a vision object and how it works.

- Explanation of the different types of vision object shapes.

- How to position and size vision objects.

- Explanation of what each vision object is used for.

- How to use each vision object.

- Review of properties and results for each vision object.

- Explanation of technical details for each vision object.

- Discuss utilities that are used with vision objects such as: histograms, statistics, vision object graphics manipulation.

4.7.4 Vision Object Definition

What is a Vision Object?

A vision object is a vision tool which can be applied to an image.

At first glance many vision developers think vision objects are just another name for the algorithms that support Vision Guide 7.0. However, after using Vision Guide 7.0 you will find that the vision algorithms are only one part of what defines a vision object. Vision objects are like containers that hold the information required to use a specific vision algorithm. The unique manner with which this data can be easily manipulated and examined (through properties and results) is what makes vision objects so powerful and allows users to develop new vision applications so much more quickly than with other vision systems.

Some of the vision object Tools supported include: Correlation Search, Blob Analysis, Polar Search, Edge Detection, Line Creation, Points, and Frames. All vision objects can be executed (applied to the current Image) and all return results such as how much time the vision object took to execute, position information, angle information, and whether or not the vision object was even found. Refer to each individual vision object explanation later in this section for more details.

4.8 Vision Sequence Properties and Results

Vision sequences, like vision objects, have properties and results. The primary difference is that the properties and results for Vision Sequences apply to the entire Vision Sequence, while vision object property settings apply to only one vision object.

4.8.1 Vision Sequence Properties

Vision sequence properties are normally used for setting up the execution of the Vision Sequence and to set the proper values for the acquisition of an image for which all the vision objects within the vision sequence will then work from.

All vision sequence properties apply to the entire vision sequence. For example, the Calibration property is set at the vision sequence level rather than the vision object Level.

In this way, all vision objects within a vision sequence will use the Calibration specified by the Calibration property and all the results are guaranteed to be with respect to the same calibration.

The following list is a summary of properties for vision sequences. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
Calibration	Specifies the camera calibration to use for the vision sequence. This must be specified in order to return Robot or Camera coordinate results. Default: none
Camera	Specifies which camera to use for the vision sequence. Default: 1
CameraBrightness	Specifies the brightness value. Default: 128
CameraContrast	Specifies the contrast value. Default: 128
ExposureDelay	Specifies the delay before the exposure is started. (Unit : microsecond) Default: 0
ExposureTime	Specifies the exposure time when used in asynchronous reset mode. (Unit : microsecond) Default: 0
ImageBuffer	Specifies which buffer to store the grabbed image in. Default: 0
ImageColor	Specifies how color images should be acquired. Default: 1 - All
ImageFile	Specifies the name of the file that contains the image on disk that you want to work from. Default: None (no image file)
ImageFileScale	Specifies the scale factor for an image file. Default: 0 (fit to ImageSize)
ImageSize	Specifies the resolution of the grabbed image. Default: Maximum resolution of the current camera.
ImageSource	Specifies the source for the image. Default: 1 – Camera
Index	Displays the index of the sequence.
HDRMode	Displays the captured image as the HDR image. Default: False
Name	The name of the vision sequence.

Property	Description
RuntimeAcquire	Defines which method to use to acquire an image for use with the vision sequence. Default: 1 – Stationary
RuntimeFreeze	Tells the vision sequence to freeze the image for display. Default: True
SaveImage	Displays a dialog for saving the current image to disk.
ShowProcessing	Specifies whether image processing will be displayed. Default: True
StrobeBlackVideo	Specifies whether to clear the image to black after the sequence starts and before the image is grabbed after the trigger is received. Default: True
StrobeDelay	Specifies the delay before the strobe output is turned on (Unit : microsecond) Default: 0
StrobeTime	Specifies the time that the strobe output is on.(Unit : microsecond) Default: 0
TriggerMode	Specifies the electronic shutter trigger mode. Default: Leading Edge
GridColor	Specifies the name of grid's color displayed in the image display.
GridPitchX	Specifies the X pitch of the grids displayed in the image display.
GridPitchY	Specifies the Y pitch of the grids displayed in the image display.
GridShow	Specifies whether to display the grids in the image display. Default: False
GridType	Specify the type of the grids displayed in the image display. 1 - Cross Hair 2 - Rectangle
GridUnits	Specifies the unit of pitches for the grids displayed in the image display. 1 - Pixel 2 - MM When "2 – MM" is selected, the grids are not displayed if calibration is not performed.

4.8.2 Vision Sequence Results

Vision sequence results apply to the entire vision sequence.

They are useful because they tell you information about the vision sequence as a whole. For example, the AllPassed result returns whether or not all vision objects within the vision sequence were found or not.

The following list is a summary of vision sequence results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Result	Description
AllPassed	Returns whether or not all vision objects within the sequence were accepted.
Time	Returns the total execution time for the vision sequence.
	This includes the cumulative time for all vision objects as well as the time required to acquire an image.

4.9 Calibration

4.9.1 Overview

Calibration is an important part of the Vision Guide 7.0 system.

In order to locate parts in the robot world coordinate system, or to make physical measurements, you must calibrate the system.

Vision Guide 7.0 supports both mobile (mounted to robot) and fixed camera orientations.

4.9.2 Calibration Definition

Vision Guide 7.0 supports multiple calibrations for each project. Any calibration can be used with one or more vision sequences in the same project.

Each calibration includes the following data:

Calibration name	The name of the calibration that is referenced by vision sequences. (Up to 16 alphabets.)
Camera number	The number for the camera being calibrated.
Camera orientation	Camera mounting method
Calibration target sequence name	The name of the vision sequence that will be used to calibrate the camera. (This can be any sequence in the current project.)
Robot calibration speed and acceleration (when required)	The speed and acceleration used during calibration.
Robot Arm number and Tool number used during calibration	You can use an Arm or Tool during calibration. These must be defined prior to teaching points or calibrating.
Robot Points for reference and camera locations	These points are saved for each calibration.
Light outputs	These are optional standard outputs that can be used to control lights during calibration.
Image distortion correction	Specifies whether to execute image distortion (camera tilt and lens distortion) correction
Image distortion correction target sequence name	The name of the vision sequence used for image distorting correction. (Any sequence in the current project can be used)

Each calibration uses nine points that span the field of view of the camera being calibrated. By spanning the field of view, the system can do the best job of compensating for camera tilt and lens inaccuracies.

When performing a distance measurement with the whole field of view, perform image distortion correction. By performing image distortion correction, camera tilt (angle error between the work plane and the camera's optical axis) and lens distortion can be corrected.

When do you need to calibrate?

Calibration is required anytime you want to make a physical measurement or locate a position in robot world coordinates. Several results reported for vision objects require calibration. For example, the CameraX, CameraY, RobotX, RobotY, RobotU results require calibration.

4.9.3 Imaging Pattern for Image Distortion Correction

The following is a sample of the image used for image distortion correction.

To perform image distortion correction, the grids must have the same pitches in both horizontal and vertical lines, and more than 100 targets must be seen within the camera's field of view.
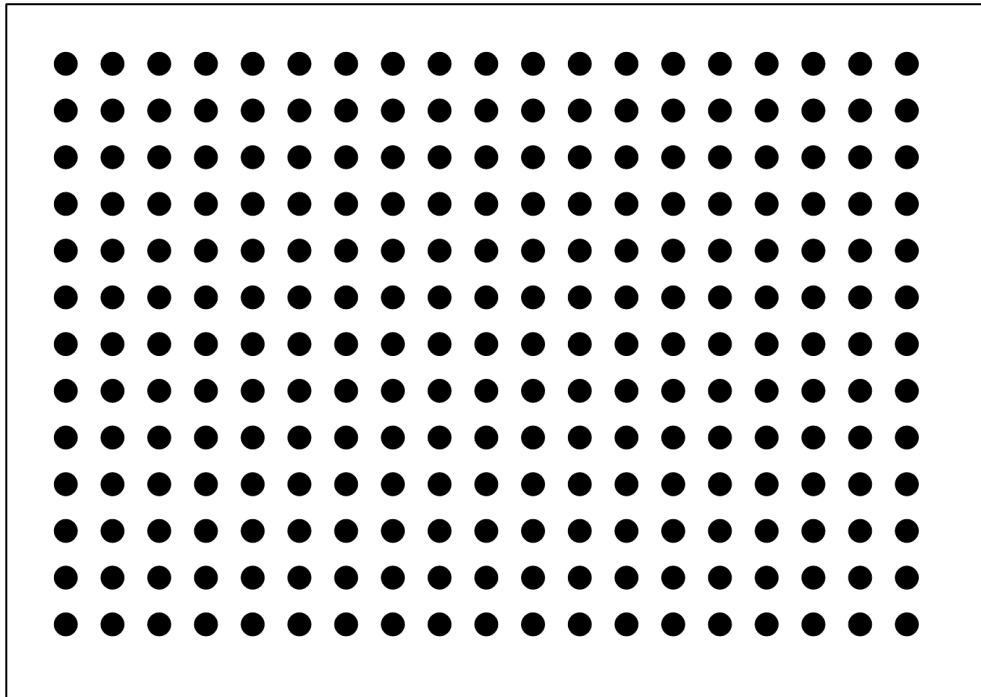
5. Vision Sequences

5.1 The Vision Sequence Window

The Sequence window is used to:

- Set properties for the vision sequence

- Examine the results for the entire vision sequence execution
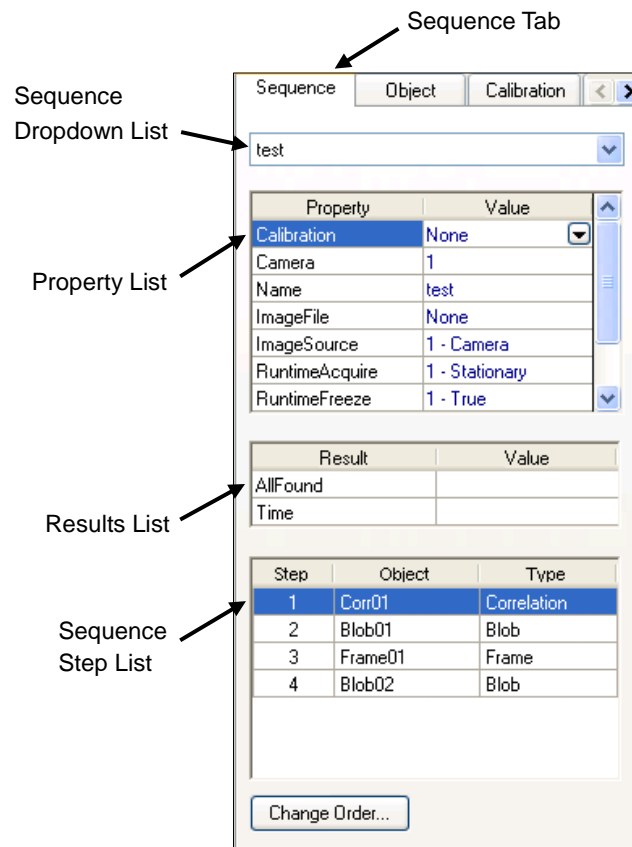
5.1.1 Selecting a Vision Sequence

There is a drop down list shown at the top of the Sequence tab. Clicking on the down arrow displays a list of all the vision sequences for this project. You can then click on any one of the vision sequences to select which one you want to work on.

5.1.2 Sequence Window Properties List

The properties for the vision sequence are shown in the Properties list that is similar to a small spreadsheet. The name of the property is on the left side and the value is shown on the right.

Vision properties are set by clicking on the Value field for a specific property and then either entering a value or selecting from a list which is displayed.

For details on the vision sequence properties, refer t*o 4.8.1 Vision Sequence Properties.*

5.1.3 Sequence Window Results List

The results for the vision sequence are shown in the Results list that is located just below the Properties list.

The Results list will only display values in the Value field after a vision sequence is executed. Prior to execution the result fields will not display anything.

For details on the vision sequence results, refer to *4.8.2 Vision Sequence Results.*

5.1.4 Sequence Flow Chart

Located at the right side of the image display is the sequence flow chart. This flow chart displays all the vision objects that have been defined for this vision sequence.

The order in the flow chart corresponds to the execution order of the vision objects.

Refer to *5.5 Change Order of Sequences* for details.

5.2 Creating a New Vision Sequence

5.2.1 Overview

Clicking the <New Sequence □ > toolbar button or selecting New Sequence from the Vision menu opens a dialog that requests a name to be given for the new sequence. The dialog can also be opened by right-clicking the sequence flow in the flow chart or sequence node in the sequence tree to select New Sequence. Type a name of up to 16 alphanumeric characters and click <OK>.

When creating a new sequence, you can copy an existing sequence by selecting it from the Copy from existing sequence dropdown list.

5.2.2 Shortcuts

Toolbar   : □

Keys   : None

5.2.3 Dialog Box Options

Option	Description
OK	Creates the new vision sequence with the associated name.
Cancel	Cancels the new vision sequence operation.
Help	Opens the help description for the New Sequence command.
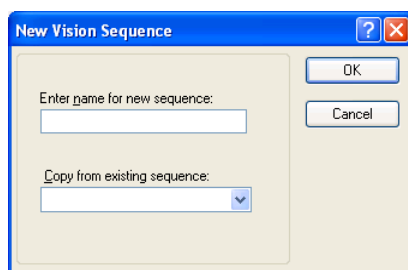
5.3 Deleting a Vision Sequence

5.3.1 Overview

Clicking on the <Delete Sequence ⊠> Vision Guide toolbar button or selecting the Delete Sequence item from the Vision menu opens a dialog with a list of all vision sequences that have been created for the current project. The dialog can also be opened by right-clicking the sequence flow in the flow chart or sequence node in the sequence tree to select Delete Sequence. Then, use the mouse or arrow keys to select which vision sequence to delete.

Once the correct vision sequence name is highlighted, click on the <Delete> button in the Delete Vision Sequence dialog. A confirmation message box will be displayed.

Note that once you delete a vision sequence and save the current project, the sequence cannot be recovered. If you accidently delete a sequence, you can execute [File]-[Restore] to restore the entire vision project to the state of the last save. If you think you may want to use a sequence later, be sure to keep it. Just because you keep it with a project doesn't mean you have to use it in your programs.

5.3.2 Shortcuts

Toolbar   : ⊠

Keys   : None

5.3.3 Dialog Box Options

Option	Description
Selection List	Used to select which sequence to delete.
Delete	Deletes the highlighted vision sequence.
Cancel	Cancels the delete vision sequence operation.
Help	Opens the help description for Delete Sequence Command.
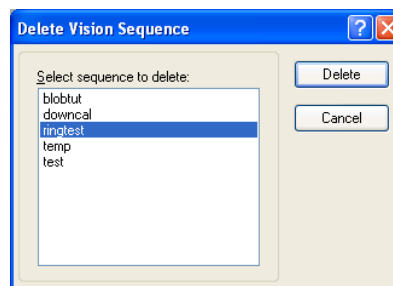
5.4 Deleting a Vision Object

To delete a vision object, first select the vision object you want to delete.

Once the vision object to delete has been selected, select [Delete] or <Delete> from the Vision menu, or click on the <Delete Object 🖳> toolbar button.

A confirmation dialog will appear. Click <Yes> to delete the object.

The <Delete Object> toolbar button is dimmed if there are no vision sequences for the current project or if there are no objects for the current vision sequence.

5.5 Change Order of Sequences

To change the order of sequence steps, right-click on the sequence flow in the flow chart or sequence node in the sequence tree and click on <Change Order> button. This opens the [Change Step Order] dialog. To change the step number for an object, select the object from the list, then click on the Move Up or Move Down buttons. Click <OK> to accept the changes.
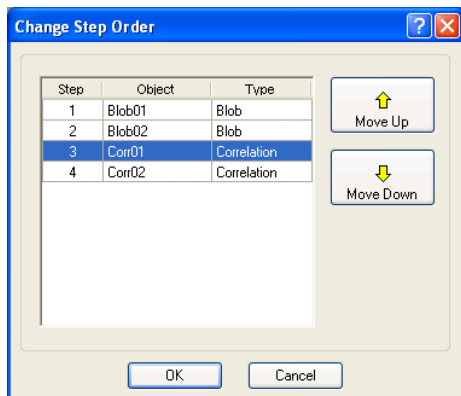
When modifying the execution order of the vision objects, pay close attention to vision objects that can be based on other objects, such as a Line object having a StartingPoint as "Corr01" and an EndPoint of "Blob01". For example, if the Line object step is moved above the "Corr01" object but remains below the "Blob01" object, then a warning will appear stating that the Line object's StartingPoint property will be modified to "Screen". There are many possible combinations of vision objects based on other vision objects so pay close attention to which objects require another object to execute properly. Vision Guide should always warn you when you modify the order of an object that was dependent upon another, but it's a good idea that you try to remember the dependencies as well.

5.6 Running Vision Sequences

5.6.1 Vision Sequence Selection

Before you can run a vision sequence, you need to select the vision sequence that you want to run from the sequence tree. This sequence tree makes it easy to change between vision sequences within a project.

5.6.2 Setting Vision Sequence Properties

After you have selected the vision sequence to run, you may need to set some parameters for the sequence. Setting vision sequence parameters is done from the Properties list on the Sequence window. Properties must be properly set up prior to running a vision sequence. For example, if you want to create and run a sequence using an image which will be acquired from Camera 2, then you must be sure and set the Camera property for that sequence to 2. One of the most common mistakes for new users is to run a vision sequence with the wrong Camera property setting.

The Camera property is used to select which camera to use for a particular vision sequence. The cameras will be automatically detected by the system at startup. If you power up a camera while Vision Guide 7.0 is displaying the video for that camera, you may not see video until you reselect the camera from the sequence Camera property.

5.6.3 Vision Sequence Results

Once you run a vision sequence, you will probably want to see the results for the sequence. The vision sequence results can be seen from the Results list located on the Sequence window.

5.6.4 Sequence Flow Chart

The step execution list for the sequence is displayed in the right side of the image display. This flow chart shows all the vision objects defined in the currently selected sequence.

The order in the flow chart corresponds to the execution order of the vision objects.

5.6.5 Running a Vision Sequence

On the left side of the Run Panel located at the bottom of the Vision Guide window, you can see the <Run> button for the sequence. Clicking this button causes the entire sequence to be executed.

After creating and testing the objects you want to use for a particular sequence, you need to test the entire vision sequence by clicking on the <Run> button on the left side in the Run Panel.

5.6.6 Running a Sequence Multiple Times (Cycles)

When using the <Run Sequence> button, pay close attention to the [Cycles] text box which can be used by right-clicking the ▼ button at the right side of the <Run> button in the Run Panel.

The number entered in this box specifies how many times to run the vision sequence. This is useful for testing the reliability of a vision sequence before writing any code to make it work with the robot.

By running a vision sequence several times and then using the Statistics feature of Vision Guide, you can see the Mean, Standard Deviation, Range, and Minimum and Maximum values for each vision object in the sequence. This helps give you a good perspective on how well your vision sequence is performing.

Any number larger than 1 will cause the [Cycles] text box color to change to yellow. This warns you that you are about to execute several cycles when you click the <Run Sequence> button.

5.6.7 The Image Display During Vision Sequence Execution

When a vision sequence is executed, the image display will show which vision objects were found and which were not found by a display of color.

The vision objects that were found will display a green outline of the search window and found position.

The vision objects that were not found will display a red outline of the search window. (Since they are not found there will be no found position displayed.)

Note that if the Graphics property for a specific vision object is set to None, no graphics will be displayed for that specific vision object. To ensure that all vision objects have their graphics displayed, click the <Force All Graphics On > Vision Guide 7.0 toolbar button. This will cause all graphics to be displayed regardless of the individual Graphics property settings.

5.6.8 Aborting Vision Sequence Cycles

Clicking the <Abort> button stops the sequence execution immediately.

5.7 Testing and Debugging a Vision Sequence

One of the first places to look to see if everything in a vision sequence is functioning properly or not is to look at the status of the vision sequence AllPassed property.

If any of the vision objects results was not accepted, the AllPassed vision sequence property will be set to False. On the other hand, the AllPassed reslt may return the result as "accepted" by mistake if any of the vision objects is not set correctly.

Therefore, eventually you need to test and debug your vision sequence because something isn't quite working properly.

The rest of this section describes some of the debugging features and techniques used with Vision Guide sequences.

5.7.1 Examine the Passed Color of the Vision Objects

One of the first things to always do when trying to find the source of a problem is to use the Pass/Fail colors which are placed on the vision objects on the image display.

Vision objects that are passed have their search windows and found position displayed in green (this can be changed by the PassColor property).

Vision objects that have failed display their search windows in red (this can be changed by the FailColor property). If any of the vision objects within your vision sequence display a red search window, this should be your first clue as to which vision object to more closely examine.

Note that if the Graphics property for a specific vision object is set to None, then no graphics will be displayed for that specific vision object. To ensure that all vision objects have their graphics displayed, click on the <Force All Graphics On > Vision Guide toolbar button. This will cause all graphics to be displayed regardless of the individual Graphics property settings.

5.7.2 Examining Individual Vision Object Results

When running a vision sequence, you can display the vision sequence results on the Sequence window or you can display an individual vision object's results on the Object window.

If you suspect a specific vision object's performance or just want to closely monitor it after the vision sequence runs, you can do so with just a few clicks.

(1) Select the vision object from the flow chart or sequence tree.

(2) Examine the results for that vision object once the vision sequence is run

You can even switch vision objects or between the Object window and Sequence window while a vision sequence is running. This gives you the ability to start a vision sequence running for 1 or more cycles and then examine the results for many of the different vision objects within that vision sequence without having to stop the vision sequence.

5.7.3 Single Stepping Through a Vision Sequence

One of the other nice debugging features for the Vision Guide 7.0 Development Environment is the Single Step feature.

Located on the Run Panel is the <Step> button. Clicking on this button causes the vision sequence to single step one vision object at a time.

When using the single step feature, you can see which vision object will execute next as the vision object name is highlighted in blue on the Sequence tab Step list.

You can also change properties for any of the vision objects prior to their execution during single stepping.

The first click of the <Step> button will cause all vision objects to go in an inactive state. They will become blue in color. The next click of the <Step> button will cause the first vision object in the Sequence to execute. It will be green in color if the object is found and red if it was not found. If the Sequence tab is currently displayed you can also see which vision object will execute next. It will be highlighted in blue.

5.7.4 Using the Statistics Feature

The Statistics feature is another debugging tool.

When testing a vision sequence for reliability, you may want to set the cycle count to a high number and then let the vision sequence run for a while.

Then check the statistical results for all the vision objects from the Statistics dialog. You can see which vision objects performed reliably and which had varying results.

For details on the vision statistics features, refer to *9. Using Vision Guide Statistics*.

5.7.5 Switching between the Run and Vision Guide Windows

There will be times when you have completed testing and think your system is OK to run from the Run window for final test and in fact something may still need adjustment. For these situations, we made the Vision Guide 7.0 Development Window accessible while running from the Run window.

Let's take a look at an example.

Assume you are running a Vision Guide 7.0 application from the Run window and you would like to see the Score result for the Correlation object called "pin1" because you have had problems with this object in the past.

While the system is running from the Run window, click on the <Vision toolbar button > from the EPSON RC+ 7.0 main toolbar. This will bring up the Vision Guide 7.0 Development Window while the application is still running.

Next, click on the Object tab to bring it forward. Then select the "Pin1" object from the drop down list box at the top of the Object tab. You will now see the results for the object called "Pin1" without actually stopping the program from running.

To return back to the Run window simply click on the Run window Toolbar button on the EPSON RC+ main toolbar.

5.8 Running Vision Sequences from SPEL⁺

One of the primary design goals for Vision Guide 7.0 was to make sure that whatever was created from the Point and Click Vision Guide 7.0 Development Environment was fully accessible from the SPEL+ language. That is, it is possible to use the Vision Guide 7.0 Development Environment as a prototyping environment and use it in the SPEL+ Language without rewriting.

Therefore, once you create a vision sequence from the Vision Guide 7.0 Development Environment, that vision sequence can be executed from VRun command in SPEL+.

The syntax for VRun is just "VRun seqname". Where "seqname" is the name of the vision sequence you want to run. In the code snippet shown below you will see 2 different VRun statements each initiating a different vision sequence.

"VRun findpin" initiates a sequence to find the center position of a pin for the robot to pick up the pin.

"VRun housing" is executed later in the program to find the position in the housing to place the pin.

```
VRun findpin
VGet findpin.pincntr.RobotXYU, found, X, Y, U
 .
 .
VRun housing
VGet housing.setpos.RobotXYU, found, X, Y, U
```

This is just a small example of how to use the VRun command. For mode details on how to use the SPEL⁺ language with Vision Guide Sequences and with the Robot, refer to *11. Using Vision Guide in SPEL+*.

5.9 Image Acquisition

5.9.1 When Are Images Acquired?

Many vision systems require that you specify a special command or step to acquire an image to do processing with. Vision Guide 7.0 helps remove this extra step since images are acquired at the start of a vision sequence.

Normally, you only need to specify the proper setting for the RuntimeAcquire property. For most applications you will never need to even set the RuntimeAcquire property because it defaults to Stationary. This means that an image will be acquired at the beginning of the vision sequence execution. There are other sequence properties that are used to configure how images are acquired.

5.9.2 Using the Same Image with Multiple Vision Sequences

If you want to use the same image for two or more vision sequences, all you have to do is to set the RuntimeAcquire property for the first vision sequence to Stationary and set the RuntimeAcquire property for the other vision sequences to None. Setting the other vision sequences RuntimeAcquire to None will prevent that sequence from acquiring another image so all processing will be done on the image acquired from the 1st vision sequence.

5.9.3 Using Image Buffers

There is one image buffer for each camera (buffer 0), and 10 image buffers that can be shared between sequences (buffers 1 – 10).

Set the ImageBuffer sequence property to specify which buffer to use. The default buffer is 0. You can use image buffers to grab and store several images in memory, then process them later.

For example, you can create a sequence with no objects that grabs images into multiple buffers, and then process the images in other sequences.

```
' seq1 has no objects - it is used to grab images
' into multiple buffers
VSet seq1.RuntimeAcquire, VISION_ACQUIRE_STATIONARY
VSet seq1.ImageBuffer, 1
VRun seq1 ' Grab an image into buffer 1
Go Image2Pos
VSet seq1.ImageBuffer, 2
VRun seq1 ' Grab an image into buffer 2
Go Image3Pos
VSet seq1.ImageBuffer, 3
VRun seq1 ' Grab an image into buffer 3
...
' Now process the previously grabbed images
VSet seq2.RuntimeAcquire, VISION_ACQUIRE_NONE
VSet seq2.ImageBuffer, 1
VRun seq2 ' Process the image in buffer 1
VGet seq2.AllPassed, allPassed
...
```

When using image buffers 1 - 10 for sequences that do not acquire an image (processing only), the Camera property value is ignored during VRun.

5.9.4 Using Strobed Image Acquisition

Vision Guide 7.0 supports a strobe input that allows the vision sequence to acquire an image and search after a strobe input has fired. To use a strobe input, follow these steps:

(1) Wire the trigger signal to the camera connector. If you are also using a strobe light, you can wire that to the strobe output signal of the camera.

For network cameras (Compact Vision), wire the trigger to the connector on the camera. The camera also provides an optional ouput for a strobe lamp.

(2) Set the RuntimeAcquire property for the sequence you want to strobe to Strobed.

(3) In your SPEL$^+$ program, execute VRun as usual, then wait for the AcquireState property to change to the value 3, which indicates that the strobe has fired. In this example, the trigger is being signaled from an external device.

```
Integer state
Boolean found

#define PICTURE_DONE 3
TmReset 0
VRun seq1
Do
  Wait 0.01
  VGet seq1.AcquireState, state
  If Tmr(0) > 10 Then
    Error ER_STROBE_OT  ' User defined overtime error
  EndIf
Loop Until state = PICTURE_DONE
VGet seq1.obj1.Found, found
```

If you do not wait for the image to be grabbed by checking AcquireState, then the next vision command in the task will automatically wait for the image to be acquired before executing. In this case, the image must be acquired before processing can continue, or you must abort the task. It is recommended that you check AcquireState, so that your program can continue if the trigger never fires, and the image is not acquired.

When you run a strobe sequence from the Vision Guide 7.0 GUI, the system will wait until the strobe has fired. A message will be displayed in the EPSON RC+ status bar indicating this after waiting a few seconds. You can abort by clicking the <Abort> button.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

5.9.5 Working with Color

If a color camera is configured, you can acquire color images or load color images from disk.

Some vision tools can process color images, and other tools only work with grayscale images, as shown in the table below.

Vision Tool	Color Processing	Grayscale Processing
Blob		×
Correlation		×
Geometric		×
Edge		×
Polar		×
Code Reader		×
OCR		×
ImageOp	×	×
ColorMatch	×	×
LineFinder		×
ArcFinder		×
DefectFinder		×

The ImageOp tool has a ColorFilter operation and a ColorStretch operation that process a color image. All other ImageOp operations use the grayscale image.

The ColorMatch tool is normally used to process color images. However, it can also be used with grayscale images, using colors that are different levels of gray.

When a color image is acquired, an internal grayscale image is also created for use with tools that require a grayscale image. When a sequence is run, the objects that perform color processing use the color image, and the objects that perform grayscale processing use the grayscale image.

A color image consists of three color bands: Red, Green and Blue. Use the ImageColor sequence property to select which color band(s) to acquire. The default setting is All, which means a full color image is acquired using all three bands.

You can also select Red, Green, Blue, or Grayscale. When Red, Green, or Blue is selected, then the grayscale image is derived from the color monochrome image. This allows you to search one color band with the grayscale processing tools.

If you click the <Run Object> button on the [Vision Guide] window and the current object is a grayscale processing tool, then the video image is shown in grayscale, just as the tool would see it during processing.

6. Vision Objects

6.1 Basic Points of Vision Objects

Before going into the details of vision objects, there are a few things you should know about vision object layouts and position/sizing methods used to manipulate them.

When a new vision object is created and placed in the image display it is given specific visual characteristics that are important to understand.

For example, a Correlation object is displayed with a search window, model window and model origin. All of which can be repositioned with simple dragging techniques.

The followings are some of the basic vision objects and the methods to manipulate them.
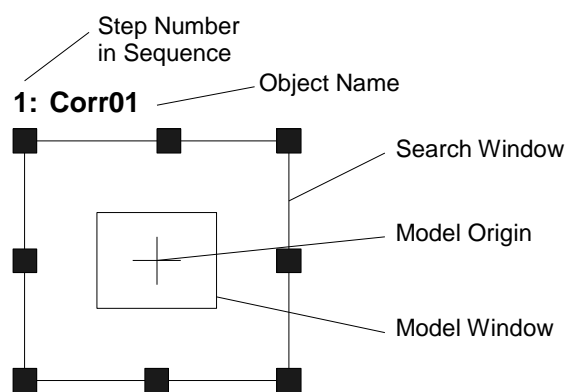
A Sample Vision Object Layout

6.1.1 The Search Window

Most vision objects have a search window as shown below. The search window is the outside box as shown in the figure above and is magenta (light purple) in color when active (the vision object you are currently working with) and blue when inactive (another vision object in the Sequence but not being worked with at the moment).
At run time, it changes color to green or red depending upon whether or not the object was found.

The search window is used to define the Region of Interest (or area within which we will search for a part or pattern).

Moving the search window is easily accomplished by clicking down on the name of the vision object (or on the search windows itself) and then dragging it to the desired new position on the image display. ("Corr01" is the name of the vision object in the figure below.)
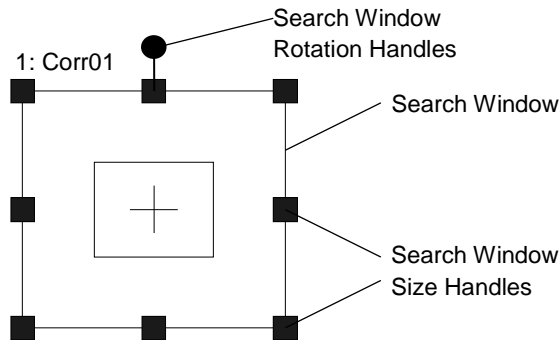
The size of the search window is very important when configuring a vision object. The larger the search window, the more time will be required to process the search. It is a good idea to make the search window large enough to handle the variation in part position while keeping it as small as possible. Sometimes the use of *6.2.10 Frame object* can help reduce the size of search windows.

Tips:
By setting the SearchWinType property to "RotatedRectangle" or "Circle", the model window can be angled or the circular model window can be used.

Positioning the Search Window

To position the search window, click on the name of the vision object, or the search window itself, and then drag the Vision object to whatever position you like.

When moving the vision object, make sure that the top left corner is located in the proper position.

Use the search window Size Handles (as described in the next section) to resize the search window.

Resizing the Search Window

The search window size handles appear when a vision object is first created or when a user selects a vision object by clicking on that object.

They are the small boxes located on the sides and corners of the search window. By using them, you can easily adjust the search window size.

When you move the mouse pointer over one of the search window size handles, it changes into a double-ended arrow pointer. Now the handle can be selected by clicking the mouse.

You can click on any of the search window size handles and then drag the side or corner to resize the search window.

Clicking on a search window size handle located on one of the sides of the search window allows you to resize the object horizontally.

Clicking on the top or bottom of the search window allows you to resize the object vertically.

Clicking on one of the corner search window size handles allows you to resize the search window both horizontally and vertically at the same time.
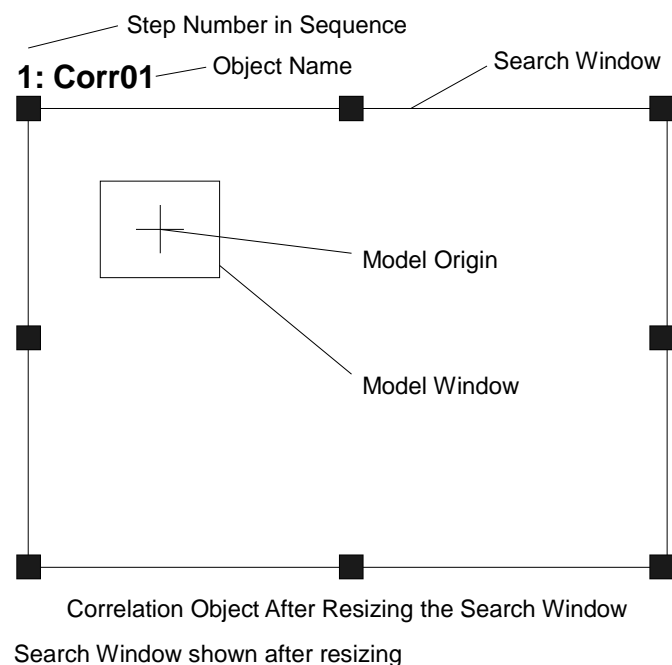
Correlation Object After Resizing the Search Window

Search Window shown after resizing

Tips:
By setting the SearchWinType property to "RotatedRectangle" or "Circle", the model window can be angled or the circular model window can be used.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

6.1.2 The Model Window

Correlation, Geometric, and Polar objects contain a model window and a search window. In this section, the Correlation object model window will be used.

Correlation objects have a model window as shown in the figure below. The model window is the inside box as shown in the figure and is magenta in color. (Except at run time where it changes to green when the object is found.) The model window is used for defining the boundaries of the model. (Remember that a model is an idealized representation of a feature. In general, it's what you will be searching for.)
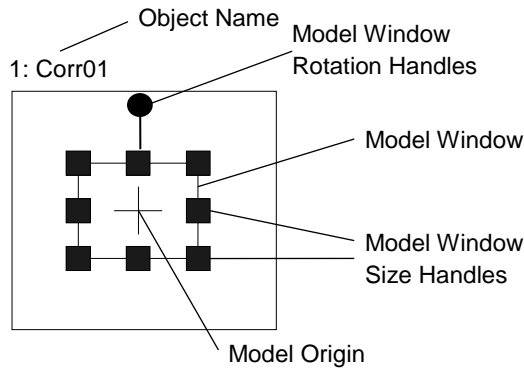
The Model Window

Positioning the Model Window

Since the size and position of the model is very important for teaching a new model, it is important to understand exactly how to set the position for and adjust the size of the model window.

To position the model window, click on one of the 4 lines which make up the model window and then drag the model window to a new position. We recommend positioning one of the four corners of the model window first and then resizing the model window.

Just as with the search window, the model window size is very important when configuring a vision object. In general, searching for a small model in a large search region takes more time than a large model in the same large search window. It is a good idea to make the model window big enough to define the part or pattern you are looking for while keeping it as small as possible but at the same time keeping the search window small as well.

Tips:
By setting the ModelWinType property to "RotatedRectangle" or "Circle", the model window can be angled or the circular model window can be used.

Resizing the Model Window

Then model window size handles appear when a model window is selected.

However, if the search window is selected then you will not see the model window size handles. You can easily locate the model window size handles because they are the small boxes that are located on the sides and corners of the model window when the search window is selected. If you cannot see the Handles on the model window, then the search window may be the active Window. To make the model window the active Window, click on one of the sides of the model window and you should then see the model window size handles.

Model window size handles are used to size the model window. Moving the mouse pointer over one of the model window size handles causes the pointer to change to a double-ended arrow pointer. This indicates that you can click the handle and resize the model window.

You can click on any of the model window size handles and then drag the side or corner to resize the search window.

Clicking on a model window size handle located on one of the sides of the model window allows you to resize the object horizontally.

Clicking on the top or bottom of the model window allows you to resize the object vertically.

Clicking on one of the corner model window size handles allows you to resize the model window both horizontally and vertically at the same time.

6.1.3 The Model Origin

Every model has a model origin. The model origin is the fixed reference point by which we describe the model's location in an image.

It should be noted that the model origin coordinate position is referenced to the upper left position of the model window.

The upper left position of the model window starts at [0,0].

1: Corr01

Model Origin

Model Window

[0,0] of Model Window
(upper left corner)
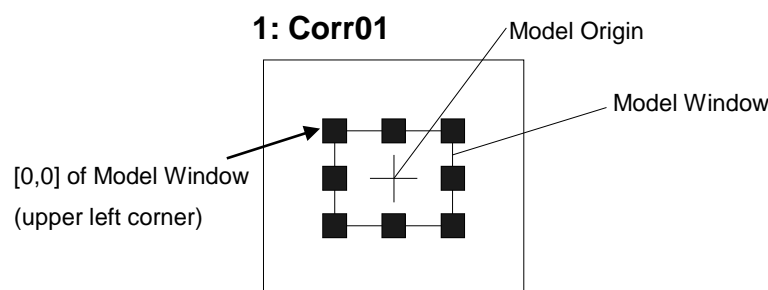
The Model Origin

Positioning the Model Origin

The model origin can be positioned automatically or manually.

To set the model origin automatically, set the ModelOrgAutoCenter property to True for that specific object. In this case, the model origin will automatically be set to the center of the model window.

NOTE

The default setting of the ModelOrgAutoCenter property is True.

To move the model origin manually, set the ModelOrgAutoCenter property to False. This can be done from the Object tab within the Properties list for the specific object. When setting the property in the Properties list, subpixels can be used for the setting value.

NOTE

Even when the subpixel is used, the model origin in the image is displayed in pixels.

To move the model origin, set the model window for the target object active. If the model window is not active, click on one of the lines which make up the model window and you will see the model window size handles appear for the model window.

If the ModelOrgAutoCenter property is set to 0 - False, the model origin can be moved by using the mouse. To move the model origin, click on the crosshair of the model origin and drag it to a new position.

NOTE

When the model origin is moved by using the mouse, it will be set in pixels. To set the model origin in subpixels, use the Properties list.

It is normally a good idea to position the model origin on a position of significance for the model you are working with. For example, you may want to place the model origin at the position on the model where you want the robot grip to pick up a part.

6.2 Using Vision Objects

This section describes the details of vision objects, such as various vision object layouts and difference in operating methods.

|	
⚠ **CAUTION**	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

6.2.1 ImageOp Object

ImageOp Object Description

ImageOp objects enable you to perform morphology (including open, close, minify, magnify), convolution (including sharpen, smooth), flip, binarize, and rotate for a specified region of interest.

Other vision objects that are placed in the ImageOp region of interest will perform their operations on the output of ImageOp. For example, you can execute a scale-down operation on the entire video image with an ImageOp tool, and then place Blob objects inside the ImageOp search window to search the minified image.

ImageOp Object Layout

The Image object has an image processing window, as shown below.

Step Number in Sequence

1: ImgOp01   Object Name

Processing Window

ImageOp Object Layout

ImageOp Object Properties

The following list is a summary of the ImageOp object properties with brief descriptions. For details on each property, refer to the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AngleObject	Defines which object to perform automatic rotation from. The image will be rotated counter-clockwise using the Angle result of the specified object. The Operation property must be set to Rotation, otherwise this property is ignored. Default: None
Caption	Used to assign a caption to the object. Default: Empty String
ColorMode	Sets which color space (RGB/HSV) to use for color operations. Default: RGB
CurrentModel	Runtime only. When Operation is set to ColorFilter, this specifies which model to use for the ModelColor property and also for VTeach. When CurrentModel = 0, the background color is selected. Default: 0
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color of an object when it is not accepted.
Frame	Specifies which positioning frame to use. Default: None
FillHoles	Specifies whether to fill the small holes in the binary image This property is displayed when Operation is set to Binarize. Default: False
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 - All
ImageBuffer1	Specifies the first image by the buffer number when the Operation property is set to SubtractAbs. SubtractAbs calculates the absolute value image of (first image – second image).
ImageBuffer2	Specifies the second image by the buffer number when the Operation property is set to SubtractAbs.
Iterations	Defines how many times to perform the specified operation. Default: 1

Property	Description
KeepRGBRatio	Specifies wheter to maintain the RGB ratio for the ColorStretch operation. Default: True
LabelBackColor	Sets the background color for the object's label. Default: Transparent
MaxRGB	Defines the maximum color for the ColorStretch operation. Default: 255, 255, 255
MinRGB	Defines the minimum color for the ColorStretch operation. Default: 0, 0, 0
ModelColor	Runtime only. When Operation is set to ColorFilter, this property is used at runtime to teach a model or the background color manually by setting the RGB color value directly. Default: RGB(0, 0, 0)
ModelColorTol	Runtime only. This property is used at runtime to set the color tolerance for a model color. If a pixel color is within the tolerance of a model color, then the pixel is unchanged. Default: 10 (ColorMode = RGB), 0,0,50 (ColorMode = HSV)
ModelName	Runtime only. This property is used at runtime to set the name of the current model.
Name	Used to assign a unique name to the ImageOp object. Default: ImgOpxx
NumberOfModels	Runtime only. This is the number of color models used. At runtime, you can set the NumberOfModels, then use CurrentModel and VTeach to teach each color model. Default: 1
Operation	Sets the type of image processing to perform. Default: Open
PassColor	Defines the color of the detected object when it is accepted. Default: LightGreen
RotationAngle	Defines how many degrees to rotate the image when the Operation property is set to Rotation. Default: 0
SearchWin	Runtime only. Sets / returns the search window left, top, height, width parameters in one call.
SearchWinHeight	Defines the height of the area to be searched in pixels. Default: 100

Property	Description
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinWidth	Defines the width of the area to be searched in pixels.
	Default: 100
ThresholdColor	Defines the color assigned to pixels within the thresholds.
	Default: Black
ThresholdHigh	Defines the upper threshold hold setting to use when the Operation property is set to Binarize.
	Default: 128
ThresholdLow	Defines the lower threshold hold setting to use when the Operation property is set to Binarize.
	Default: 0
ZoomFactor	Defines the zoom value to use when the Operation property is set to Zoom.
	Default: 1

ImageOp Object Results

The following list is a summary of the ImageOp object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Result	Description
Time	Returns the amount of time in milliseconds required to process the object.
FocusValue	Returns the relative focus level.
	The focus is optimum when the value becomes the minimum.
	This result is displayed only when Operation property is set to DetextFocus.

Using ImageOp Objects

Now we have set the foundation for understanding how to use Vision Guide ImageOp objects. This next section will describe the steps required to use ImageOp objects as listed below:

- How to create a new ImageOp object

- Position and size the search window

- Configure the properties associated with the ImageOp object

- Test the ImageOp object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence □ > toolbar button. You can also select a sequence which was created previously by clicking on the [Sequence] tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the [Sequence] tab. See *5.2 Creating a New Vision Sequence* for more details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New ImageOp Object

(1) Click on the <New ImageOp> toolbar button on the [Vision Guide] Toolbar. (In this case click means to click and release the left mouse button)

(2) Move the mouse over the image display. You will see the mouse pointer change to the ImageOp object icon.

(3) Continue to move the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "ImgOp01" because this is the first ImageOp object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

There is an ImageOp object similar to the one shown below:

Step Number in Sequence

Object Name

1: ImgOp01
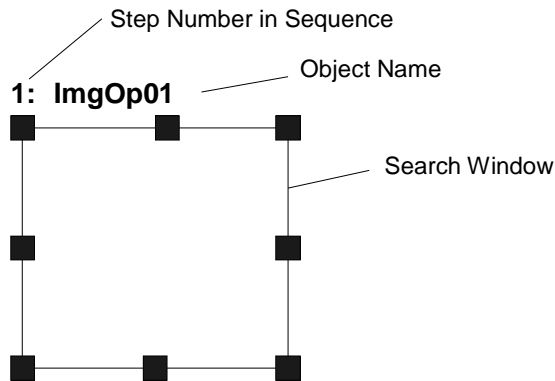
Search Window

New ImageOp Object Layout

(1) Click on the name label of the ImageOp object and while holding the mouse down, drag the ImageOp object to the position where you would like the top left position of the search window to reside.

(2) Resize the ImageOp object search window as required using the search window size handles. Move the mouse pointer over a size handle, then while holding down the left mouse button, drag the handle to resize the window.

Step 3: Configure the ImageOp Object Properties

Now, set property values for the ImageOp object. Some of the commonly used properties that are specific to the ImageOp object are described below. Explanations for other properties such as Graphics, etc. which are used on many of the different vision objects can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual* or in the ImageOp properties list.

Operation property - Determines which image operation to perform.
This is the most important property for the ImageOp object.

Iterations property - Determines the number of iterations to perform.

Polarity property - Detemine whether the operation will be performed on dark objects or light objects.
The default setting is dark objects. If you want to change it, click on the Value field of the Polarity property and you will see a drop down list with 2 choices: DarkOnLight or LightOnDark. Click on the choice you want to use.

You can test the ImageOp object and then come back to set any other properties as required later.

Step 4: Teach colors for the ImageOp object

When the Operation property is set to ColorFilter, then you will need to teach one or more colors to be filtered along with a background color. The <Teach> button on the Vision Guide window is enabled and a rectangular model window appears inside the ImageOp main window.

When the Teach button is clicked, the following modeless dialog is displayed:
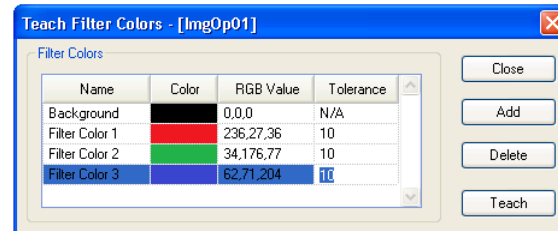
Clicking the <Add> button adds a new filter color with a default color of black. After selecting the background color or one of the filter colors, click the <Teach> button to use the average color of the pixels in the model window, or you can enter the RGB value of the color directly.

You can change the size and position of the model window while this dialog remains open. So you can add a color, size and position the model window, and then click Teach to teach the new color without closing the dialog.

You can change the ColorMode property while the Teach dialog is opened.

When ColorMode is RGB, then the Tolerance for each color has one value. When ColorMode is HSV, then the Tolerance for each color has three values (hTol, sTol, vTol).

The <Delete> button is used to delete filter colors. The background color cannot be deleted.

Step 5: Test the ImageOp Object

To run the ImageOp object, click on the <Run Object> button located at the bottom left of the [Object] tab. You will be able to see the effect of your ImageOp tool on the image.

Step 6: Make Adjustments to Properties and Test Again

After running the ImageOp object a few times, you may have encountered problems or just want to fine-tune some of the property settings. Some common fine tuning techniques are described below:

Fine-tuning of the ImageOp object may be required for some applications. The primary properties associated with fine-tuning of an ImageOp object are described below:

Iterations - Determines how many times to perform the desired image operation.

ThresholdColor, ThresholdLow, ThresholdHigh, ThresholdAuto
- These properties adjust parameters for the Binarize operation. Refer to the descriptions of these properties in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Once you have completed making adjustments to properties and have tested the ImageOp until you are satisfied with the results you are finished with creating this vision object and can go on to creating other vision objects or configuring and testing an entire vision sequence.

6.2.2 Geometric Object

Geometric Object Description

The Geometric object finds a model based on geometric features. It employs an algorithmic approach that finds models using edge-based geometric features instead of a pixel-to-pixel correlation. As such, the Geometric object offers several advantages over correlation pattern matching, including greater tolerance of lighting variations (including specular reflection) and models, as well as variations in scale and angle.

The Geometric object is normally used for determining the position and orientation of an object by locating features on the object. This is commonly used to find part positions to help guide the robot to pickup and placement positions.

Geometric Object Layout

The Geometric object has a search window and a model window, as shown below.
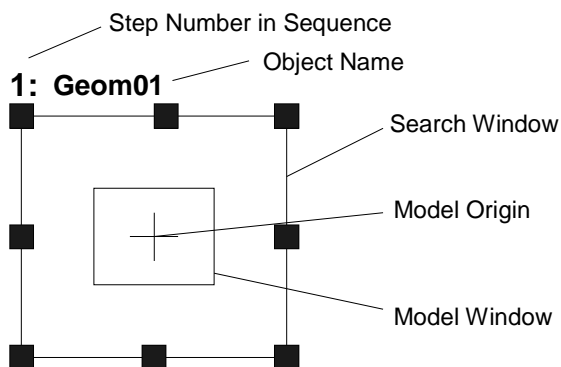
Geometric Object Layout

Geometric Object Properties

The following table lists general descriptions of Geometric object properties.

For details on each property, refer to the *Vision Guide 7.0 Properties and Results Reference Manual.*

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. If the value is small, it may result in false detection. Default: 700
AngleEnable	Specifies whether or not a Geometric search will do a search with angle (rotation). Specified prior to teaching a Model of the Geometric object.
AngleOffset	Specifies the offset value for rotation. Default: 0.000
AngleRange	Specifies the range within which to train a series of rotated models.
AngleStart	Specifies the center of the angle search.
Caption	Assigns a caption to the Geometric object.
CenterPointObject	Specifies the position to be used as the center point for the Geometric Search. This property is normally set to base the center of the Geometric Search on the XY position result from another vision object.
CenterPntObjResult	Specifies which result to use from the CenterPointObject. If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results. Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject.
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject.
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject. If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result. Default: False
Confusion	Indicates the amount of confusion expected in the image to be searched. This is the highest shape score a feature can get that is not an instance of the feature for which you are searching.

Property	Description
CurrentResult	Defines which result to display in the Results list (on the Object tab) or which result to return data for when the system is requested to find more than one of a like feature within a single search window.
DetailLevel	Selects the level at which an edge is considered found during the search.
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color of an object when it is not accepted.
Frame	Defines the current object searching position with respect to the specified frame. (Allows the Geometric object to be positioned with respect to a frame.)
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics mode to use to display the object.
LabelBackColor	Selects the background color for an object.
ModelObject	Determines which model to use for searching.
ModelOrgAutoCenter	A model has a fixed reference point by which a location of the model in the image is indicated. This point is referred to as the model origin. The ModelOrgAutoCenter property causes the model origin to be placed at the center of the model window.
ModelOrgFindCenter	Sets the model origin at the center of the registered model's edge.
ModelOrgX	Contains the X coordinate value of the model origin. (subpixels can be used)
ModelOrgY	Contains the Y coordinate value of the model origin. (subpixels can be used)
ModelWin	Runtime only. Sets / returns the model window left, top, height, width parameters in one call.
ModelWinAngle	Defines the angle of the model window.
ModelWinCenterX	Defines the X coordinate value of the center of the model window.
ModelWinCenterY	Defines the Y coordinate value of the center of the model window.
ModelWinHeight	Defines the height of the model window.
ModelWinLeft	Defines the left most position of the model window.
ModelWinTop	Defines the top most position of the model window.
ModelWinType	Defines the model window type.
ModelWinWidth	Defines the width of the model window.
Name	Used to assign a unique name to the Geometric object.

Property	Description
NumberToFind	Defines the number of objects to find in the current search window. (Geometric objects can find more than 1 object at once.)
PassColor	Selects the color for passed objects.
PassType	Selects the rule that determines if the object passed. Default: SomeFound.
RejectOnEdge	Determines whether the part will be rejected if found on the edge of the search window. Normally, this property should be set to True to avoid false detection caused by parts that are not completely within the search window. When set to False, you can use the FoundOnEdge result for each part result.
ScaleEnable	Enables scaling.
ScaleFactorMax	Sets / returns the maximum scale factor applied to the ScaleTarget value.
ScaleFactorMin	Sets / returns the minimum scale factor applied to the ScaleTarget value.
ScaleTarget	Sets / returns the expected scale of the model you are searching for.
ScaleTargetPriority	Sets / returns whether or not to detect objects near the ScaleTarge preferentially.
SearchReducedImage	Sets / returns whether to use a size reduced image when searching.
SearchPolarity	Sets / returns the search polarity.
SearchWin	Runtime only. Sets / returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched. (i.e. the search window height)
SearchWinLeft	Defines the left most position of the area to be searched. (i.e. the search window left side)
SearchWinTop	Defines the upper most position of the area to be searched. (i.e. the search window top side)
SearchWinType	Defines the type of the area to be searched (i.e. Rectangle, RotatedRectangle, Circle).
SearchWinWidth	Defines the width of the area to be searched. (i.e. the search window width)
SeparationAngle	Sets / returns the minimum angle allowed between found objects.

Property	Description
SeparationMinX	Sets / returns the minimum distance along the X axis allowed between found objects.
SeparationMinY	Sets / returns the minimum distance along the Y axis allowed between found objects.
SeparationScale	Sets / returns the minimum scale difference allowed between found objects.
SharedEdges	Sets / returns whether to allow edges to be shared between found objects.
ShowModel	Displays a previously taught model at various zoom settings. Can be used to change the model origin and don't care pixels.
SkewFitEnable	Specifies whether to adopt skew on the model. Default: False
Smoothness	Sets / returns the smoothness level for the geometric edge extraction filter.
Sort	Selects the sort order used for the results of an object.
Timeout	Sets / returns the maximum search time for a Geometric object.

Geometric Object Results

The following table lists general descriptions of Geometric object results.

For details on each result, refer to the *Vision Guide 7.0 Properties and Results Reference Manual*.

Results	Description
Angle	Returns the amount of rotation associated with a part that was found. (i.e. This defines the amount of rotation a part may have in relation to the originally taught orientation.
CameraX	Returns the X coordinate position of the found part's position (referenced by model origin) in the camera coordinate system.
CameraY	Returns the Y coordinate position of the found part's position (referenced by model origin) in the camera coordinate system.
CameraXYU	(Available from SPEL language only) Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system.
Found	Returns whether or not the object was found. (i.e. whether or not the feature or part have a shape score which is above the Accept property's current setting.)
FoundOnEdge	Returns TRUE when the Geometric object is found too close to the edge of the search window. When FoundOnEdge is TRUE the Found result is set to FALSE.

Results	Description
NumberFound	Returns the number of objects found within the search window. (This number can be anywhere from 0 up to the number of objects you requested the Geometric object to find with the NumberToFind property.)
Overlapped	Returns True when the detected objects overlap.
PixelX	Returns the X coordinate position of the found part's position (referenced by model origin) in pixels.
PixelY	Returns the Y coordinate position of the found part's position (referenced by model origin) in pixels.
PixelXYU	Runtime only. Returns the PixelX, PixelY, and PixelU coordinates of the found part's position in pixels.
Reversed	Returns True when the detected object has reverse polarity to the model.
RobotX	Returns the X coordinate position of the found part's position (referenced by model origin) with respect to the Robot's Coordinate System.
RobotY	Returns the Y coordinate position of the found part's position (referenced by model origin) with respect to the Robot's Coordinate System.
RobotU	Returns the U coordinate position of the found part's position with respect to the Robot's Coordinate System.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the found part's position with respect to the Robot's World Coordinate System.
Scale	Returns the scale factor. ScaleEnabled must be set to True for this result to be valid.
Score	Returns an integer value which represents the level at which the feature found at runtime matches the model for which Geometric is searching.
ShowAllResults	Displays a dialog which allows you to see all results for a specified vision object in a table form. This makes it easy to compare results.
SkewDirection	Returns the direction of skew on the object detected during execution.
SkewRatio	Returns the skew ratio of the object detected during execution.
Time	Returns the amount of time in milliseconds required to process the object.
TimedOut	Returns whether or not the object execution terminates due to the time-out.

Understanding Geometric Search

The purpose of the Geometric object is to locate a previously trained model in a search window. Geometric objects can be used to find parts, detect presence and absence of parts or features, detect flaws, and a wide variety of other things.

This section will explain the basics of the Geometric objects. This section will cover the following topics:

- Geometric object models and Features
- Basic Searching Concepts
- Setting the Accept and Confusion property
- Additional Search Parameters
- Using the Multiple Results Dialog to Debug Searching Problems
- Geometric objects and Rotation
- Geometric objects and Scale
- Model Training for Angle Searching
- Searching Repeatability and Accuracy
- Calibrating the Camera to Subject Distance

Geometric Object Models and Features

It is important to understand the difference between features and models when using Geometric objects. A feature is any specific pattern of edges in a search window. A feature can be anything from a simple edge a few pixels in area to a complex pattern of tens of thousands of pixels in area. The Geometric operation measures the extent to which a feature in a search window matches a previously taught Model of that feature. A feature is something within an actual search window as opposed to a model or template that is an idealized representation of a feature.

It is common to train a representative model from one search window to be used to search for similar features in that search window. The picture below shows a search window containing a feature of interest (a cross). To train a model of the cross, you define the model window and click the <Teach> button on the Object tab. (For details on teaching models, refer to the section *Using Geometric Objects*.)

As a result, the model is created as shown on the right side of the picture and can be used to search for other crosses within the search window.
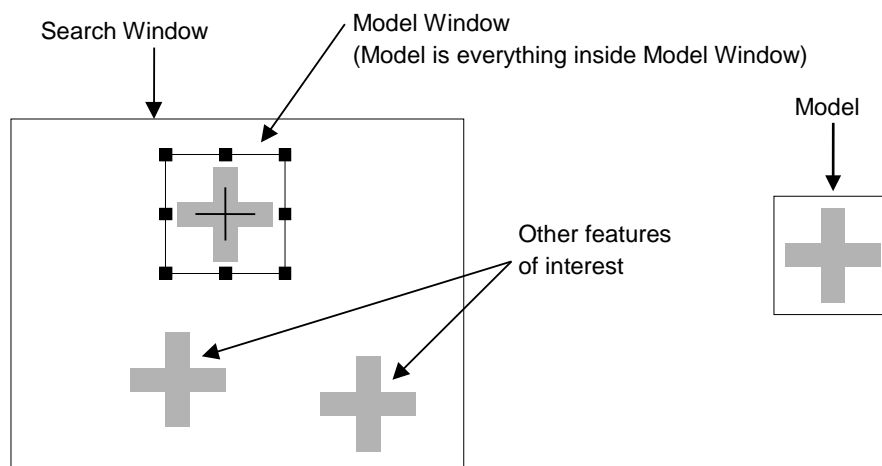
Search window containing several features of interest (left) and model trained from image (right)

While finding models based on geometric features is a robust and reliable technique, there are a few pitfalls that you should be aware of when allocating your models so that you choose the best possible model.

Make sure your images have enough contrast

Contrast is necessary for identifying edges in your model source and target image with sub-pixel accuracy. Images with weak contrast should be avoided since the Geometric search tool uses a geometric edge-based searching algorithm; the weaker the contrast, the less the amount and accuracy of edge-based information with which to perform a search. For this reason, it is recommended that you avoid models that contain slow gradations in grayscale values. You should maintain a difference in grayscale values of at least 10 between the background and the edges in your images.

Avoid poor geometric models

Poor geometric models suffer from a lack of clearly defined geometric characteristics, or from geometric characteristics that do not distinguish themselves sufficiently from other image features. These models can produce unreliable results.

Poor geometric model

Simple curves lack distinguishing features and can produce false matches.
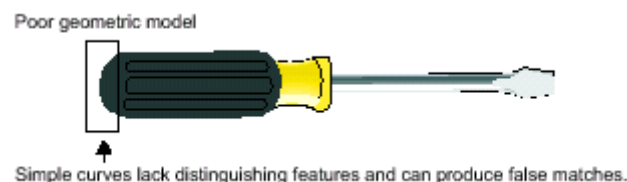
Avoid ambiguous models

Certain types of geometric models provide ambiguous results in terms of position, angle, or scale.

Models that are ambiguous in position are usually composed of one or more sets of parallel lines only. For example, models consisting of only parallel lines should be avoided since it is impossible to establish an accurate position for them. An infinite number of matches can be found since the actual number of line segments in any particular line is theoretically limitless. Line segments should always contain some distinguishing contours that make them distinct from other image details.

Model ambiguous in position

Models consisting of sets of parallel lines, without any distinguishing features, should be avoided.
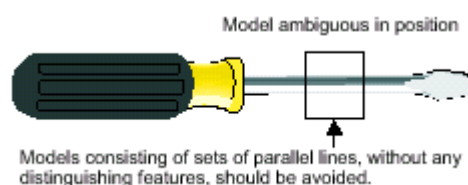
Models that consist of small portions of objects should be tested to verify that they are not ambiguous in scale. For example, models that consist of isolated corners are ambiguous in terms of scale.

This is an ambiguous candidate for searching through a range of scale. The lack of distinguishing geometric characteristics produces superfluous results.
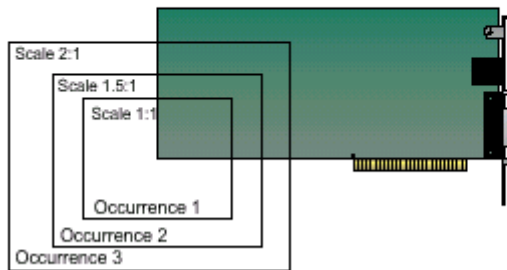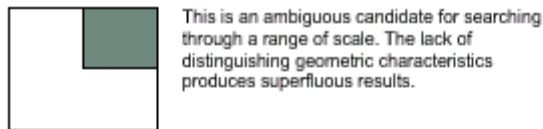
Symmetric models are often ambiguous in angle due to their similarity in features. For example, circles are completely ambiguous in terms of angle. Other simple symmetric models such as squares and triangles are ambiguous for certain angles.
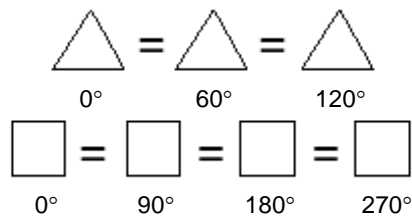
Nearly ambiguous models

When the major part of a model contains ambiguous features, false matches can occur because the percentage of the occurrence's edges involved in the ambiguous features is great enough to be considered a match. To avoid this, make sure that your models have enough distinct features to distinguish the model from other target image features. This will ensure that only correct matches are returned as occurrences. For example, the model below can produce false matches since the greater proportion of active edges in the model is composed of parallel straight lines rather than distinguishing curves.
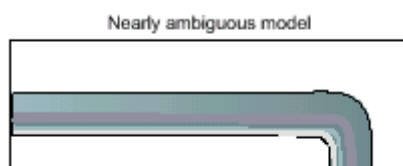
Nearly ambiguous model

Basic Searching Concepts

Searching locates features by finding the area of the search window to which the model is the most similar. The figure below shows a model and a search window, and the areas within the search window that are most similar to the model. A model similar to that shown below might be used to search for a feature such as a fiducial mark on a printed circuit board. A robot could then use the position data returned by the search function to find the location of the board for placing components or for positioning the board itself.
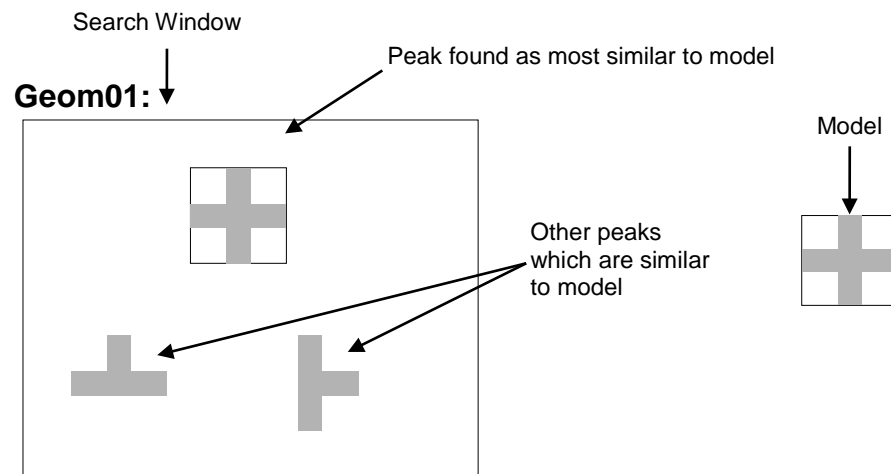
Setting the Accept and Confusion Property Thresholds

The Accept and Confusion properties are the main search parameters for Geometric objects. The Accept property influences searching speed by providing a hint as to when to pursue the search in a given region of the scene. When the Accept property is set high, features must be very similar to the model, so many regions can be ruled out by a cursory examination and not pursued further. If the Accept property is set to a low value, features that are only slightly similar to the model may exceed the Accept property threshold, so that a detailed examination of more regions in the scene is needed. Thus increasing the Accept property tends to increase speed. (i.e. higher Accept property values can make Geometric objects run faster.)

The Confusion property interacts with the number of results expected to influence searching speed. Together, the Confusion property and the number of results expected allow the system to quit the search before exploring all possible regions of the image.

Set the Accept property so that it will allow the system to find features that are examples of the "worst case degradation" you are willing to accept. The degradation may be caused by defects, scale, rotation or video noise. For the Accept property Vision Guide 7.0 sets the default value to 700. This is usually a good starting point for many applications. However, experimentation and correction will help you home in on the best value for your situation. Note that you do not always have to get perfect or nearly perfect scores for an application to function well. Shape scores of 200 may provid good positional information for some applications, depending on distortion of a feature.

Set the Confusion property based on the highest value you expect the "wrong thing" to get (plus a margin for error). The Confusion property threshold should be greater than or equal to the Accept property threshold. Setting the Confusion property to a high value will increase the time of the search, but may be necessary to insure that the right features are found. The Confusion property default value is 800 but should be adjusted depending upon the specific application requirements.

The figure below shows a scene where there is little confusion: the round pad is not very similar to the fiducial (cross). The Confusion property can therefore be set to a fairly low value (around 700). The Accept property is normally set less than or equal to the Confusion property, depending upon the amount of degradation you are willing to accept. Assuming this scene has little degradation, a shape score of 920 could be expected.
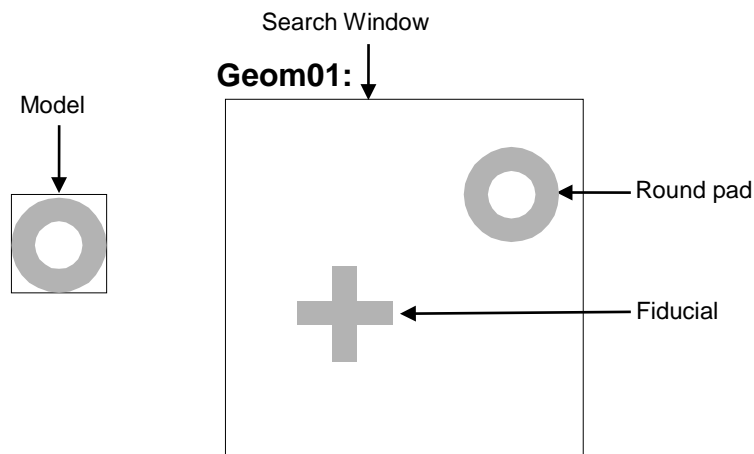
A scene with little confusion

The figure below shows a scene where there is a lot of confusion; both the feedthrough and the IC pad are similar to the round pad. The Confusion property should therefore be set to a fairly high value (around 820).
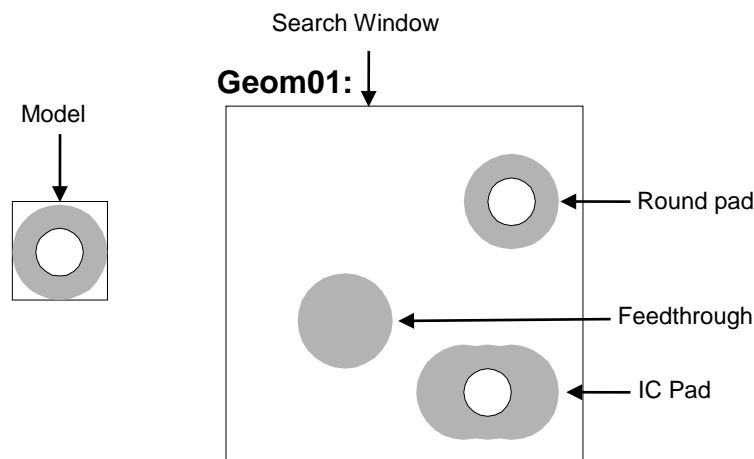
A scene with a high degree of confusion

A search window that has a region of constant gray value will always get a 0 Geometric value in that region. If a scene has a primarily uniform background (e.g. a white piece of paper), so that at most places there will be not Geometric, you can set the Confusion property to a low value because if the Geometric object finds anything, it must be the feature for which you are searching.

The Accept and Confusion properties can be thought of as hints that you provide to the system to enable it to locate features more quickly. In general, these properties should be set conservatively, but need not be set precisely. The most conservative settings are a low Accept property and High Confusion property. Use very conservative settings when you know very little about a scene in which you are searching; the search will be careful but slower. (This is especially important when using Geometric property positional results to give to the robot to move to.)

Use more liberal settings when you know a lot about a scene in which you are searching. For example, if you know that you are looking for one feature, and the rest of the scene is blank, a careful search is unnecessary; use more liberal settings and the search will be faster.

Additional Geometric search parameters

DetailLevel

DetailLevel determines what is considered an edge during the search.

Edges are defined by the transition in grayscale value between adjacent pixels. The default setting of Medium offers a robust detection of active edges from images with contrast variation, noise, and non-uniform illumination. In some cases where objects of interest have a very low contrast compared to high contrast areas in the image, some low contrast edges can be missed. If your images contain low-contrast objects, a DetailLevel setting of High should be used to ensure the detection of all important edges in the image. The VeryHigh setting performs an exhaustive edge extraction, including very low contrast edges. However, it should be noted that this mode is very sensitive to noise.

ScaleTargetPriority

Specifies whether or not to detect objects near the ScaleTarget preferentially. Set this property to True when detecting objects with small differences in size such as factory products.

Default: True

SearchReducedImage

Reduces the size of the input image when performing rough object detection.

This property may reduce search time when the input image has numerous features (e.g. edges).

Default: False

Smoothness	The Smoothness property allows you to control the smoothing level of the edge extraction filter. The smoothing operation evens out rough edges and removes noise. The range of this control varies from 0 (no smooth) to 100 (a very strong smooth).
	Default: 50
SharedEdges	You can choose to allow multiple results to share edges, by setting SharedEdges to True. Otherwise, edges that can be part of more than one result are considered part of the result with the greatest score.
TimeOut	In time critical applications, you can set a time limit in milliseconds for the Geometric object to find occurrences of the specified model. If the required number of occurrences is not found before the time limit is up, the search will stop. Results are still returned for those occurrences found. However, it is not possible to predict which occurrences will be found before the time limit isearched.
	Default: 2000 milliseconds

Using Multiple Results Dialog to Debug Searching Problems

Sometimes the parts that you are working with vary considerably (even within the same production lot) and sometimes there are 2 or more features on a part which are similar. This can make it very difficult to determine a good Accept property value. Just when you think you have set the Accept property to a good value, another part will come in which fools the system. In these cases it can be very difficult to see what is going on.

The [Show All Results] dialog was created to help solve these and other problems. While you may only be interested in 1 feature on a part, requesting multiple results can help you see why a secondary feature is sometimes being returned by Vision Guide 7.0 as the primary feature you are interested in. This generally happens a few different ways:

1. When 2 or more features within the search window are very similar and as such have very close Score results.

2. When the Confusion or Accept properties are not set high enough which allow other features with lower scores than the feature you are interested in to meet the Accept property setting.

Both of the situations above can be quite confusing for the beginning Vision Guide 7.0 user when searching for a single feature within a search window.

If you have a situation where sometimes the feature you are searching for is found and sometimes another feature is found instead, use the Show All Results dialog to home in on the problem. Follow the following steps to get a better view of what is happening:

(1) Set your NumberToFind property to 3 or more.

(2) Run the vision object from the Vision Guide 7.0 Development Environment.

(3) Click on the <ShowAllResults> property button to bring up the [Show All Results] dialog.

(4) Examine the scores of the top 3 or more features which were found.

Once you examine the scores of the top 3 or more features which were found as described above, it should become clear to you what is happening. In most cases you will see one of these two situations.

1. Each of the features that were found has a score greater than the Accept property setting.
    In this case, simply adjust your Confusion property value up higher to force the best feature to always be found rather that allowing other features to be returned because they meet the Accept threshold. You may also want to adjust the Accept property setting.

2. Each of the features is very close in score.
    In this case, then you will need to do something to differentiate between the feature which you are primarily interested in such as:

    - Readjust the search window so that the features which are randomly returning as the found feature are not contained inside.

    - Teach the Model again for the feature which you are most interested in.

    - Adjust the lighting for your application so that the feature which you are most interested in gets a much higher score than the other features which are currently fooling the system.

For details on using multiple results, refer to the section *6.2.16 Working with Multiple Results from a Single Object*.

Geometric Objects and Separation

You can specify the minimum amount of separation from other occurrences (of the same model) necessary for an occurrence to be considered distinct (a match). In essence, this determines what amount of overlap by the same model occurrence is possible.

You can set the minimum separation for four criteria, which are: the X position, Y position, angle, and scale. For an occurrence to be considered distinct from another, only one of the minimum separation conditions needs to be met. For example, if the minimum separation in terms of angle is met, then the occurrence is considered distinct, regardless of the separation in position or scale. However, each of these separation criteria can be disabled so that it is not considered when determining a valid occurrence.

The minimum positional separation properties SeparationMinX and SeparationMinY determine how far apart the found positions of two occurrences of the same model must be. This separation is specified as a percentage of the model size at the nominal scale (ScaleTarget).

The default value is 10%. A value of 0% disables the property. For example, if your model is 100 pixels wide at the ScaleTarget, setting SeparationMinX to 10% would require that occurrences be at least 10 pixels apart in the X direction to be considered distinct and separate occurrences.

The minimum angular separation (SeparationAngle) determines the minimum difference in angle between occurrences. This value is specified as an absolute angle value. The default value is 10.0 degrees. A value of 0 degrees disables the property.

The minimum scale separation (SeparationScale) determines the minimum difference in scale between occurrences, as a scale factor. The default value is 1.1. A value of 1.0 disables the property.

Geometric Objects and Scale

The scale of the model establishes the size of the model that you expect to find in the target image. If the expected occurrence is smaller or larger than that of the model, you can set the scale according to the supported scale factors. The expected scale is set using the ScaleTarget property. (range 0.5 - 2.0).

By default, searching through a scale range is disabled. If necessary, you can also enable a search through a range of scales by setting ScaleEnable to True. This allows you to find models in the target image through a range of different sizes from the specified ScaleTarget, either smaller or larger. To specify the range of scales, use ScaleMinFactor and ScaleMaxFactor. The ScaleMinFactor (0.5-1.0) and ScaleMaxFactor (1.0-2.0) together determine the scale range from the nominal ScaleTarget.

These maximum and minimum factors are applied to the ScaleTarget setting as follows:

```
max scale = ScaleTarget x ScaleMaxFactor

min scale = ScaleTarget x ScaleMinFactor
```

Note that the range is defined as factors so that if you change the expected scale (ScaleTarget), you do not have to modify the range. A search through a range of scales is performed in parallel, meaning that the actual scale of an occurrence has no bearing on which occurrence will be found first. However, the greater the range of scale, the slower the search becomes.

Geometric Objects and Rotation

Geometric objects are ideal for finding rotated parts. Since a geometric pattern of edges is being searched for, rotated parts can generally be found much more reliably than with other vision tools.

To use the search with angle capabilities of the Geometric object, the Model for the Geometric object must be taught with the AngleEnable property set to True. The AngleStart and AngleRange properties must also be set.

Model Training for Angle Searching

To Search with angle measurement, you must first configure the Geometric object for angle search. This is done by setting the AngleEnable property to True and using the AngleRange property to specify the range of angles over which models will be taught. You can also change the center of the angle search by setting the AngleStart property. This is the angle that the AngleRange is based on. For example, if AngleStart is 45 and AngleRange is 10, then the search will occur for models from 35 to 55 degrees.

Keep in mind that when training models with angle, the search window must be large enough to allow the model to be rotated without any part of the model going outside of the search window.

Searching Repeatability and Accuracy

Searching repeatability and accuracy is affected by the size and details of the model (shape, coarseness of features, and symmetry of the model), and the degradation of the features as seen in the search window (noise, defects, and rotation and scale effects).

To measure the effect of noise on position, you can perform a search in a specific search window that contains a non-degraded feature, and then perform the exact same search again (acquiring a second image into the frame buffer) without changing the position of the object, and then comparing the measured positions. This can be easily done by the following steps:

(1)   Click on the <Run Object> button of the Object tab two or more times

(2)   Click on the <Statistics> toolbar button

(3)   The Statistics dialog can then be used to see the difference in position between the two object searches.

For a large model (30x30) on a non-degraded feature, the reported position can be repeatable to 1/20 of a pixel. However, in most cases it is more realistic to achieve results of just below a pixel. (1/2, 1/3, or 1/4 pixel)

Searching accuracy can be measured by performing a search in a specific search window that contains a non-degraded feature, moving the object an exact distance and then comparing the reported position difference with the actual difference. If you have a large model (30x30 or greater), no degradation, no rotation or scale errors, and have sufficient edges in both X and Y directions, searching can be accurate to 1/4 pixel. (Keep in mind that this searching accuracy is for the vision system only and does not take into effect the inaccuracies which are inherent with all robots. So if you try to move the part with the robot you must also consider the inaccuracies of the robot mechanism itself.)

Calibrating the Camera to Subject Distance

For optimal searching results, the size of the features in an image should be the same at search time as it was when the model was taught. Assuming the same camera and lens are used, if the camera to subject distance changes between the time the model is trained and the time the search is performed, the features in the search window will have a different apparent size. That is, if the camera is closer to the features they will appear larger; if the camera is farther away they will appear smaller.

If the camera to subject distance changes, you should retrain the Model.

Using Geometric Objects

We have reviewed how normalized Geometric and searching works and set the foundation for understanding how to use Vision Guide Geometric objects. This section will describe the steps required to use Geometric objects as listed below:

- Create a new Geometric object

- Position and size the search window

- Position and size the model window

- Position the model origin

- Configure properties associated with the Geometric object

- Teach the Model

- Test the Geometric object and examine the results

- Make adjustments to properties and test again

- Working with Multiple Results from a Single Geometric object

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence ▢ > toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. See vision sequences for more details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New Geometric Object

(1) Click on the <New Geometric [icon] > toolbar button on the Vision Guide Toolbar. (In this case click means to click and release the left mouse button).

(2) You will see a Geometric icon appear above the <New Geometric> toolbar button.

(3) Click on the Geometric icon and drag to the image display of the Vision Guide window.

(4) A name for the object is automatically created. In the example, it is called "Geom01" because this is the first Geometric object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a Geometric object similar to the one shown below:
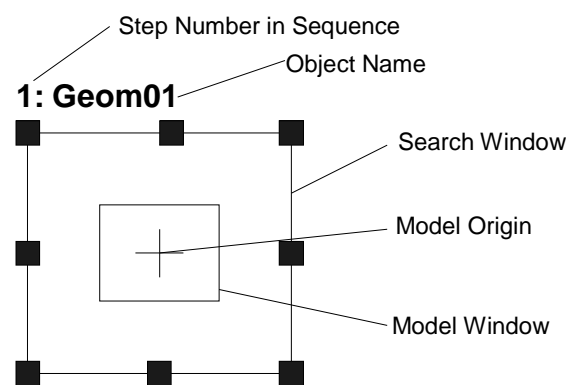
New Geometric Object Layout

(1) Click on the name label of the Geometric object and while holding the mouse down, drag the Geometric object to the position where you would like the top left position of the search window

(2) Resize the Geometric object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.) (The search window is the area within which we will search.)

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Step 3: Position and Size the Model Window

(1) The search window for the Geometric object you want to work on should be highlighted where you can see the size handles on the search window. If you cannot see the size handles click on the name field of the Geometric object. If you can see the size handles disregard this item and move on to the next item below.

(2) Click on one of the lines of the box that forms the model window. This will cause the model window to be highlighted. (You should see the size handles on the model window now.)

(3) Click on one of the lines of the box that forms the model window and while holding the mouse down, drag the model window to the position where you would like the top left position of the model window to reside.

(4) Resize the model window as required using the model window size handles. (This means click on a size handle and drag the mouse.) (The model window should now be outlining the feature that you want to teach as the model for this Geometric object.)

Geometric object layout should now look like the example shown below, where the search window covers the area to be searched and the model window outlines the feature you want to search for. Although the actual search window and model window may be different, take this as an idea of what was expected so far.
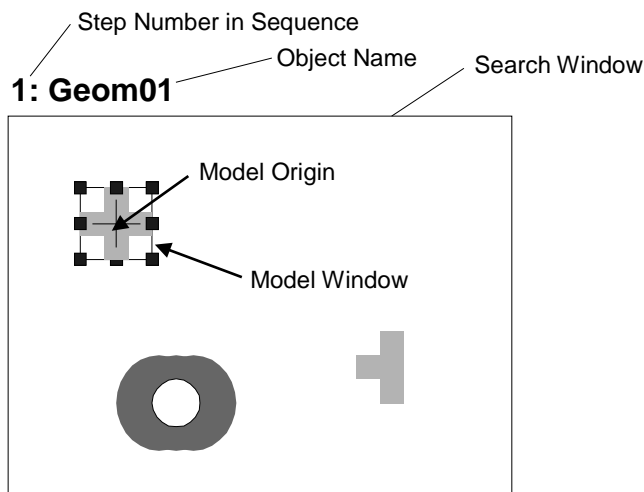
Geometric object after search and model window positioning and resizing

Tips for Setting Proper Size and Position of the Model Window:

The size and position of the model window is very important since it defines the feature to be searched for. When creating a model window for a Geometric object, there are 2 primary items you must pay attention to: (1) If you expect that your part will rotate much, smaller models are better because they require smaller search windows to search within. (2) Execution time can be reduced by making the model window as close in size to the search window as possible.

When two objects are positioned right next to each other and almost are touching, make the model window just a bit larger than the actual feature. This makes it easy to distinguish the object from others.

Note that the best model window size varies from application to application.

Step 4: Position the Model Origin

The model origin defines the position on the model that will be returned back as the position of the feature when you run the Geometric object. That is, the model origin should be placed in a place of significance if the position data is important.

For example, when using a Geometric object to find parts for a robot to pick up or place, it is important that the position of the model origin is in a location where the robot can easily grip the part because that is the position the robot will move to based on the RobotXYU, RobotX or RobotY result.

When a new Geometric object is created the ModelOrgAutoCenter property is set to True (default value). This means that the model origin is set to the center of the model window automatically and cannot be moved manually. However if you want to move the model origin manually, you must first set the ModelOrgAutoCenter property to False. The steps to do this and also actually position the model origin are shown below.

(1) Click the Object tab on the right top of the Vision Guide window. The Object Tab will be displayed in front. Find the ModelOrgAutoCenter property on the properties list and click in the value field.

(2) You will see a drop down list with 2 choices: True and False. Click on the False choice. Now the ModelOrgAutoCenter property is set to False and the model origin can be moved with the mouse.

(3) Click on the model window to highlight the model window.

(4) Click on the model origin and drag the model origin to a new position by keeping the mouse button held down. The model origin can only be positioned within the bounds of the model window.

Step 5: Configure the Geometric object properties

Using the Object tab displayed in front in Step 4, set the Geometric object property. To set properties, click on the associated property's value field and enter a new value or click on one of the items if a drop down list is displayed.

The following list shows some of the more commonly used properties for the Geometric object. Descriptions for other properties, such as AbortSeqOnFail and Graphics which are used on many of the different vision objects, are in Geometric properties. When testing the Geometric object, it is not necessary to set these properties. However, if you are working with Geometric objects for the first time, this section could be a good reference.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Name property	The default name given to a newly created Geometric object is "Geomxx" where xx is a number which is used to distinguish between multiple Geometric objects within the same vision sequence.
	If this is the first Geometric object for this vision sequence, the default name will be "Geom01".
	To change the name, click on the Value field of the Name property, type a new name and hit return. Once the name property is changed, everywhere the Geometric object's name is displayed is updated to reflect the new name.
Accept property	The Accept property sets the shape score that a feature must meet or beat to be considered Found.
	The value returned to the Score result is compared against this Accept property Value.
	Default: 700.
Confusion property	If there are many features within the search window which look similar, the Confusion property is useful to help "home in" on the exact feature you want to find.
	Default: 800
ModelOrgAutoCenter property	If you want to change the position of the model origin you must first set the ModelOrgAutoCenter property to False.
	Default: True
Frame property	The Frame property allows you to selects a previously defined Frame object as a reference frame for the Geometric object.
	The details for Frames are described in *Frame object*.
NumberToFind property	You can set the NumberToFind property larger than 1 depending on the number of features you want to find. This will allow the Geometric object to find multiple features within one search window.
AngleEnable property	Set this property to True if you want to use a Geometric model to search with angle. To search multiple models with angle, set the property to True before teaching the model for the Geometric object.
AngleRange property	Used with the AngleEnable property, the property searches the Geometric model with angle.
RejectOnEdge property	By using this property, you can exclude the parts touching the boundary of the search window. Normally, this property should be set to True.

It is possible to leave the properties as default and go on to the next step. The properties can be set later as necessary.

Step 6: Teach the model for the Geometric object

The Geometric object needs a model to search for and this is accomplished a process called teaching the model. You should have already positioned the model window for the Geometric object to outline the feature which you want to use as a model. Teaching the model is accomplished by following steps:

(1) Make sure that the Geometric object is the currently displayed object in the Object tab. You can check the current object drop down list at the top of the Object tab to see which object is the object you are currently working on. Also, you can check the image display to see which object is highlighted in magenta.

(2) Click the <Teach> button at the bottom right of the Object tab. It will take a few seconds for the Model to be taught in most cases. However, if you are teaching a model when the AngleEnable property is set to True, it takes more time to teach the model since the system is teaching many models each with a slight angle offset.

Step 7: Test the Geometric Object/Examine the Results

To run the Geometric object, click on the <Run Object> button located at the bottom left of the Object tab.

Results for the Geometric object will be displayed. The primary results to examine at this time are:

Found result	Returns whether or not the Geometric object was found. If the feature you are searching for is found, this result returns as True. If the feature is not found, the Found result returns False and is highlighted in red. If the feature was not found, refer to Step 8 for the more common reasons why a Geometric object is not found.
FoundOnEdge result	This result will return True if the feature was found where a part of the feature is touching the boundary of the search window. In this case the Found result will return False.
Score result	This shows how much the feature that most closely to the model matches to the model. The Score result ranges from 0 to 1000 with 1000 being the best match possible. Examine the Score result after running a Geometric object as this is your primary measure of how well the feature was found.
Time result	The amount of time it took for the Geometric object to execute. Remember that small search windows and small Models help speed up the search time.
NumberFound result	When searching for more than one Geometric object, the NumberFound result returns the number of features which matched the Geometric object's Model.

Angle result	The angle at which the Geometric is oriented.
	This is computed based on the original angle of the model. However, this value may sometimes be coarse in value and not so reliable. (We strongly recommend using the Polar object for finding Angles. Especially for robot guidance.)
PixelX, PixelY results	The XY position (in pixels) of the feature.
	Remember that this is the position of the model origin with respect to the found feature. If you want to return a different position, you must first reposition the model origin and then re-teach the Model.
CameraX, CameraY results	These are the XY position of the found feature in the Camera's Coordinate system.
	The CameraX and CameraY results will only return a value if the camera has been calibrated. If it has not, then [No Cal] will be returned.
RobotX, RobotY results	These are the XY position of the found feature in the Robot's Coordinate system.
	The robot can be told to move to this XY position. (No other transformation or other steps are required.)
	This value is the position of the model origin with respect to the found feature. If you want to return a different position, you must first reposition the model origin and then re-teach the Model. The RobotX and RobotY results will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.
RobotU result	This is the angle returned for the found feature translated into the Robot's Coordinate system.
	The RobotU result will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.
ShowAllResults	If you are working with multiple results, you may want to click on the button in the ShowAllResults value field. This will bring up a dialog to allow you to examine all the results for the current vision object.

NOTE
☞

The RobotX, RobotY, RobotU, RobotXYU and CameraX, CameraY, CameraU, CameraXYU results will return "no cal" since a calibration is not done in the above example steps. This means it is impossible for the vision system to calculate the coordinate results with respect to the Robot coordinate system or Camera coordinate system since the calibration is not executed. Refer to *Calibration* for details.

Step 8: Make Adjustments to properties and test again

After running the Geometric object a few times, you may encounter problems with finding a Geometric or just want to fine tune some of the property settings. Some common problems and fine tuning techniques are described in the next section *Geometric Object Problems.*

Geometric Object Problems

If the Geometric object returns a Found result of False:

- Try to change the Accept property to a lower value (for example, below the current score result) and run the Geometric object again.

- Check the FoundOnEdge result if it has a return value of True. If it is True, the feature was found but it was found where a part of the feature is touching the search window. This causes the Found result to be returned as False. To correct this situation, make the search window larger or if this is impossible, try changing the position of the camera, or resizing the model window.

If the Geometric object finds the wrong feature:

- Check the Accept property if it is set high enough. If it is set rather low this could allow another feature to be found in place of the feature you are interested in.

- Was the Confusion property set high enough? Is it higher than the Accept property? The Confusion property should normally be set to a value equal to or higher than the Accept property. This is a good starting point. But if there are features within the search window which are similar to the feature you are interested in, then the Confusion property must be moved to a higher value to make sure that your feature is found instead of one of the others.

- Adjust the search window so that it more closely isolates the feature which you are interested in.

Geometric Object Fine Tuning

Fine tuning of the Geometric object is normally required to get the object working just right. Following is a description for the primary properties associated with fine tuning of a Geometric object and model addition:

Accept property   When you set the Accept property lower, the Geometric object can run faster. However, lower Accept property values can also cause features to be found which are not what you want to find. Find an appropriate value through several trial runs in order to acquire reliable feature at the best execution speed.

Confusion property   If there are multiple features within the search window which look similar, you need to set the Confusion property relatively high. This will guarantee that the feature you are interested in is found rather than one of the confusing features. However, higher confusion costs execution speed. If you don't have multiple features within the search window which look similar, you can set the Confusion property lower to help reduce execution time.

Add another sample   "Add another sample" can be selected when teaching is performed with the model window whose size is same as the current model's model window. When the model slightly changes (shape or pattern is slightly different, shadow appears differently, etc.), selecting the changed model to add a new model may stabilize the score at object execution. The original model will be kept if the angle is largely out of position, or the model cannot be added due to significant difference.

Once you have completed adjusting and have tested the Geometric object until you are satisfied with the results, you are finished with creating this vision object. Go on to creating other vision objects or configuring and testing an entire vision sequence.

Other utilities for Geometric Objects

At this point you may want to consider examining the Histogram feature of Vision Guide 7.0. Histograms are useful because they graphically represent the distribution of grayscale values within the search window. The details regarding Vision Guide Histogram usage are to examine the Geometric object's results statistically.

6.2.3 Correlation Object

Correlation Object Description

The Correlation object is the most commonly used tool in Vision Guide 7.0. Once a Correlation object is trained it can find and measure the quality of previously trained features very quickly and reliably. The Correlation object is normally used for the following types of applications:

Alignment   For determining the position and orientation of a known object by locating features (e.g. registration marks) on the object. This is commonly used to find part positions to help guide the robot to pickup and placement positions.

Gauging   Finding features on a part such as diameters, lengths, angles and other critical dimensions for part inspection.

Inspection   Look for simple flaws such as missing parts or illegible printing.

Correlation Object Layout

The Correlation object has a search window and a model window, as shown below.
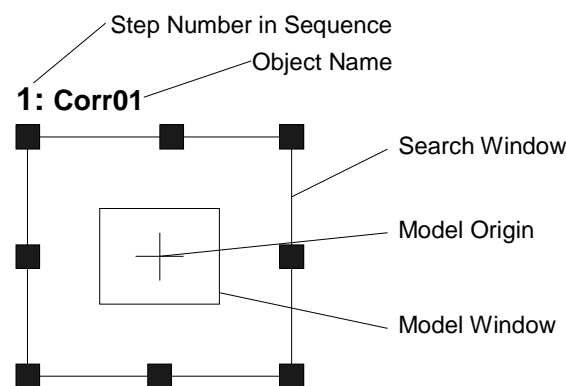
Correlation Object Layout

Correlation Object Properties

The following list is a summary of properties for the Correlation object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. If the value is small, it may result in false detection. Default: 700

Property	Description
AngleAccuracy	Specifies the desired accuracy for angle search in degrees. Default: 1
AngleEnable	Specifies whether or not a correlation search will do a search with angle (rotation). Specified prior to teaching a Model of the Correlation object. Default: 0 - False
AngleMaxIncrement	Maximum angle increment amount for teaching a correlation model for searching with angle. The maximum value is 10. Default: 10
AngleOffset	Specifies the offset value for rotation. Default: 0.000
AngleRange	Specifies the range within which to train a series of rotated models. The maximum value is 45. Default: 10
AngleStart	Specifies the center of the angle search. Default: 0
Caption	Used to assign a caption to the Correlation object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the Correlation Search. This property is normally set to base the center of the Correlation Search on the XY position result from another vision object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. If All is specified, the Frame object will ve applied to all (NumberFound) of the specified vision object results Default: 1
CenterPntOffsetX	Specifies the X offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntOffsetY	Specifies the Y offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject. If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result. Default: False

Property	Description
Confusion	Indicates the amount of confusion expected in the image to be searched. This is the highest shape score a feature can get that is not an instance of the feature for which you are searching Default: 800
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one result. Default: 1
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color for an object when it is failed. Default: Red
Frame	Defines the current object searching position with respect to the specified frame. (Allows the Correlation object to be positioned with respect to a frame.) Default: None
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies a graphic to be displayed. Default: 1 - All
LabelBackColor	Sets the background color for the object's label. Default: Transparent
ModelObject	Determines which model to use for searching. Default: Self
ModelOrgAutoCenter	A model has a fixed reference point by which we describe its location in a model window. This point is referred to as the model's Origin. The ModelOrgAutoCenter property causes the model origin to be placed at the center of the model window. Default: 1 - True
ModelOrgX	Contains the X coordinate value of the model origin. (subpixels can be used)
ModelOrgY	Contains the Y coordinate value of the model origin. (subpixels can be used)
ModelWin	Runtime only. Sets / returns the model window left, top, height, width parameters in one call.
ModelWinLeft	Defines the left most position of the model window.
ModelWinHeight	Defines the height of the model window. Default: 50
ModelWinTop	Defines the top most position of the model window.
ModelWinWidth	Defines the width of the model window.

Property	Description
Name	Assigns a unique name to the Correlation object. Default: Corr01
NumberToFind	Defines the number of objects to find in the current search window. Default: 1
PassColor	Selects the color for passed objects. Default: LightGreen
PassType	Selects the rule that determines if the object passed. Default:SomeFound
RejectOnEdge	Determines whether the part will be rejected if found on the edge of the search window. Normally, set True for the property. This avoids false detection caused by parts that are not completely within the search window. Default: 0 – False
SearchWin	Runtime only. Sets / returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched in pixels. Default: 100
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinWidth	Defines the width of the area to be searched in pixels. Defalut: 100
ShowModel	Displays a previously taught model at various zoom settings. Can be used to change the model origin and don't care pixels.
SkewFitEnable	Specifies whether to adopt skew on the model. Default: False
Sort	Selects the sort order used for the results of an object. Default: 0 – None
Timeout	Sets / returns the maximum search time for a Correlation object.

Correlation Object Results

The following list is a summary of the Correlation object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual.*

Results	Description
Angle	Returns the amount of part rotation in degrees.
CameraX	Returns the X coordinate position of the found part's position (referenced by model origin) in the camera coordinate system. Values are in millimeters.
CameraY	Returns the Y coordinate position of the found part's position (referenced by model origin) in the camera coordinate system. Values are in millimeters.
CameraXYU	Accessible from the SPEL+ Language only. Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system. Values are in millimeters.
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
FoundOnEdge	Returns TRUE when the Correlation object is found too close to the edge of the search window. If RejectOnEdge is TRUE, then the Found result is set to FALSE.
NumberFound	Returns the number of objects found within the search window. (This number can be anywhere from 0 up to the number of objects you requested the Correlation object to find with the NumberToFind property.)
PixelX	Returns the X coordinate position of the found part's position (referenced by model origin) in pixels.
PixelY	Returns the Y coordinate position of the found part's position (referenced by model origin) in pixels.
RobotX	Returns the X coordinate position of the found part's position (referenced by model origin) with respect to the Robot's Coordinate System.
RobotY	Returns the Y coordinate position of the found part's position (referenced by model origin) with respect to the Robot's Coordinate System.
RobotU	Returns the U coordinate position of the found part's position with respect to the Robot's Coordinate System.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the found part's position with respect to the Robot's World Coordinate System.
Scale	Returns the scaling value of the object detected during execution.

Results	Description
Score	Returns an integer value between 0-1000 that represents the level at which the feature found at runtime matches the model for which Correlation is searching.
ShowAllResults	Displays a dialog showing all results of the specified vision object in a table form. This makes it easy to compare results.
SkewDirection	Returns the direction of skew on the object detected during execution.
SkewRatio	Returns the skew rate of the object detected during execution.
Time	Returns the amount of time in milliseconds required to process the object.

Understanding Normalized Correlation

The purpose of Correlation is to locate and measure the quality of one or more previously trained features in a search window. Correlation objects can be used to find parts, detect presence and absence of parts or features, and detect flaws and a wide variety of other things.

While Vision Guide 7.0 has a wide variety of vision object tools, the Correlation object is the most commonly used tool due to its speed and overall reliability.

For example, in many applications Edge Tools can be used to find the edges of an object. However, if there are many potential edges within the same area that may confuse an Edge object, then a Correlation object may be able to be used instead to find the position of the edge.

There are also instances where Correlation objects are used over Blob objects (when a model can be taught) because they are more reliable.

Over the next few pages we will explain the basics behind the search tools as they apply to the Correlation object. This information will include the following sections:

- Features and Models: Description

- Basic Searching Concepts

- Normalized Correlation

- Normalized Correlation Shape Score

- Normalized Correlation Optimization (Accept and Confusion)

- Setting the Accept and Confusion property values

- Additional Hints Regarding Accept and Confusion properties

- Correlation objects and rotation

- Model Training for angle searching

- Searching repeatability and accuracy

- Calibrating the camera to subject distance

Correlation object models and features

It is important to understand the difference between features and models when using Correlation objects.

A feature is any specific pattern of gray levels in a search window. It can be anything from a simple edge a few pixels in area to a complex pattern of tens of thousands of pixels in area.

The correlation operation measures the extent to which a feature in a search window matches a previously taught model of that feature. A feature is something within an actual search window as opposed to a model or template that is an idealized representation of a feature.

A model is a pattern of gray levels used to represent a feature. It is equivalent to a template in a template matching system.

If the model has two gray levels, it is a binary model. If it has more than two gray levels, it is a gray model. All models used with Vision Guide 7.0 are gray models because gray models are more powerful in that they more closely represent the true feature than a binary model can. This helps produce more reliable results.

It is common to train a representative model from one search window to be used to search for similar features in that search window. The figure below shows a search window containing a feature of interest; a cross.

To train a model of the cross, you define the model window and click the Teach Button on the Object tab. (For details on teaching models, refer to *Using Correlation Objects* later in this chapter.

As a result, the model is created as shown in the right side figure and can be used to search for other crosses within the search window.
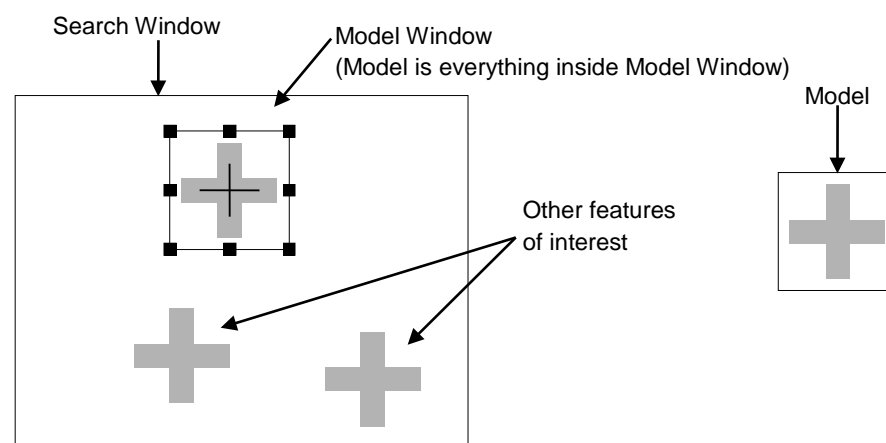
Search window containing several features of interest (left) and model trained from image (right)

Basic Searching Concepts

Searching locates features by finding the area of the search window to which the model is the most similar. The figure below shows a model and a search window, and the areas within the search window that are most similar to the model (peaks). A model similar to that shown in the figure might be used to search for a feature such as a fiducial mark on a printed circuit board. A robot could then use the position data returned by the search function to find the location of the board for placing components or for positioning the board itself.
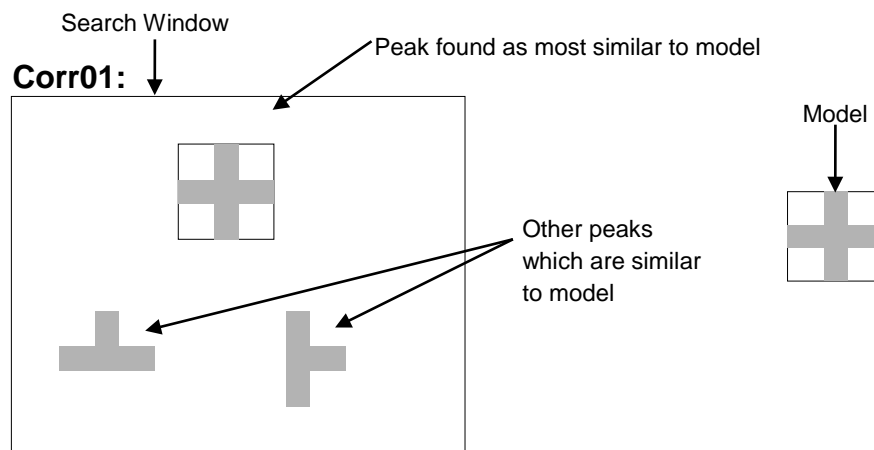
Search window, model, and peaks

There are a number of strategies that can be used to search for a model in a search window. The most commonly used method is the exhaustive search method for finding a match for the model. In the exhaustive search method, the model is evaluated at every possible location in the search window. The location in the search window with the greatest similarity is returned. Assume for example, a search window is 36 pixels square (36 x 36), and the model was 6 pixels square. To locate a match for the model, similarity would be assessed at all 961 possible locations. This method can return reliable results but the trade off is that it is very slow.

The Vision Guide 7.0 searching method is a vast improvement over exhaustive searching. First, the search window is scanned to locate positions where a match is likely. Similarity is then assessed only for those candidate locations. The candidate of the best match is returned. This technique results in execution times that can be several thousand times faster than those achieved by an exhaustive search.

Normalized Correlation

Normalized correlation is a measure of the geometric similarity between an image and a model, independent of any linear differences in search window or model brightness.

Normalized correlation is used as the searching algorithm for Correlation objects of the Vision Guide 7.0 since it is powerful and robust. The normalized correlation value does not change in any of the following situations:

- If all search window or model pixels are multiplied by some constant.

- If a constant is added to all search window or model pixels.

One of the most important characteristics of the normalized correlation is that the correlation value is independent of linear brightness changes in either the Search window or the Model. This is important because the overall brightness and contrast of an image are determined by factors such as illumination intensity, scene reflectivity, camera aperture, sensor gain and offset (including possible automatic gain control circuitry), and the gain and offset of the image digitizer. All of them are hard to be controled in most production situations. For example, bulbs age, ambient light levels can change with time of day, cameras and digitizers may be defective and need to be replaced, and the reflectivity of objects being inspected can vary.

Another important characteristic of normalized correlation is that the shape score value (see Normalized Correlation Shape Score subheading on the next page) has absolute significance; that is, you can define a perfect match that is the same for all models and search windows.

It is independent not only of image brightness and contrast but also of model brightness, contrast, and size. It is this property that allows the shape scores to be used as a measure of feature quality for inspection applications.

If the shape score calculated for a search window and model is 900, they are very similar. If the value is 100, they are not similar. These statements are valid without knowing anything about the model or search window area. This means the correlation coefficient has absolute meaning.

Normalized Correlation Shape Score

For the normalized correlation, the shape score of a feature (the extent to which it is similar to the model) is defined as a value between 0 and 1000. The larger the shape score, the greater the similarity between the feature and the model. (A shape score of 1000 represents a perfect match.)

The shape score is the value returned by the Correlation object as the Score result. (Additional information for the Score result is available in the *Vision Guide 7.0 Properties and Results Reference Manual*.)

Normalized Correlation Optimization (Accept and Confusion)

Traditional template matching is much too slow for most practical searching applications. This is because it is based on exhaustive correlation at every possible search position.

The solution to this problem lies in the use of a suitable directed search. In a directed search, data derived during the search are used to direct the search toward those locations that are promising and away from those that are not.

Searching uses an adaptation of a type of directed search called hill climbing. The idea of hill climbing is that the peak of a function of one or more variables can be found entirely from local data, by always moving in the direction of steepest ascent. The peak isearched when each of the neighboring points is lower than that point.

Hill climbing alone can fail under certain circumstances:

- A false maximum can be mistaken for the peak of the hill.

- If a plateau is reached there is no way to determine the direction in which the search should resume.

Another problem with hill climbing is determining the starting points: too few expeditions might miss significant hills; too many will eliminate the speed advantage that hill climbing provides by causing the same hill to be climbed from different directions.

Certain estimates can be made about the hills based on the search window area and the Model which help to overcome these problems.

Correlation search functions mathematically equivalent to a filtering operation: the correlation function is the output of a specific, known filter (the model), a filter that amplifies certain spatial frequencies and attenuates others. In addition, if the frequency content of a given portion of a search window is not similar to that of the model, there is no need to start a hill climbing expedition in this region.

By examining the transfer function of the model, the system can estimate the spatial frequency content of the correlation function, and, in turn the minimum spacing and size of the hills. Knowing the minimum spacing allows the system to plan where to start the hill climbing expeditions. Knowing the minimum size allows the system to avoid getting trapped by false peaks.

In addition to information obtained from the model, a pair of Vision Guide properties (i.e. the Accept property and the Confusion property) is specified to control the search.

The Accept property specifies the shape score that a feature must equal or exceed to be considered "Found" (i.e. Found result returns TRUE) by the searching software. If a rough estimate of the height of a hill does not exceed the Accept property value in a given region of a search window, hill climbing is terminated for that region.

The Confusion property indicates the amount of confusion expected in the search window. Specifically, it is the highest shape score a feature can get that is not an instance of the feature for which you are searching. The Confusion property gives the system an important hint about the scene to be searched; namely, that if a feature gets a shape score above the confusion threshold, it must be an instance of the feature for which you are searching.

The system uses the Confusion property and the number of results (specified by the NumberToFind property) to determine which hills need to be climbed and which do not. Specifically, the search can terminate as soon as the expected number of features are found whose shape score is above the Confusion property threshold and the Accept property threshold.

To start the search, a number of hill climbing expeditions are conducted in parallel, starting at positions determined from the transfer function of the Model. As hill climbing progresses, a more and more accurate estimate of each hill's height emerges until the actual peak is found.

The hills are climbed in parallel, one step at a time, as long as no hill has an estimated height that exceeds the threshold set by the Confusion property. If an expedition reaches the Confusion property threshold, the hill is immediately climbed to its peak.

Setting the Accept and Confusion Property Thresholds

Both the Accept and Confusion properties affect searching speed for Correlation objects.

The Accept property influences searching speed by providing a hint as to when to pursue the search in a given region of the scene.

When the Accept property is set high, features must be very similar to the model. Therefore, many regions can be ruled out by a cursory examination and not pursued further. If the Accept property is set to a low value, features that are only slightly similar to the Model may exceed the Accept property threshold, so that a detailed examination of more regions in the scene is needed.

Thus increasing the Accept property tends to increase speed. (i.e. higher Accept property values can make Correlation objects run faster.)

The Confusion property interacts with the number of results expected to influence searching speed. Together, the Confusion property and the number of results expected allow the system to quit the search before exploring all possible regions of the image.

Set the Accept property so that it will allow the system to find features that are examples of the "worst case degradation" you are willing to accept. The degradation may be caused by defects, scale, rotation or video noise.

For the Accept property, the default value is set to 700 in the Vision Guide 7.0. This is usually a good starting point for many applications. However, experimentation and correction will help you home in on the best value for your situation. Keep in mind that you do not always have to get perfect or nearly perfect scores for an application to function well. Even the shape scores of 200 provide good positional information for some applications, depending on the type of degradation to which a feature is subject. However, it is normally recommended that a shape score of above 500 is used for the Accept property for most applications.

Set the Confusion property based on the highest value you expect the "wrong thing" to get (plus a margin for error). The confusion threshold should be greater than or equal to the Accept property threshold. Setting the Confusion property to a high value will increase the time of the search, but may be necessary to insure that the right features are found. The Confusion property default value is 800 but should be adjusted depending upon the specific application requirements.

The figure below shows a scene where there is little confusion: the round pad is not very similar to the fiducial (cross). The Confusion property can therefore be set to a fairly low value (around 500).

The Accept property is normally set less than or equal to the Confusion property, depending upon the amount of degradation you are willing to accept. Assuming this scene has little degradation, a shape score of 920 could be expected.
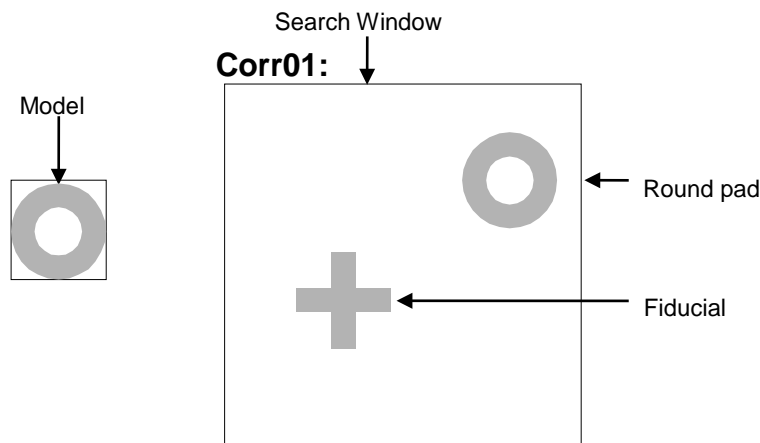
A scene with little confusion

The figure below shows a scene where there is a lot of confusion; both the feed through hole and the IC pad are similar to the round pad. The Confusion property should therefore be set to a fairly high value (around 820).
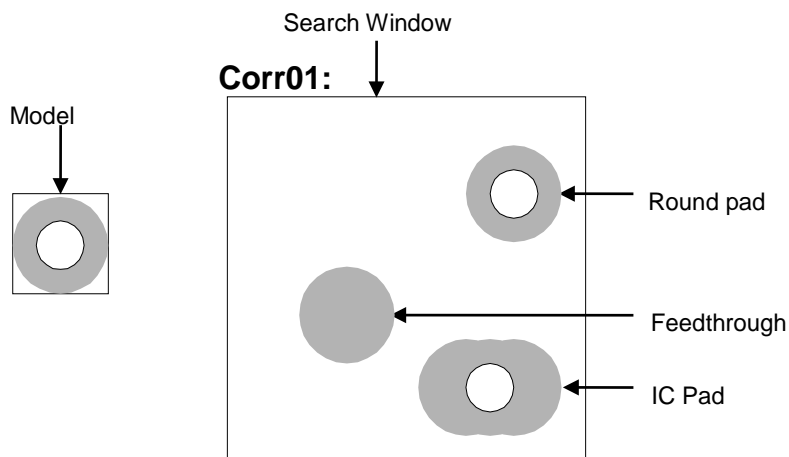
A scene with a high degree of confusion

Additional Hints Regarding Accept and Confusion Properties

A search window that has a region of constant gray value will always get a 0 correlation value in that region. If a scene basically has a uniform background (e.g. a white piece of paper), there will be no correlation in most areas. Therefore, if the Correlation object finds anything, you can set the Confusion property to a low value since the found part should be the feature you are searching for.

The Accept and Confusion properties can be thought of as hints that you provide to the system to enable it to locate features more quickly.

In general, these properties should be set conservatively, but need not be set precisely. The most conservative settings are a low Accept property and high Confusion property.

Use very conservative settings when you know very little about a scene in which you are searching; the search will be careful but slower. (This is especially important when using Correlation property positional results to give to the robot to move to.)

Use more liberal settings when you know a lot about a scene in which you are searching. For example, if you know that you are looking for one feature, and the rest of the scene is blank, a careful search is unnecessary; use more liberal settings and the search will be faster.

Using the Multiple Results Dialog to Debug Searching Problems

Sometimes the parts that you are working with vary considerably (even within the same production lot) and sometimes there are 2 or more features on a part which are similar. This can make it very difficult to determine a good Accept property value. Just when you think you have set the Accept property to a good value, another part will come in which fools the system. In these cases it can be very difficult to see what is going on.

The ShowAllResults dialog was created to help solve these and other problems.

While you may only be interested in one feature on a part, requesting multiple results can help you see why a secondary feature is sometimes being returned by Vision Guide 7.0 as the primary feature you are interested in. This generally happens a few different ways:

1. When two or more features within the search window are very similar and as such have very close Score results.

2. When the Confusion or Accept properties are not set high enough which allow other features with lower scores than the feature you are interested in to meet the Accept property setting .

Both of the situations above can be quite confusing for the beginning Vision Guide 7.0 user when searching for a single feature within a search window.

If you have a situation where sometimes the feature you are searching for is found and sometimes another feature is found instead, use the Show All Results dialog to home in on the problem. Follow the following steps to get a better view of what is happening:

(1) Set your NumberToFind property to 3 or more.

(2) Run the vision object from the Vision Guide 7.0 Development Environment.

(3) Click on the <ShowAllResults> property button to bring up the Show All Results dialog.

(4) Examine the scores of the top 3 or more features that were found.

(5) If only one or two features were found (Vision Guide 7.0 will only set scores for those features that are considered found) reduce your Accept property so that more than one feature will be found and Run the vision object again. (You can change the Accept level back after examining the ShowAllResults dialog)

(6) Click on the <ShowAllResults> property button to bring up the Show All Results dialog.

(7) Examine the scores of the top three or more features that were found.

Once you examine the scores of the top three or more features that were found as described above, it should become clear to you what is happening. In most cases you will see one of these two situations.

1. Each of the features that were found has a score greater than the Accept property setting. If this is the case, simply adjust your Confusion property value up higher to force the best feature to always be found rather that allowing other features to be returned because they meet the Accept threshold. You may also want to adjust the Accept property setting.

2. Each of the features are very close in score. If this is the case, then you will need to do something to differentiate between the feature which you are primarily interested in such as:

   - Readjust the search window so that the features that are randomly returning as the found feature are not contained inside.

   - Teach the Model again for the feature that you are most interested in.

   - Adjust the lighting for your application so that the feature that you are most interested in gets a much higher score than the other features that are currently fooling the system.

See the section *6.2.16 Working with Multiple Results from a Single Object* later in this chapter for more information on using multiple results.

Correlation Objects and Rotation

As with any template matching procedure, shape score and location accuracy degenerate if the size or the angle of the feature differs from that of the model. If the differences are large, the shape score will be very low, or the search operation will not find the feature.

The exact tolerance to angle and size changes depends on the Model, but typically falls between 3 and 10 degrees for angle and 2 to 5 percent for size. Exceptions include rotationally symmetric models, such as circles, which have no angle dependence; and simple edge and corner models, which have no size dependence.

Visualize two different scenes within a search window. The first scene is the picture of a human face where the nose was taught as the model. The nose is not considered XY symmetric so rotation will dramatically affect the accuracy of position for this feature. The second scene is a printed circuit board where a fiducial (similar to one in the figure) is the mode. In this case the fiducial mark (a cross) is XY symmetric, so rotation does not have an immediate devastating effect on positional accuracy for this feature.

Basically, models that have a predominance of fine features (such as a nose, picture of a flower, tree or others) will be less tolerant to rotation. Features that are more symmetric, like the cross, will be more tolerant of rotation. However, with this in mind we strongly recommend that you use Polar objects to determine rotation angles. The Correlation object can be used to find the XY position and then the Polar object can be associated with the Correlation object such that it can use the XY position as a center to work from to find

the angle of the feature. See the *6.2.6 Polar Object* in this chapter for more information on how to associate Polar objects with Correlation and other vision objects.

There are a variety of techniques that can be used in situations where significant angle or scale changes are expected. The primary techniques are:

- Break up complex features into smaller, simpler ones. In general, small simple models are much less sensitive to scale and angle changes than larger, more complex models.

- Use the angle related properties (AngleEnable, AngleRange, and AngleMaxIncrement) to help find rotational angles when working with Correlation objects. This is appropriate for locating complex features in complex scenes, when those features cannot be broken up into simpler ones. This capability is implemented as a series of models at various angles. It can be many times slower than a normal search.

NOTE
☞
To use the search with angle capabilities of the Correlation object, the Model for the Correlation object must be taught with the AngleEnable property set to True. This causes the Correlation object to be taught at a variety of angles as defined by the AngleRange and AngleMaxIncrement properties.

- Use Polar objects in conjunction with Correlation objects to determine the angle of rotation of a part. (See *6.2.6 Polar Object* ⌖ in this chapter.)

Model Training for Angle Searching

To Search with angle measurement, you must first direct the system to teach a series of rotated models. Setting the AngleEnable property to True and using the AngleRange property to specify the range of angles over which models will be taught does this. When you teach rotated models this way, searching automatically creates a set of models at various rotations in equal angular increments over that range.

You can also specify a maximum angle increment at which the models will be taught within the angular range. This is accomplished by setting an increment value in the AngleMaxIncrement property of a Correlation object.

However, keep in mind the following regarding the AngleMaxIncrement property:

- If you provide a maximum angle increment, the model training function selects an angular increment automatically and uses the smaller of the automatically selected increment and the maximum angle increment you provide.

- If you set the AngleMaxIncrement property to 0, the model teaching function selects an angle increment automatically and uses that angle increment. In this case the system typically sets an angle increment between 2-5 degrees. This results in the smallest model storage requirements and the fastest search times, but may produce results that are more coarse than desired.

If you wish to measure angle precisely, you should set the AngleMaxIncrement property to an increment corresponding to the degree of precision you desire. Keep in mind though, that the smaller the angle increment, the more storage will be required for the model and the slower the search time will be.

NOTE
☞
We recommend using the Polar object to determine angle whenever possible. This provides much more reliable and accurate results that are required using vision for robot guidance.

Keep in mind that when training models with angle, the search window must be large enough to allow the model to be rotated without any part of the model going outside of the search window.

Searching Repeatability and Accuracy

Searching repeatability and accuracy is a function of the size and details of the Model (shape, coarseness of features, and symmetry of the model), and the degradation of the features as seen in the search window (noise, defects, and rotation and scale effects).

To measure the effect of uncorrelated noise on position, you can perform a search in a specific search window that contains a non-degraded feature, and then perform the exact same search again (acquiring a 2nd image into the frame buffer) without changing the position of the object, and then comparing the measured positions.

This can be easily done by following steps:

1. Click the <Run Object> button on the Object tab two or more times

2. Click the <Statistics> toolbar button.

3. The Statistics dialog can then be used to see the difference in position between the 2 object searches.

For a large model (30x30) on an non-degraded feature the reported position can be repeatable to 1/20 of a pixel. However, in most cases it is more realistic to achieve results of just below a pixel. (1/2, 1/3, or 1/4 pixel)

Searching accuracy can be measured by performing a search in a specific search window that contains an undegraded feature, moving the object an exact distance and then comparing the reported position difference with the actual difference. If you have a large model (30x30 or greater), no degradation, no rotation or scale errors, and have sufficient edges in both X and Y directions, searching can be accurate to 1/4 pixel. (Keep in mind that this searching accuracy is for the vision system only and does not take into effect the inaccuracies that are inherent with all robots. So if you try to move the part with the robot you must also consider the inaccuracies of the robot mechanism itself.)

The effects of rotation and scale on searching accuracy depend on the Model:

- Models that are rotationally symmetric do well.

- Models that have fine features and no symmetry do not do well.

Calibrating the Camera to Subject Distance

For optimal searching results, the size of the features in an image should be the same at search time as it was when the model was taught.

Assuming the same camera and lens are used, if the camera to subject distance changes between the time the model is trained and the time the search is performed, the features in the search window will have a different apparent size. That is, if the camera is closer to the features they will appear larger; if the camera is farther away they will appear smaller.

NOTE
☞   If the camera to subject distance changes, you must re-train the model.

Using Correlation Objects

Now that we've reviewed how normalized correlation and searching works we have set the foundation for understanding how to use Vision Guide 7.0 Correlation objects.

This section will describe the steps required to use Correlation objects as listed below:

- Create a new Correlation object

- Position and Size the search window

- Position and size the model window

- Position the model origin

- Configure properties associated with the Correlation object

- Teach the Model

- Test the Correlation object & examine the results

- Make adjustments to properties and test again

- Working with Multiple Results from a Single Correlation object

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence ☐ > toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for more details on how to create a new vision sequence or select one that was previously defined.

Step 1: Create a new Correlation object

(1) Click on the <New Correlation ⬚ > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button)

(2) The mouse cursor will change to a correlation icon.

(3) Move the mouse cursor over the image display of the Vision Guide window and click the left mouse button to place the Correlation object on the image display

(4) Notice that a name for the object is automatically created. In the example, it is called "Corr01" because this is the first Correlation object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a Correlation object similar to the one shown below:

New Correlation object layout
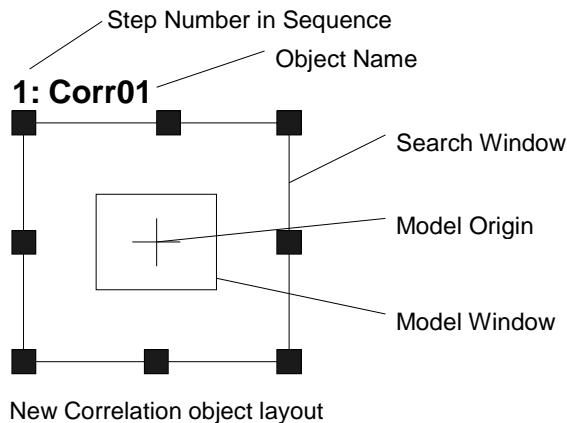
(1) Click on the name label (or on one of the sides) of the Correlation object and while holding the mouse down drag the Correlation object to the position where you would like the top left position of the search window to reside.

(2) Resize the Correlation object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.) The search window is the area within which we will search.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Step 3: Position and size the model window

(1) The search window for the Correlation object you want to work on should be magenta in color and the sizing handles should be visible on each corner and along the middle of each side of the search window. If you cannot see the size handles click on the name field of the Correlation object. Once you can see the sizing handles of the Correlation object you want to work on and it is magenta in color go to step 2.

(2) Click on one of the lines of the box that forms the model window. This will cause the model window to be highlighted. (You should see the size handles on the model window now.)

(3) Move the mouse pointer over one of the lines of the box which forms the model window and while holding the mouse down drag the model window to the position where you would like the top left position of the model window to reside.

(4) Resize the model window as required using the model window size handles . (This means click on a size handle and drag the mouse.) (The model window should now be outlining the feature that you want to teach as the model for this Correlation object.)

Your Correlation object layout should now look something like the example in the figure below where the search window covers the area to be searched and the model window outlines the feature you want to search for. Of course your search window and Model will be different but this should give you an idea of what was expected so far.
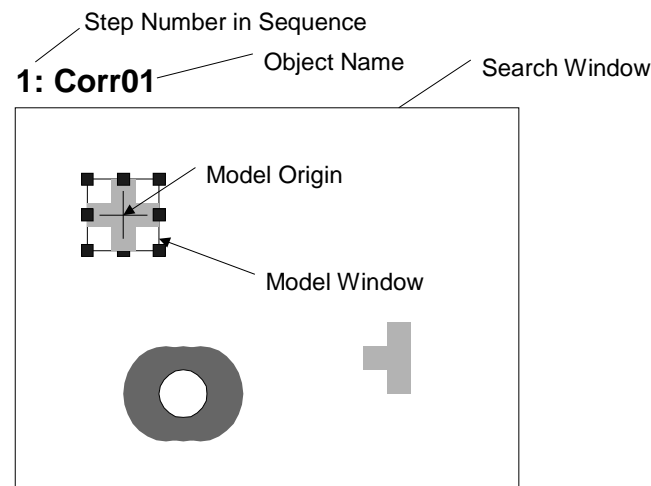
Correlation object after search and model window positioning and resizing

Tips on Setting Proper Size and Position for the model window:

The size and position of the model window is very important since it defines the feature to be searched for. When creating a model window for a Correlation object, there are 2 primary items you must pay attention to: (1) If you expect that your part will rotate much, smaller models are better because they require smaller search windows to Search within. (2) Making the model window as close in size to the search window as possible can reduce execution time.

It is also sometimes a good idea to make the model window just a bit larger than the actual feature you are interested in. This will give the feature some border that may prove useful in distinguishing this object from others. Especially when two objects are positioned right next to each other and are touching. However, this additional border should only be a few pixels wide. Keep in mind that each vision application is different so the best model window sizing technique will vary from application to application.

Step 4: Position the Model Origin

The model origin defines the position on the model that will be returned back as the position of the feature when you run the Correlation object. This means that the model origin should be placed in a place of significance if the position data is important. For example, when using a Correlation object to find parts for a robot to pick up or place, it is important that the position of the model origin is in a location where the robot can easily grip the part. This is because that will be the position the robot will move to based on the RobotXYU, RobotX or RobotY result.

When a new Correlation object is created the ModelOrgAutoCenter property is set to True. (True is the default value for the ModelOrgAutoCenter property) This means that the model origin is set to the center of the model window automatically and cannot be moved manually.

If you want to move the model origin manually, you must first set the ModelOrgAutoCenter property to False. The steps to do this and also actually position the model origin are shown below.

(1) Look at the Object tab that you just brought forward in Step 4 as explained on the previous page. Find the ModelOrgAutoCenter property on the properties list and click in the value field.

(2) You will see a drop down list with 2 choices: True and False. Click on the False choice. You have now set the ModelOrgAutoCenter property to False and can move the model origin with the mouse.

(3) Click on the model window to highlight the model window.

(4) Click on the model origin and keep the mouse button held down while dragging the model origin to a new position. It should be noted that the model origin can only be positioned within the bounds of the model window.

Step 5: Configure the Correlation object properties

We can now set property values for the Correlation object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the Correlation object. Descriptions of other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects, can be seen in the Vision Guide 7.0 Properties and Results Reference Manual or in the *Correlation Object Properties* list.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Name property   The default name given to a newly created Correlation object is "Corrxx" where xx is a number which is used to distinguish between multiple Correlation objects within the same vision sequence. If this is the first Correlation object for this vision sequence then the default name will be "Corr01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Correlation object's name is displayed is updated to reflect the new name.

Accept property   The Accept property sets the shape score that a feature must meet or beat to be considered Found. The value returned in the Score result is compared against this Accept property Value. The default value is 700 which will probably be fine before we try running the Correlation object for the first time.

Confusion property   If there are many features within the search window which look similar, the Confusion property is useful to help "home in" on the exact feature you want to find. The default value is 800 that will probably be fine before running the Correlation object for the first time.

ModelOrgAutoCenter   If you want to change the position of the model origin you must first set the
property   ModelOrgAutoCenter property to False.
Default: True

Frame property   Allows you to select a previously defined Frame object as a reference frame for this Correlation object. The details for Frames are defined in *6.12.3 Frame Object* in this chapter.

NumberToFind   Depending upon the number of features you want to find, you may want to
property   set the NumberToFind property larger than 1. This will allow one Correlation object to find multiple features within one search window.

AngleEnable property   You must set this property to True if you want to use a Correlation model to search with angle. This must be set to True at the time you teach the Model so that multiple models can be configured.

AngleMaxIncrement and AngleRange properties

These properties are used along with the AngleEnable property for using the Correlation model to search with angle.

RejectOnEdge   Allows you to exclude the parts touching the boundary of the search
property   window. Normally, this should be set to True.

It is possible to leave the properties as default and go on to the next step. The properties can be set later as necessary.

Step 6: Teach the model for the Correlation object

The Correlation object needs a model to search for and this is accomplished a process called teaching the model. You should have already positioned the model window for the Correlation object to outline the feature that you want to use as a model. Teaching the model is accomplished as follows:

(1) Make sure that the Correlation object is the currently displayed object in the Object tab. You can look at the current object drop down list at the top of the Object tab to see which object is the object you are currently working on or you can look at the image display and see which object is highlighted in magenta.

(2) Click on the <Teach> button at the bottom right of the Object tab. It will take only a few seconds for the Model to be taught in most cases. However, if you are teaching a model when the AngleEnable property is set to True it can take quite a few seconds to teach the model because the system is actually teaching many models each at a slight angle offset from the previous.

Step 7: Test the Correlation object / examine the results

To run the Correlation object, click the <Run Object> button located at the bottom left of the Object tab.

Results for the Correlation object will now be displayed. The primary results to examine at this time are:

Found result - Returns whether or not the Correlation was found. If the feature you are searching for is found this result returns as True. If the feature is not found, the Found result returns a False and is highlighted in red. If the feature was not found read on to Step 9 for some of the more common reasons why a Correlation object is not found.

FoundOnEdge result

- This result will return as True if the feature was found where a part of the feature is touching the boundary of the search window. In this case the Found result will return as False.

Score result - This tells us how well we matched the model with the feature that most closely resembles the model. Score results range from 0 to 1000 with 1000 being the best match possible. Examine the Score result after running a Correlation object as this is your primary measure of how well the feature was found.

Time result - The amount of time it took for the Correlation object to execute. Remember that small search windows and small Models help speed up the search time.

NumberFound result

- When searching for more than 1 Correlation object, the NumberFound result returns the number of features that matched the Correlation object's Model.

Angle result - The angle at which the Correlation is oriented. This is computed based on the original angle of the Model. However, this value may sometimes be coarse in value and not so reliable. (We strongly recommend using the Polar object for finding Angles. Especially for robot guidance.)

PixelX, PixelY results

- The XY position (in pixels) of the feature. Remember that this is the position of the model origin with respect to the found feature. If you want to return a different position, you must first reposition the model origin and then re-teach the Model.

CameraX, CameraY results

- These define the XY position of the found feature in the Camera's Coordinate system. The CameraX and CameraY results will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.

RobotX, RobotY results

- These define the XY position of the found feature in the Robot's Coordinate system. The robot can be told to move to this XY position. (No other transformation or other steps are required.) Remember that this is the position of the model origin with respect to the found feature. If you want to return a different position, you must first reposition the model origin and then re-teach the Model. The RobotX and RobotY results will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.

RobotU result - This is the angle returned for the found feature translated into the Robot's Coordinate system. The RobotU result will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.

ShowAllResults - If you are working with multiple results, you may want to click on the button in the ShowAllResults value field. This will bring up a dialog to allow you to examine all the results for the current vision object.

NOTE

☞ The RobotX, RobotY, RobotU, RobotXYU and CameraX, CameraY, CameraU, CameraXYU results will return "no cal" at this time since we have not done a calibration in the example steps described above. This means that no calibration was performed so it is impossible for the vision system to calculate the coordinate results with respect to the Robot coordinate system or Camera coordinate system. Refer to *7. Vision Calibration* for more information.

Step 8: Make adjustments to properties and test again

After running the Correlation object a few times, you may have encountered problems with finding a Correlation or just want to fine-tune some of the property settings. Some common problems and fine-tuning techniques are described in the next section called "Correlation object Problems".

Correlation Object Problems

If the Correlation object returns a Found result of False, there are a few places to immediately examine.

- Look at the Score result which was returned. Is the score result lower than the Accept property setting. If the Score result is lower, try changing the Accept property a little lower (for example, below the current score result) and run the Correlation object again.

- Look at the FoundOnEdge result. Does it have a return value of True? If it is True, this means that the feature was found but it was found where a part of the feature is touching the search window. This causes the Found result to be returned as False. To correct this situation, make the search window larger or if this is impossible, try changing the position of the camera, or resizing the model window.

If the Correlation object finds the wrong feature, there are a couple of places to look:

- Was the Accept property set high enough? If it is set rather low this could allow another feature to be found in place of the feature you are interested in.

- Was the Confusion property set high enough? Is it higher than the Accept property? The Confusion property should normally be set to a value equal to or higher than the Accept property. This is a good starting point. But if there are features within the search window which are similar to the feature you are interested in, then the Confusion property must be moved to a higher value to make sure that your feature is found instead of one of the others.

- Adjust the search window so that it more closely isolates the feature that you are interested in.

Correlation Object Fine Tuning

Fine-tuning of the Correlation object is normally required to get the object working just right. Following is a description for the primary properties associated with fine-tuning of a Correlation object and model addition:

Accept property	After you have run the Correlation object a few times, you will become familiar with the shape score which is returned in the Score result. Use these values when determining new values to enter for the Accept property. The lower the Accept property, the faster the Correlation object can run. However, lower Accept property values can also cause features to be found which are not what you want to find. A happy medium can usually be found with a little experimentation that results in reliable feature finds that are fast to execute.
Confusion property	If there are multiple features within the search window which look similar, you will need to set the Confusion property relatively high. This will guarantee that the feature you are interested in is found rather than one of the confusing features. However, higher confusion costs execution speed. If you don't have multiple features within the search window that look similar, you can set the Confusion property lower to help reduce execution time.
Add another sample	"Add another sample" can be selected when teaching is performed with the model window whose size is same as the current model's model window. When the model slightly changes (shape or pattern is slightly different, shadow appears differently, etc.), selecting the changed model to add a new model may stabilize the score at object execution. The original model will be kept if the angle is largely out of position, or the model cannot be added due to significant difference.

Once you have completed adjusting and have tested the Correlation object until you are satisfied with the results, you are finished with creating this vision object and can go on to creating other vision objects or configuring and testing an entire vision sequence.

Other Useful Utilities for Use with Correlation Objects

At this point you may want to consider examining the histogram feature of Vision Guide 7.0. Histograms are useful because they graphically represent the distribution of gray-scale values within the search window. The details regarding Vision Guide histogram usage are described in *8.1 Using Histograms*.

You may also want to use the statistics feature of Vision Guide to examine the Correlation object's results statistically. An explanation of the Vision Guide statistics features are explained in *9. Using Vision Guide Statistics*.

6.2.4 Blob Object

Blob Object Description

Blob objects compute geometric, topological and other features of images. Blob objects are useful for determining presence/absence, size and orientation of features in an image. For example, Blob objects can be used to detect the presence, size and location of ink dots on silicon wafers, for determining the orientation of a component, or even for robot guidance.

Some of the features of the computed by Blob objects are:

- -Area and perimeter

- Center of mass

- Principal axes and moments

- Connectivity

- Extrema

- Coordinate positions of the center of mass in pixel, camera world, and robot world Coordinate systems

- Holes, roughness, and compactness of blobs

Blob Object Layout

The Blob object layout is rectangular just like the correlation objects. However, Blob objects do not have models. This means there is no need for a model window or model origin for the Blob object layout. As shown below, Blob objects only have an object name and a search window. The search window defines the area within which to search for a blob. The Blob object layout is shown below:
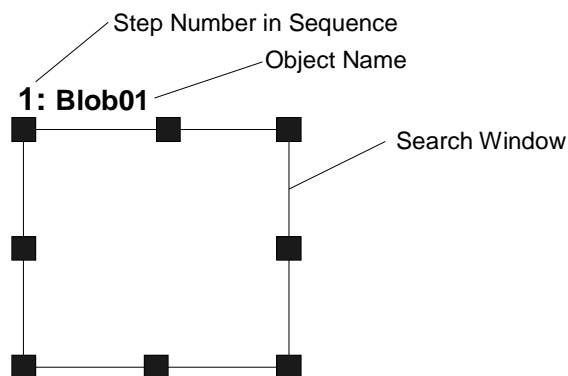
Blob Object Layout

Blob Object Properties

The following list is a summary of properties for the Blob object. The details for each property are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Caption	Assigns a caption to the Geometric object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the Geometric Search. This property is normally set to base the center of the Geometric Search on the XY position result from another vision object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results. Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0

Property	Description
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result.
	Default: False
CurrentResult	Defines which result to display in the Results list (on the Object tab) or which result to return data for when the system is requested to find more than one of a like feature within a single search window.
	Default: 1
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color of an object when it is not accepted.
	Default: Red
FillHoles	Specifies whether to fill the holes in a binary image.
	Default: False
Frame	Specifies which positioning frame to use.
	Default: None
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
LabelBackColor	Selects the background color for an object label.
	Default: Transparent
MaxArea	Defines the upper Area limit for the Blob object. For a Blob to be found it must have an Area result below the value set for MaxArea property.
	Default: 100,000
MinArea	Defines the lower Area limit for the Blob object. For a Blob to be found it must have an Area result above the value set for MinArea property.
	Default: 25
MinMaxArea	Runtime only. Sets or returns both MinArea and MaxArea in one statement.
Name	Used to assign a unique name to the Blob object.
	Default: Blob01
NumberToFind	Defines the number of blobs to find in the search window.
	Default: 1
PassColor	Selects the color for an object when it is passed.
	Default: Light Green
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound

Property	Description
Polarity	Defines the differentiation between objects and background. (Either "Dark Object on Light Background" or "Light Object on Dark Background".) Default: 1 - DarkOnLight
RejectOnEdge	Determines whether the part will be rejected if found on the edge of the search window. Default: 0 - False
SearchWin	Runtime only. Sets or returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched in pixels. Default: 100
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinType	Defines the type of the area to be searched (i.e. Rectangle, RotatedRectangle, Circle).
SearchWinWidth	Defines the width of the area to be searched in pixels. Default: 100
SizeToFind	Selects which size of blobs to find. Default: 1 - Largest
Sort	Selects the sort order used for the results of an object. Default: 0 – None
ThresholdAuto	Specifies whether or not to automatically set the threshold value of the gray level that represents the feature (or object), the background, and the edges of the image. Default: Disables
ThresholdColor	Defines the color assigned to pixels within the thresholds. Default: Black

Property	Description
ThresholdHigh	Works with the ThresholdLow property to define the gray level regions that represent the feature (or object), the background, and the edges of the image.
	The ThresholdHigh property defines the upper bound of the gray level region for the feature area of the image. Any part of the image that falls within gray level region defined between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is part of the feature.)
	If the ThresholdAuto property is "True" and the ThresholdColor property is "White", this property value will be set to 255 and cannot be changed. Default: 128
ThresholdLow	Works with the ThresholdHigh property to define the gray level regions that represent the feature (or object), the background, and the edges of the image. The ThresholdLow property defines the upper bound of the gray level region for the feature area of the image. Any part of the image that falls within gray level region defined between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is part of the feature.)
	If the ThresholdAuto property is "True" and the ThresholdColor property is "Black", this property value will be set to 0 and cannot be changed. Default: 0

Blob Object Results

The following list is a summary of the Blob object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
Angle	Returns the amount of part rotation in degrees.
Area	Returns the area of the blob in pixels.
CameraX	Returns the X coordinate position of the found part's position in the camera coordinate system.
CameraY	Returns the Y coordinate position of the found part's position in the camera coordinate system.
CameraXYU	Runtime only. Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system.
Compactness	Returns the compactness of a blob.
Extrema	Runtime only. Returns MinX, MaxX, MinY, MaxY pixel coordinates of the blob Extrema.

Results	Description
Found	Returns whether or not the object was found. (i.e. was a Connected Blob found which has an Area result that falls between the MinArea and MaxArea properties.)
FoundOnEdge	Returns TRUE when a Blob object is found too close to the edge of the search window.
Holes	Returns the number of holes found in the blob.
MaxX	Returns the maximum X pixel coordinate of the blob Extrema in pixels.
MaxY	Returns the maximum Y pixel coordinate of the blob Extrema in pixels.
MinX	Returns the minimum X pixel coordinate of the blob Extrema in pixels.
MinY	Returns the minimum Y pixel coordinate of the blob Extrema in pixels.
NumberFound	Returns the number of blobs found within the search window. (This number can be anywhere from 0 up to the number of blobs you requested the Blob object to find with the NumberToFind property.)
PassColor	Selects the color for passed objects. Default: LightGreen
Perimeter	The number of pixels along the outer edge of the found blob.
PixelX	Returns the X coordinate position of the found part's position in pixels.
PixelY	Returns the Y coordinate position of the found part's position in pixels.
PixelXYU	Runtime only. Returns the PixelX, PixelY, and PixelU coordinates of the found part's position in pixels.
RobotX	Returns the X coordinate of the detected object in the robot coordinate system.
RobotY	Returns the Y coordinate of the detected object in the robot coordinate system.
RobotU	Returns the U coordinate of the detected object in the robot coordinate system.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the detected object in the robot coordinate system.
Roughness	Returns the roughness of a blob.
ShowAllResults	Displays a dialog which allows you to see all results for a specified vision object in a table form. This makes it easy to compare results when the NumberToFind property is set to 0 or greater than 1.

Results	Description
Time	Returns the amount of time in milliseconds required to process the object.
TotalArea	Returns the sum of areas of all results found.

How Blob Analysis Works

Blob analysis processing takes place in the following steps:

1. Segmentation, which consists of
    - Thresholding
    - Connectivity analysis

2. Blob results computation

Segmentation

In order to make measurements of an object's features, blob analysis must first determine where in the image an object is: that is, it must separate the object from everything else in the image. The process of splitting an image into objects and background is called segmentation.

The Blob object is intended for use on an image that can be segmented based on the grayscale value of each of its pixels. A simple example of this kind of segmentation is thresholding.

While the Blob object described in this chapter produces well defined results even on arbitrary gray scale images that cannot be segmented by gray-scale value, such cases are typically of limited utility because the results are influenced by the size of the window defining the image.

For whole image blob analysis, the feature of interest must be the only object in the image having a particular gray level. If another object in the image has the same gray-scale value, the image cannot be segmented successfully. Figure A through Figure D illustrate which can and cannot be segmented by gray-scale value for whole image blob analysis.
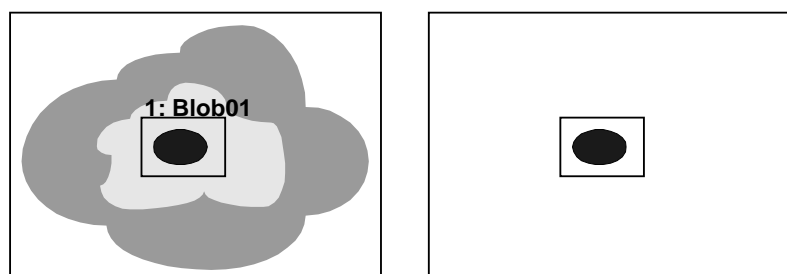
Figure A: Scene that can be segmented by gray-scale value

A shows the camera's field of view (left). The scene to be processed using a Blob object falls within the search window labeled "Blob01". After segmentation by gray-scale value, the object and the background are easily distinguishable as shown on the right side of Figure A.
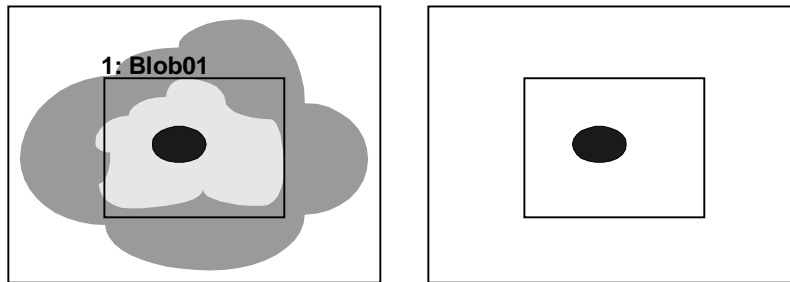
Figure B: Scene from Figure A with larger search window

Changing the size of the search window as shown in Figure B, changes only the size of the background. It has no effect on the features of the blob.

Figure C and Figure D show a similar field of view for an image. However, this scene cannot be segmented by gray-scale value because there are two objects in the image which have the same gray-scale value. Only eliminating one or the other from within the search window can separate these objects. While the image in Figure C could be segmented by gray-scale value, enlarging the search window as shown in Figure D, completely changes the segmented image. For this scene, both the area of the background and the measured features of the blob vary depending on the size of the search window and the portion of the image it encloses.
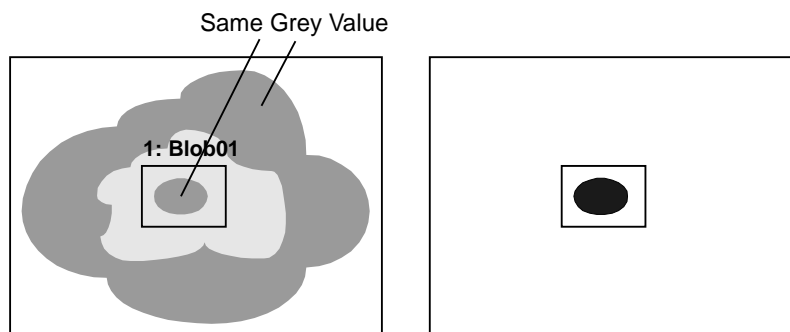
Figure C: Scene that cannot be segmented by gray-scale value
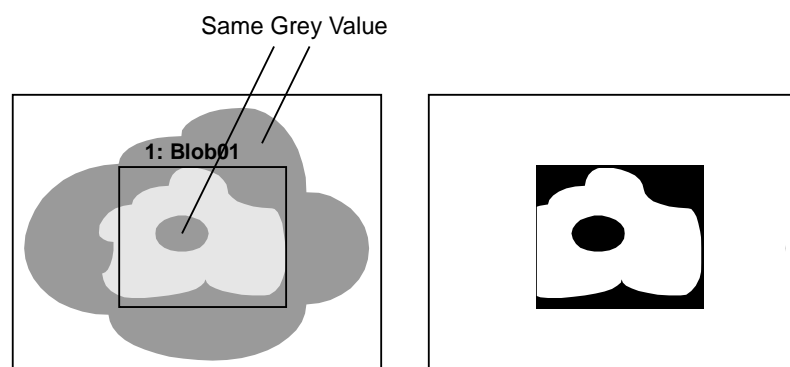
Figure D: Enlarged search window encloses two objects with the same gray scale value

You should also watch out for situations as shown in Figure E. In this example, the inner blob and part of the background have become connected. This forms one large blob that has very different features from the original center blob that we are trying to segment.

Now One Object

1: Blob01
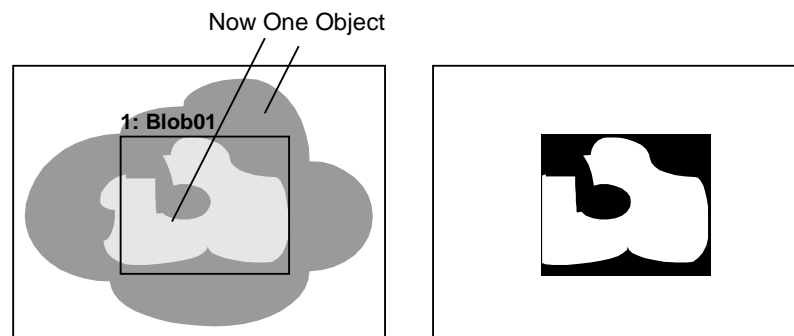
Figure E: Object and background have been connected

Thresholding

The Blob object uses thresholding to determine the weight of each pixel in an image. Two user defined thresholds are used: ThresholdLow and ThresholdHigh. Pixels whose grayscale values are between the thresholds are assigned a pixel weight of 1 and all others 0. The ThresholdColor property is used to define the color of the pixels for weight 1. This is the color (black or white) between the thresholds.

Based on these derived pixel weights, the Blob object segments the image into feature (pixels having weight 1) and background (pixels having weight 0). The Polarity property is used to configure the Blob object to find blobs containing either black or white pixels. When Polarity is DarkOnLight, then blobs contain black pixels. When Polarity is LightOnDark, then blobs contain white pixels.

Using Histograms to Determine Thresholds

By using the Vision Guide 7.0 Histogram tool, the user can determine which values to use for ThresholdLow and ThresholdHigh.

As an example, consider an ideal binary image of a black blob on a white background. The figure below illustrates such an image and its histogram.

Pixel Count

Gray Value

Feature gray value   Background gray value
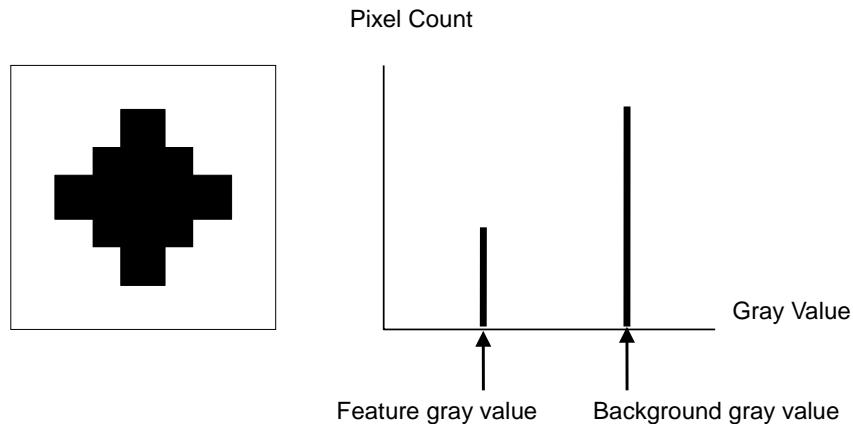
Ideal binary image and its histogram

Note that only two gray-scale values have non-zero contents in this histogram: the gray-scale value of the blob and the gray-scale value of the background.

Real images never have histograms such as this. The effects of noise from various sources (uneven printing, irregular lighting and electrical noise, for example) combine to spread out the peaks. A more realistic histogram is shown in the figure below.

Pixel
Count
ThresholdHigh

ThresholdLow

Gray
Values

Feature   Background

Blob edges
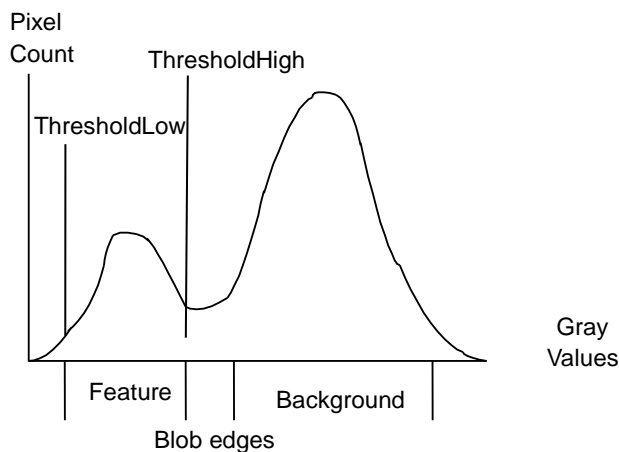
Setting the threshold values *using a histogram*

In the histogram in the figure above, each of the peaks is clearly evident. The peak areas are in the same proportion as those in the previous ideal histogram, but each has now spread to involve more than one gray-scale value. The less populated grayscale values between the two principal peaks represent the edges of the blob, which are neither wholly dark nor wholly light. You should adjust the threshold values so that the blob feature will have pixels weights of 1.

Connectivity (Connected Blob Analysis)

Connectivity can be defined as analysis based on connected pixels having non zero weight. More simply stated, connectivity is used to find a group of connected pixels that are referred to as blobs. Connectivity is performed automatically by the Blob object where connectivity is performed and then measurements are computed for the blob(s) that are found. Connectivity for Blob objects returns the number of blobs found based upon the NumberToFind property that is set before running the Blob object.

Blob Results Computations

Once all the other steps for blob analysis are complete, results can be computed for the blob that was found. The list of all results returned for the Blob object is shown in the section *Blob Object Results* previously described in this section.

A detailed explanation of every result for all vision objects is given in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Some of the results described in the *Vision Guide 7.0 Properties and Results Reference Manual* apply to many different vision objects such as the Found, Time, or PixelX results. While the Found and Time results are pretty generic and generally can be applied the same across all vision objects, some of the results such as the position related results have special meaning when applied to Blob objects. These are described below:

MinX, MinY, MaxX, MaxY Results

The MinX, MinY, MaxX and MaxY results when combined together create what is called the Extrema of the blob. The Extrema is the coordinates of the blob's minimum enclosing rectangle. The best way to understand this is to examine the figure below.
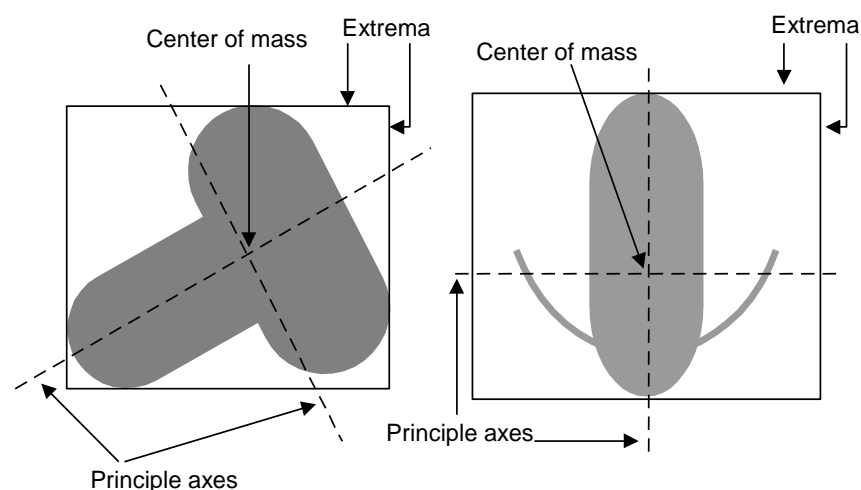
Principle axes, center of mass, and Extrema

Robot World, Camera World, and Pixel Position Data

Coordinate position results for the Blob object return the position of the center of mass. Keep in mind that the center of mass is not necessarily the center of the part. This can cause trouble for some parts if you try to use the center of mass to pick up the part. If you use the RobotX, RobotY and RobotU coordinate position results from a Blob object as a pick up position make sure that picking up the part at the center of mass is possible. If you don't want to pick up the part at the center of mass you will either have to calculate an offset or use another vision object such as a Correlation object to find the part and return a more useful position.

TotalArea Result

The TotalArea result is the sum of the areas for all results found. This is useful for pixel counting. By setting NumberToFind to 0, the Blob object will find all blobs with areas between MinArea and MaxArea. TotalArea will then show the total area of all results.

Angle Result Limitations for the Blob Object

Just a reminder that the Angle result for the Blob object is limited in its range. The Angle result for a Blob object returns angle values that range from +90 to −90. The Blob object is not able to return an angular result that ranges through an entire 360 degrees.

NOTE
☞

It should be noted that a Blob object does not always return results as reliably as a Polar object. Because the range of the Blob object Angle result is limited and in some cases not reliable, we DO NOT recommend using a Blob object Angle result for robot guidance. Instead we strongly recommend using a Polar object to compute angular orientation of a part. The Polar object can use the X, Y position found as the center of mass from a Blob object and then compute an angle based from the center of mass of the Blob object. This is explained in detail later in *6.2.6 Polar Object*.

Using Blob Objects

Now that we've reviewed how blob analysis works we have set the foundation for understanding how to use Vision Guide 7.0 Blob objects. This next section will describe the steps required to use Blob objects as listed below:

- How to create a new Blob object

- Position and Size the search window

- Configure the properties associated with the Blob object

- Test the Blob object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab.

Refer to *5. Vision Sequences* for more details on how to create a new vision sequence or select one that was previously defined.

Step 1: Create a New Blob Object

(1) Click on the <New Blob   > toolbar button on the Vision Guide 7.0 toolbar. (In this case click means to click and release the left mouse button)

(2) Move the mouse over the image display. You will see the mouse pointer change to the Blob icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "Blob01" because this is the first Blob object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a Blob object similar to the one shown below:

New Blob Object Layout
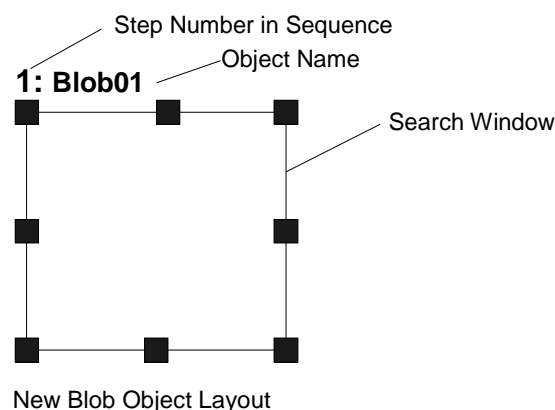
(1) Click on the name label of the Blob object and, while holding the mouse down, drag the Blob object to the position where you would like the top left position of the search window to reside.

(2) Resize the Blob object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.) (The search window is the area within which we will search for Blobs.)

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Step 3: Configure the Blob Object Properties

We can now set property values for the Blob object. Shown below are some of the more commonly used properties that are specific to the Blob object. Explanations for other properties such as AbortSeqOnFail, Graphics, etc. which are used on many of the different vision objects can be seen in the Vision Guide 7.0 Properties and Results Reference Manual or in the *BlobObject Properties* list earlier in this chapter.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Name property	The default name given to a newly created Blob object is "Blobxx" where xx is a number which is used to distinguish between multiple Blob objects within the same vision sequence. If this is the first Blob object for this vision sequence then the default name will be "Blob01". To change the name, click on the Value field of the Name property and type a new name and hit return. You will notice that every place where the Blob object's name is displayed is changed to reflect the new name.

Polarity property	Select either one of the following in Polarity property:
	  - detect a dark object on a light background (DarkOnLight)
	  - detect a light object on a dark background (LightOnDark)
	
	The default setting is DarkOnLight (a dark object on a light background). If you want to change it, click on the Value field of the Polarity property and you will see a drop down list with 2 choices: DarkOnLight or LightOnDark. Click on the choice you want to use.

MinArea, MaxArea	These properties define the area limit for a Blob object to be considered "Found". (i.e. the Found result returned as True) The default range is set as 25-100,000 (MinArea, MaxArea) which is a very broad range. This means that most blobs will be reported as Found when you first run a new Blob object before adjusting the MinArea and MaxArea properties. Normally, you will want to modify these properties to reflect a reasonable range for the blob you are trying to find. This way if you find a blob which is outside of the range you will know it isn't the blob you wanted to find.
RejectOnEdge property	Allows you to exclude the parts touching the boundary of the search window. Normally, this should be set to True.

You can test the Blob object now and then come back and set the any other properties as required later.

Step 4: Test the Blob Object and Examine the Results

To run the Blob object, click on the <Run Object> button located at the bottom left of the object Tab. Results for the Blob object will now be displayed. The primary results to examine at this time are shown below. There are others that you will find useful in the future as well though.

Found result   - Returns whether or not the blob was found. If the blob that was found does not meet the area constraints defined by the MinArea and MaxArea properties then the Found result will return as False.

Area result   - The area (in pixels) of the blob found.

Angle result   - The angle at which the Blob is oriented. This is computed from the angle of the minor axis and will be a value between +/- 90 degrees.

Time result   - The amount of time it took for the Blob object to execute.

PixelX, PixelY   - The XY position (in pixels) of the center of mass of the found blob.

MinX, MinY, MaxX, MaxY

     - Combined these 4 values define the Extrema of the blob. (A rectangle which is formed by touching the outermost points of the blob.)

NOTE
☞
The RobotXYU, RobotX, RobotY, RobotU and CameraX, CameraY, CameraXYU results will return "no cal" at this time. This means that no calibration was performed so it is impossible for the vision system to calculate the coordinate results with respect to the Robot World or Camera World. Refer to *7. Vision Calibration* for more information.

Step 5: Make Adjustments to Properties and Test Again

After running the Blob object a few times, you may have encountered problems with finding a blob or just want to fine-tune some of the property settings. Some common problems and fine tuning techniques are described below:

Problems : If the Blob object returns a Found result of False, there are a few places to immediately examine.

- Value of the Polarity property may differ from the actual image. Check the value of the Polarity property, and make sure that the value matches the light and dark of the object you want to detect and its background. Also, it must match the light and dark of the object and its background displayed in the search window.

- Look at the Area result and compare this area with the values defined in the MinArea and MaxArea properties. If the Area result does not fall between the limits defined by the MinArea and MaxArea properties, then you may want to adjust these properties and run the Blob object again.

- Use Histograms to examine the distribution of gray-scale values in an image. The Histogram tool is excellent for setting the ThresholdHigh and ThresholdLow properties. Histograms are described in detail in *8. Histograms Tools*.

Fine Tuning : Fine-tuning of the Blob object may be required for some applications. The primary properties associated with fine-tuning of a Blob object are described below:

- MinArea, MaxArea - After you have run the Blob object a few times, you will become familiar with the approximate values returned for the Area result. Use these values when determining new values to enter to the MinArea and MaxArea properties. It is generally a good idea to have MinArea And MaxArea properties set to values which constrain the Found result such that only blobs which you are interested in are returned with the Found result equal to True. (This helps eliminate unwanted blobs that are different in area from the desired blob.)

- ThresholdHigh, ThresholdLow - These properties adjust parameters for the setting the gray levels thresholds for distinguishing between what is background and what is part of the blob. These properties are best set through using the Histogram tool. Please refer to the descriptions of the ThresholdHigh and ThresholdLow properties in the *Vision Properties and Results Reference Manual*. Histograms are described in detail in *8. Histograms Tools*.

Once you have completed making adjustments to properties and have tested the Blob until you are satisfied with the results you are finished with creating this vision object and can go on to creating other vision objects or configuring and testing an entire vision sequence.

Other Useful Utilities for Use with Blob Objects

At this point you may want to consider examining the Histogram feature of Vision Guide 7.0. Histograms are useful because they graphically represent the distribution of gray-scale values within the search window. The Vision Guide Histogram tool provides a useful mechanism for setting the gray levels for the ThresholdLow and ThresholdHigh properties which then define what is considered a part of the blob and what is considered part of the background. When you are having problems with finding a blob the Histogram feature is invaluable. The details regarding the Vision Guide Histogram usage are described in *8.1 Using Histograms*.

You may also want to use the Statistics feature of Vision Guide to examine the Blob object's results statistically. An explanation of the Vision Guide Statistics features are explained in the chapter *Histograms and Statistics Tools*.

Using Blob Objects as a Pixel Counter

The Blob object can be used as a pixel counter. A pixel counter counts all of the pixels in an image that fall within the blob thresholds.

Follow these steps:

(1) Create a Blob object.

(2) Set desired polarity.

(3) Set High and Low thresholds.

(4) Set NumberToFind to 0. This will cause the Blob object to find all blobs in the image.

(5) Set MinArea to 1 and MaxArea to 999999. Blobs of one pixel or more will be counted.

(6) Run the sequence.

Use the TotalArea result to read the total number of pixels that fall within the blob thresholds.

6.2.5 ⊣ Edge Object

Edge Object Description

The Edge object is used to locate edges in an image. The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels. The Edge object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and defines that position as the edge position for a single edge. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is returned as the result. The Edge object supports multiple results, so you can specify how many single edges or edge pairs you want to find.

An Edge object can be configured to search along a line or along an arc using the SearchType property.

Edge objects with SearchType = Line are similar to Line objects in shape with a search length. The search length of the Edge object is the length of the Edge object. One of the powerful features of the Edge object with SearchType = Line is its ability to be positioned at any angle. This allows a user to maintain an Edge object vector that is perpendicular to the region where you want to find an edge by changing the angle of the Edge object. Normally this is done by making the Edge object relative to a Frame that moves with the region you are interested in.

Edge Object Layouts

The Edge object has two different layouts.

Layout when SearchType is Line

When SearchType = Line, the Edge object's search window is the line along which the Edge object searches. The Edge object searches for a transition (light to dark or dark to light) somewhere along this line in the direction indicated by the Direction of Search Indicator.

Step Number in Sequence
Object Name
Size & Direction Handle

1: Edge01

Direction Indicator
(Direction of Search)
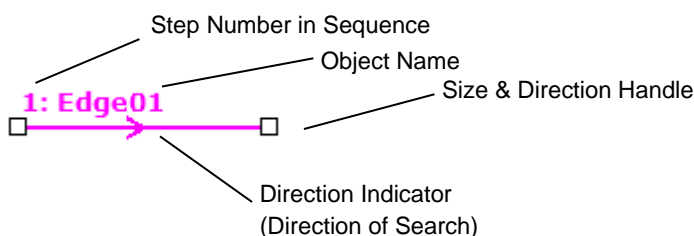
Edge Object Line Layout

The Edge object can be positioned to search in any direction (not just along the vertical and horizontal directions). This is done by using the size and direction handles of the Edge object to move either end of the Edge object along the direction and distance required to find the edge you are interested in. To move the entire object, drag the label or line.

Layout when SearchType is Arc

When SearchType is Arc, the Edge object's search window is the arc along which the Edge object searches. The Edge object searches for a transition (light to dark or dark to light) somewhere along this arc in the direction indicated by the Direction of Search Indicator.

Step Number in Sequence
Object Name
1: Edge01
Direction of Search Indicator
Size Handle
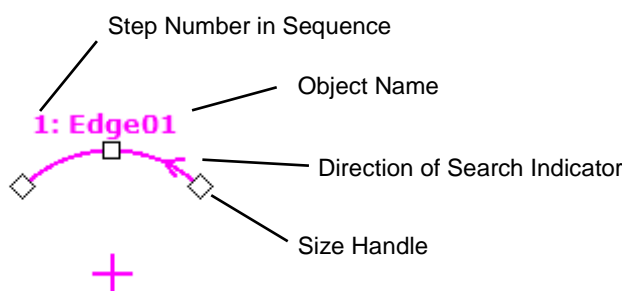
Edge Object Arc Layout

To change the size of the arc, drag one of the size handles on either end of the arc. To change the radius, drag the middle size handle. To move the entire object, drag the label or center point.

Edge Object Properties

The following list is a summary of properties for the Edge object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Note that when you change the SearchType property, the property set changes according to the specified type. When SearchType is Line, then the following properties are not visible in the Vision Guide window property grid:

AngleEnd

AngleStart

CenterPointObject

CenterPntObjResult

CenterPntOffsetX

CenterPntOffsetY

CenterPntRotOffset

When SearchType is Arc, then the following properties are not visible:

X1

Y1

X2

Y2

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. Default: 100
AngleEnd	Specifies the end angle of the range to perform a circular search Default: 135
AngleStart	Specified the start angle of the range to perform a circular search Default: 45
Caption	Used to assign a caption to the Edge object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the Edge Search. This property is normally set to base the center of the Edge Search on the XY position result from another vision object. Default: Screen

Property	Description
CenterPntObjResult	Specifies which result to use from the CenterPointObject.
	If All is specified, the Frame object will ve applied to all (NumberFound) of the specified vision object results
	Default: 1
CenterPntOffsetX	Specifies the X offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntOffsetY	Specifies the Y offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result.
	Default: False
ContrastTarget	Sets the desired contrast for the edge search.
	Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget.
	Default: 0
EdgeThreshold	Sets the threshold at which edges below this value are ignored.
	Default: 2
EdgeType	Select the type of edge to search for: single or pair.
	Default: 1 – Single
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color for an object when it is failed.
	Default: Red
Frame	Specifies which positioning frame to use.
	Default: none
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
Graphics	Specifies which graphics to display.
	Default: 1 – All
LabelBackColor	Selects the background color for an object label.
	Default: Transparent
Name	Used to assign a unique name to the Edge object.
	Default: Edge01
NumberToFind	Defines the number of edges to find.
	Default: 1

Property	Description
PassColor	Selects the color for an object when it is passed.
	Default: Light Green
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound
Polarity	Defines whether the Edge object should search for a LightToDark or DarkToLight transition.
	Default: 1 - LightToDark
SearchType	Sets whether to use Line or Arc search.
	Default: Line
ScoreWeightContrast	Sets the percentage of the score that depends on contrast.
	Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength.
	Default: 50
SearchWidth	Defines the width of the edge search.
	Range is from 3 - 99.
	Default: 3
StrengthTarget	Sets the desired edge strength to search for.
	Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget.
	Default: 0
X1	The X coordinate position of the start point of the edge.
X2	The X coordinate position of the end point of the edge.
Y1	The Y coordinate position of the start point of the edge.
Y2	The Y coordinate position of the end point of the edge.

Edge Object Results

The following list is a summary of the Edge object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
CameraX	Returns the X coordinate position of the found Edge's position in the camera coordinate system.
CameraY	Returns the Y coordinate position of the found Edge's position in the camera coordinate system.
CameraXYU	Runtime only. Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system.
Contrast	Returns the contrast of the found Edge.
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a score that is above the Accept property's current setting.)

Results	Description
NumberFound	Returns the number of Edges found. (This number can be anywhere from 1 up to the number of edges you requested the Edge object to find with the NumberToFind property.)
PassColor	Selects the color for passed objects.
	Default: LightGreen
PixelX	Returns the X coordinate position of the found Edge's position in pixels.
PixelY	Returns the Y coordinate position of the found Edge's position in pixels.
PixelXYU	Runtime only.
	Returns the PixelX, PixelY, and PixelU coordinates of the found part's position in pixels.
RobotX	Returns the X coordinate position of the found Edge's position with respect to the Robot's Coordinate System.
RobotY	Returns the Y coordinate position of the found Edge's position with respect to the Robot's Coordinate System
RobotXYU	Runtime only.
	Returns the RobotX, RobotY, and RobotU coordinates of the found Edge's position with respect to the Robot's World Coordinate System.
Score	Returns an integer value that represents the overall score of the found edge.
Strength	Returns the strength of the found edge.
Time	Returns the amount of time in milliseconds required to process the object.

Using Edge Objects

The next few sections guide you through how to create and use an Edge object.

- How to create a new Edge object
- Position and Size the search window
- Configure the properties associated with the Edge object
- Test the Edge object & examine the results
- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence ▢ toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New Edge Object

(1) Click on the <New Edge ⇥ > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the Edge object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "Edge01" because this is the first Edge object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the Edge Object

You should now see an Edge object similar to the one shown below:

New Edge Object

After creating the new Edge object, you can change whether to search along a line or an arc by setting the SearchType property. When the SearchType is Line (default), you can change the search length and rotation by clicking down on either size handle, and then dragging that end of the line to a new position. When the SearchType is Arc, you can change the arc by dragging either of the handles on each end of the arc. To change the radius, drag the middle handle.

You can also click on the name label of the Edge object or anywhere along the edge line and while holding the mouse down drag the entire Edge object to a new location on the screen. When you find the position you like, release the mouse and the Edge object will stay in this new position on the screen.

Step 3: Configuring Properties for the Edge Object

We can now set property values for the Edge object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the Edge object. Explanations for other properties such as AbortSeqOnFail, Graphics, etc. which are used on many of the different vision objects can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual*.

EdgeType (Single)   Select the type of the edge to be searched.

For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.

Name property ("Edgexx") The default name given to a newly created Edge object is "Edgexx" where xx is a number which is used to distinguish between multiple Edge objects within the same vision sequence. If this is the first Edge object for this vision sequence then the default name will be "Edge01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Edge object's name is displayed is updated to reflect the new name.

NumberToFind (1) You can search for 1 or more edges along the search line.

Polarity (LightToDark) If you are looking for a DarkToLight edge, change polarity.

Step 4: Running the Edge Object and Examining the Results

To run the Edge object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab. Results for the Edge object will now be displayed. The primary results to examine at this time are:

PixelX, PixelY results The XY position (in pixels) of the edge found along the edge search line.

CameraX, CameraY results These define the XY position of the Edge object in the Camera's World Coordinate system. The CameraX and CameraY results will only return a value if the camera has been calibrated. If it has not then "no cal" will be returned.

RobotX, RobotY results These define the XY position of the Edge object in Robot World Coordinates. The robot can be told to move to this XY position. (No other transformation or other steps are required.) The RobotX and RobotY results will only return a value if the camera has been calibrated. If it has not then "no cal" will be displayed.

6.2.6 Polar Object
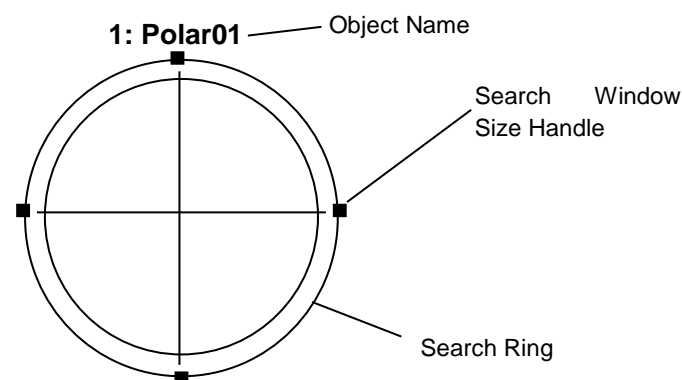
Polar Object Description

The Polar object provides a fast method of transforming an image having Cartesian coordinates to a corresponding image having polar coordinates. Polar objects are an excellent tool for determining object rotation. Because of the reliability and speed of the Polar object, we highly recommend using it when you need to calculate the angular rotation of an object.

The major characteristic to remember about Polar objects, is that they are highly insensitive to object rotation about their center, but intolerant of object translation in x-y space. This means that Polar objects are very good at calculating the rotation of an object, but only if that object's center position does not move. For this reason, Vision Guide 7.0 has a CenterPoint property for the Polar object. The CenterPoint property allows the user to lock the Polar object's center point onto the position result from another object such as the Correlation and Blob objects. This means that if you first find the XY position with a Correlation or Blob object, the Polar object can then be centered on the XY position and then used to calculate the rotation of the object.

Polar Object Layout

The Polar object layout is quite different looking than the other object layouts described so far. However, its usage is quite similar. The Polar object has a circular basic layout and as such has a center and radius. The position of the Polar object can be moved by clicking on the name of the Polar object (or anywhere on the circle that makes up the outer perimeter) and then dragging the object to a new position.

Polar Object Layout

NOTE
☞
The Polar object center position (defined by the CenterPoint property) can also be based upon the position of another object. This means that even though you may reposition a Polar object, once you run the object or Sequence the Polar object center position may change. For more details see the Polar object later in this chapter.

The search window for a Polar object is circular. It's outer boundary is defined by the outer ring shown as shown below. Its inner ring is a circle that is smaller in size than the outer ring and is located n pixels inside the outer ring. The number of pixels is defined by the Thickness property. As you change the Thickness property you will notice that the "thickness" or distance between the inner and outer ring changes. This provides a visual indicator for the area in which you are searching.

To resize the search window outer boundary for the Polar object, click on one of the 4 search window size handles and drag the ring inward or outward as desired.

Polar Object Properties

The following list is a summary of properties for the Polar object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Specify that the entire sequence to be aborted and no further objects to be processed if the specified object execution fails (i.e. not satisfied PassType). Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. If the value is small, it may result in false detection. Default: 700
AngleOffset	Defines an offset angle which is applied to the Polar object after teaching to adjust the angle indicator graphic with the feature you are searching for. Default: 0
Caption	Assigns a caption to the Geometric object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the Geometric Search. This property is normally set to base the center of the Geometric Search on the XY position result from another vision object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results. Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject. If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result. Default: False
CenterX	Specifies the X coordinate position to be used as the center point for the Polar Search Tool. This property is filled in automatically when the CenterPointObject property is set to another vision object.

Property	Description
CenterY	Specifies the Y coordinate position to be used as the center point for the Polar Search Tool. This property is filled in automatically when the CenterPointObject property is set to another vision object.
Confusion	Indicates the amount of confusion expected in the image to be searched. This is the highest shape score a feature can get that is not an instance of the feature for which you are searching. Default: 800
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color for an object when it is failed. Default: Red
Frame	Specifies which positioning frame to use. Default: none
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 - All
LabelBackColor	Selects the background color for an object label. Default: Transparent
ModelObject	Determines which model to use for searching. Default: Self
Name	Used to assign a unique name to the Polar object. Default: Polar01
PassColor	Selects the color for an object when it is passed. Default: Light Green
PassType	Selects the rule that determines if the object passed. Default:SomeFound
Radius	Defines the distance from the CenterPoint of the Polar object to the outer most search ring of the object. Default: 50
ShowModel	Allows the user to see the internal grayscale representation of a taught model.
Thickness	Defines the thickness of the search ring which is the area searched for the Polar object. The thickness is measured in pixels starting from the outer ring defined by the Radius of the Polar object. Default: 5

Polar Object Results

The following list is a summary of the Polar object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual.*

Results	Description
Angle	Returns the amount of part rotation in degrees.
CameraX	Returns the X coordinate position of the Polar object's CenterPoint position in the camera coordinate system.
CameraY	Returns the Y coordinate position of the Polar object's CenterPoint position in the camera coordinate system.
CameraXYU	Runtime only. Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system. Values are in millimeters.
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
PassColor	Selects the color for passed objects. Default: LightGreen
PixelY	Returns the Y coordinate position of the found part's position (referenced by model origin) in pixels.
RobotX	Returns the X coordinate position of the Polar object's CenterPoint position with respect to the Robot's Coordinate System.
RobotY	Returns the Y coordinate position of the Polar object's CenterPoint position with respect to the Robot's Coordinate System.
RobotU	Returns the U coordinate position of the Polar object's Angle result with respect to the Robot's Coordinate System.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the found part's position with respect to the Robot's World Coordinate System.
Score	Returns an integer value that represents the level at which the feature found at runtime matches the model for which Polar object is searching.
Time	Returns the amount of time in milliseconds required to process the object.

NOTE

☞ All results for the Polar object which return X and Y position coordinates (CameraX, CameraY, PixelX, PixelY, RobotX, RobotY....) receive those coordinate values from the CenterPoint property. Whatever the CenterPoint property is set to before running the Polar object is then passed through as results for the Polar object. The Polar object does not calculate an X and Y center position. However, the X and Y position results are provided to make it easy to get X, Y and U results all from the Polar object rather than getting the X and Y results from one vision object and then the U results from the Polar object.

Understanding Polar Objects

The purpose of Polar objects is to find the amount of rotation of a specific object or pattern. This is done by first teaching a polar model which is basically a circular ring with a specified thickness. After teaching we can then run the Polar object and the circular ring is compared against the current rotation of the part we are interested in and an angular displacement is calculated and returned as one of the results for Polar objects.

Let's take a look at an example to help make it easier to see. Consider a part that rotates around a center point and needs to be picked up and placed on a pallet by a robot. Since this part is always oriented in a different position, it is difficult to pick up unless we can determine the rotation of the object. If this object looked something like the part shown in black & gray in Figure A, we could define a Polar Model which intersects the outer part of the object. Notice the area within the inner and outer rings. This area defines the model of the Polar object that was taught for this part. You should be able to see that some areas within the ring are very distinctive because they have strong feature differences from the rest of the ring. These areas will be key in finding the amount of rotation the part will have.

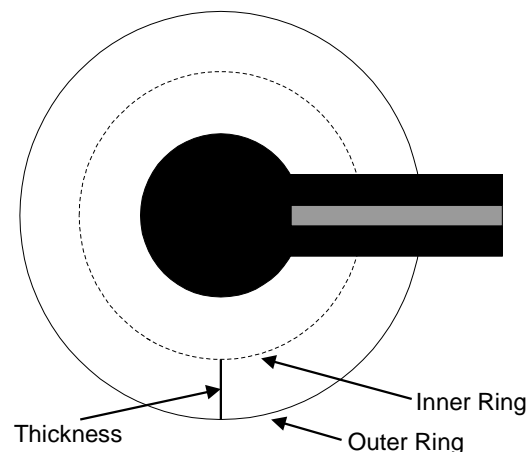Figure A: Example part for use with Polar Object

Figure B shows what the taught model looks like. It is just a ring that is mostly white with one section of black and gray. When the Polar object is run it will search the search window for this type of pattern where one section is black and gray while the rest of the ring is white.
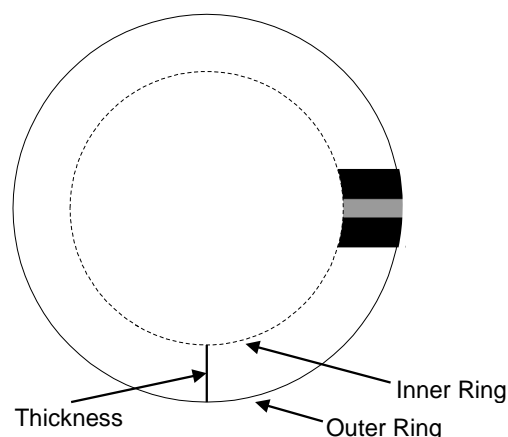
Figure B: The Polar Model representation of the Part from Figure A

In Figure C we show the original part which has rotated about 90 degrees as compared to the model. The Polar object will calculate the angular displacement between the model and the new position of the object.
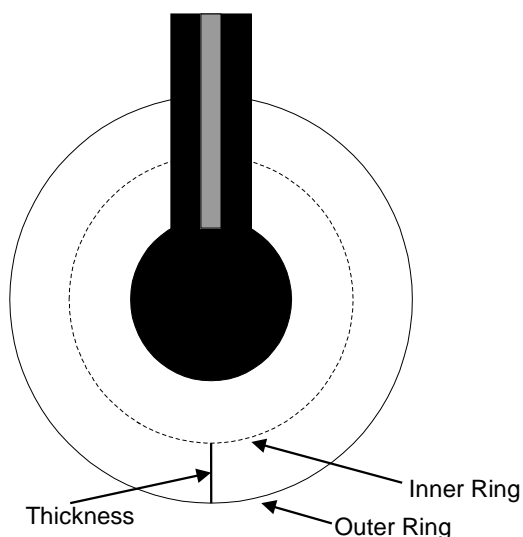
Figure C: Part rotated 90 degrees from original taught model

Figure D shows the original Model taught at 0 degrees on the left and the part that was rotated 90 degrees on the right.
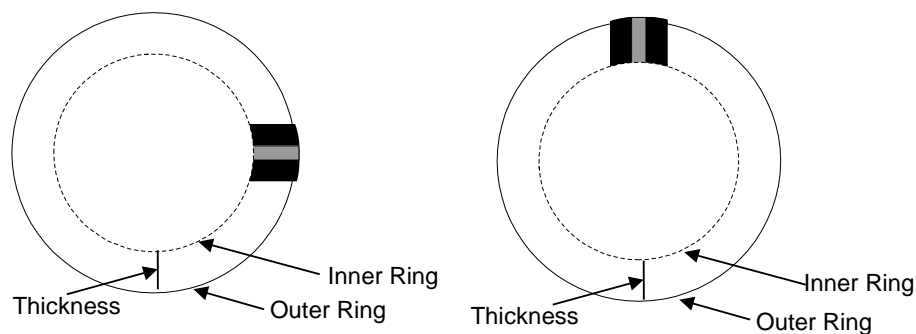
Figure D: Original Polar Model and rotated part's Polar representation

The Primary Parameters Associated with Polar Objects

As you can probably derive from looking at the Figures in the last section, there are 3 primary parameters associated with Polar objects that are each important to achieve the best results possible. These parameters are defined by the following properties:

- CenterPoint property

- Radius property

- Thickness property

- AngleOffset property

The CenterPoint property defines the center position of the Polar object. As mentioned earlier, the center point of the Polar object must be aligned properly so that the center point of the taught model and the center point of the part you are searching for an angular shift within are perfectly aligned. Otherwise, the translation in XY space will cause the angular results to be inaccurate.

The Radius property defines the distance from the center of the Polar object to the outermost ring of the Polar object. This defines the outer Search Area boundary.

The Thickness property defines distance (in pixel units) from the outer ring to the imaginary inner ring. This is in effect the thickness of the Search Area.

The AngleOffset property provides a mechanism for the user to set the angular value for the graphics line that is used to display the rotational position for Polar objects. This line can be seen at 3 O'clock (the 0 degrees position) after running a Polar object. But since you may want a graphical indicator as to the rotational position of the feature you are most interested in, the Angle Offset property allows you to reposition this graphical indicator according to your needs. (Remember that this AngleOffset property is normally set after Teaching a Model and running the Polar object.)

Determining Object Rotation

A typical application requiring object rotation determination involves an integrated circuit die to be picked up by a robot that must know the die's XY position and its rotational orientation. The die is a different shade of gray from the background, and the surface of the die contains gray level information that indicates its rotation from 0 to 37.0 degrees.

For this application, a Blob object (called "Blob01") is used to find the center of the die. (Alternatively, you could also have used a Correlation object.) A Polar object is created using the XY position result from the Blob object as the center of the Polar object. (This is done by setting the Polar object's CenterPoint property to "Blob01".) The proper Radius and Thickness property values are then set for the Polar object. A gray level Polar Model is then trained providing a model of the die at 0 degrees.

When searching for a new die at a different orientation, a Polar search window is constructed from the XY position result from the "Blob01" Blob object, and the Radius and Thickness properties. This search window is then searched for the rotational equivalent of the model that was previously taught. The angle at which the model is found is then returned as the Angle result (in Pixels) and RobotU result (in Robot World coordinates).

NOTE

☞

While the RobotU result returned will be the actual rotation of the part in the Robot's coordinate system, remember that your gripper was probably not mounted with exactly 0 degrees of rotation. It is probably off by at least a few degrees. So make sure to adjust for this in your program when moving to the part.

Adjusting the Taught Model's Angle Offset

When a Polar Model is taught, the original circular model is considered as 0 degrees where the 0-degree position is at 3 o'clock. The figure below shows a model which is taught where the area of interest is at about 1 o'clock. When this model is taught, the model's 0 degrees position will be at 3 o'clock as with all Polar Models. However, since we want to see a visual indication of the part's actual rotation we must use an Angle Offset to properly adjust the positioning of the polar angle indicator. For example, our part which was taught pointing at about 1 o'clock requires an Angle Offset of about 60 degrees. (The AngleOffset property for the Polar object should be set to 60 degrees.)
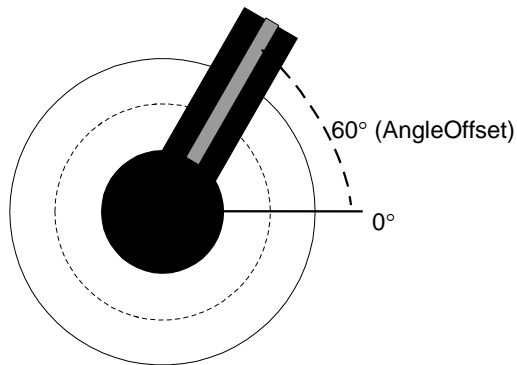
Polar Model where AngleOffset property is set to 60°

Performance Issues for Polar Objects

When teaching Polar objects, the primary concern is that the Polar Model contain enough information to determine the object's orientation. Therefore, the desired accuracy and execution speed of the Polar object determine the size of the Polar Model used for the Polar object search.

If the Polar object's rotational result is to be accurate to 1/2 a degree, then the Polar object need be only 180 pixels wide (2 degrees per pixel or 0.5 degrees per Polar object resolution unit, assuming quarter pixel searching accuracy).

Accuracy of the Polar object is also dependent on the gray level information in the Model.

The thickness of the Polar Model (in pixels) is also chosen to contain enough information that the rotation signature yields reliable results even when there is minor variation in the location of the center point of the image you are searching. Choosing a Thickness property setting of 1 pixel would mean that if the image were misplaced by one pixel, the polar transformation would send completely different pixels to the polar image. To provide some tolerance in the source image location, a Thickness property of 5 is used so that if the source image location is off by one pixel, the corresponding polar image pixel would only be one fifth off. (This is why the minimum value for the Thickness property is 5.)

The choice of the Polar object's Radius and Thickness properties should be based on the amount of gray information in the Model, and on the desired searching speed. Searching speed is proportional to the Radius and Thickness properties. The fastest Search times will result in a small Radius property setting with the Thickness property set to 5. However, in many cases a Thickness property set to 5 may not be enough to accurately find the Polar Model.

Using Polar Objects

The next few sections guide you through how to create and use a Polar object.

- How to create a new Polar object

- Position and Size the search window

- Configure the properties associated with the Polar object

- Test the Polar object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence ☐ > toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New Polar Object

(1) Click on the <New Polar 🕐 > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the Polar object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "Polar01" because this is the first Polar object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the Polar Object

You should now see a Polar object similar to the one shown below:
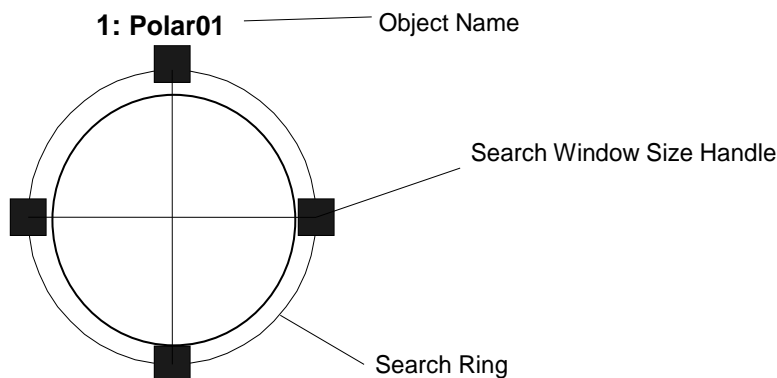
New PolarObject

Polar objects have a circular search window. You can change the position of the entire object or change its radius.

To move the entire object, click on the object name or anywhere along the outer circumference and while holding the left mouse button down drag the entire object to a new location on the screen. When you find the position you like, release the mouse and the Polar object will stay in this new position on the screen.

To change the radius, move the mouse pointer over one of the size handles, press the left mouse button down, then move the mouse to change the size of the radius.

Step 3: Configuring Properties for the Polar Object

We can now set property values for the Polar object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the Polar object. Explanations for other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual* or in *Polar Object Properties* previously described in this section.

Name property   - The default name given to a newly created Polar object is "Polarxx" where xx is a number which is used to distinguish between multiple Line objects within the same vision sequence. If this is the first Line object for this vision sequence then the default name will be "Polar01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Polar object's name is displayed is updated to reflect the new name.

     Vision Guide 7.0 (Ver.7.4) Software Rev.1

CenterPointObject property - Typically you will set this property to one of the objects that occur previously in the sequence. This will determine the center point of the Polar object at runtime.

Thickness property - Typically you will set this property a value large enough to contain sufficient model information to locate the angle of the part.

AngleOffset property - Typically you will set this property to position the final angle result at the desired position. For example, if you were looking for the minute hand of a watch, you would adjust AngleOffset so that the displayed angle matched the minute hand.

Step 4: Running the Polar Object and Examining the Results

To run the Polar object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab. The CenterPointObject will be run first.

Results for the Polar object will now be displayed. The primary results to examine at this time are:

Angle result - The angle of the model found in degrees.

6.2.7 OCR OCR Object

OCR Object Description

The OCR object (Optical Character Recognition) is used to recognize single line character strings in an image for a specific font and character size. The OCR object GUI includes a Font wizard that is used to create a font based on SEMI standards, or create a user defined font based on characters in an image or an ASCII description file. You can export the fonts you create to disk and import them into other OCR objects in the same project or other projects.

NOTE
☞
OCR objects will only work if the OCR option is installed and enabled.

OCR Object Layout

The OCR object has a search window and a model window, as shown below.
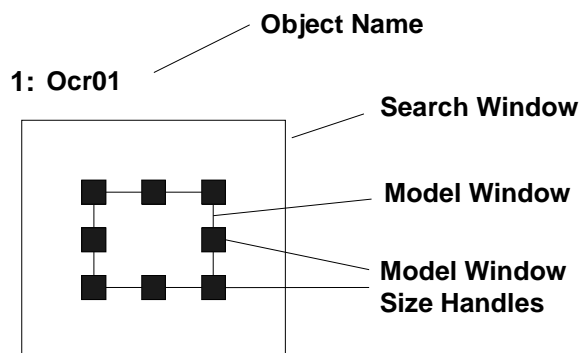
OCR Object Layout

OCR Object Properties

The following list is a summary of properties for the OCR object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Caption	Assigns a caption to the Geometric object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the OCR Search. This property is normally set to base the center of the OCR Search on the XY position result from another vision object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results. Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0

Property	Description
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	Default: False
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one result.
	Grayed out when FindChar=False
	Default: 1
Dictionarymode	Specifies the dictionary mode
	Default: All
Enabled	Specifies whether to execute the object.
	Default: True
ExportFont	Exports the current font to disk.
FailColor	Selects the color of an object when it is failed.
	Default: Red
FindChar	Specifies whether to treat each character as an individual object
	Default: False
Frame	Specifies which positioning frame to use.
	Default: None
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
Graphics	Specifies which graphics to display.
	Default: 1 – All
ImportFont	Runs a file dialog from the Vision Guide GUI that allows you to import a font file.
InvalidChar	Sets / returns the character used in the Text result to represent an invalid character.
	Default: "?"
LabelBackColor	Selects the background color for an object label.
	Default: Transparent
ModelWin	Runtime only.
	Sets / returns the model window left, top, height, width parameters in one call.
ModelWinLeft	Defines the left most position of the model window.
ModelWinHeight	Defines the height of the model window.
	Default: 50
ModelWinTop	Defines the top most position of the model window.

Property	Description
ModelWinWidth	Defines the width of the model window.
	Default: 50
Name	Used to assign a unique name to the OCR object.
	Default: Ocrxx
PassColor	Selects the color for an object when it is passed.
	Default: Light Green
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound
Polarity	Defines the difference of the object and background (dark object on the bright background or bright object on the dark background).
	Default: 1 - DarkOnLight
PassType	Selects the acceptance condition for object detection.
SearchWin	Runtime only.
	Sets / returns the search window left, top, height, width parameters in one call.
SearchWinHeight	Defines the height of the area to be searched in pixels.
	Default: 100
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinWidth	Defines the width of the area to be searched in pixels.
	Default: 100

OCR Object Results

The following list is a summary of the OCR object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Results	Description
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a score which is above the Accept property's current setting.)
Passed	Returns whether or not the object detection result was accepted.
CameraX	Returns the X coordinate position of the found character in the camera coordinate system.
CameraY	Returns the Y coordinate position of the found character in the camera coordinate system.
NumberFound	Returns the number of characters found within the search window.

Results	Description
MaxX	Returns the maximum X pixel coordinate of the character Extrema in pixels.
MaxY	Returns the maximum Y pixel coordinate of the character Extrema in pixels.
MinX	Returns the minimum X pixel coordinate of the character Extrema in pixels.
MinY	Returns the minimum Y pixel coordinate of the character Extrema in pixels.
PixelX	Returns the X coordinate position of the found character's position in pixels.
PixelY	Returns the Y coordinate position of the found character's position in pixels.
RobotX	Returns the X coordinate position of the found character in the robot coordinate system.
RobotY	Returns the Y coordinate position of the found character in the robot coordinate system.
ShowAllResults	Displays a dialog showing all results of the specified vision object in a table form. This makes it easy to compare results.
Text	Returns the text found in a search operation.
Time	Returns the amount of time in milliseconds required to process the object.

Using OCR Objects

This next section will describe the steps required to use OCR objects as listed below:

- Create a new OCR object

- Create or import a font for the new object.

- Calibrate the font (if required).

- Configure properties associated with the OCR object

- Test the OCR object and examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence ☐ toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab.

Creating a New OCR Object

1. Click on the New OCR toolbar button on the Vision Guide Toolbar. (In this case click means to click and release the left mouse button).

2. You will see an OCR icon appear above the OCR object toolbar button.

3. Click on the OCR icon and drag to the image display of the Vision Guide window.

4. Notice that a name for the object is automatically created. In the example, it is called "Ocr01" because this is the first OCR object created for this sequence. (We will explain how to change the name later.)

**Step Number
in Sequence**

Object Name

1: Ocr01

Search Window

Model Window

OCR Object Layout
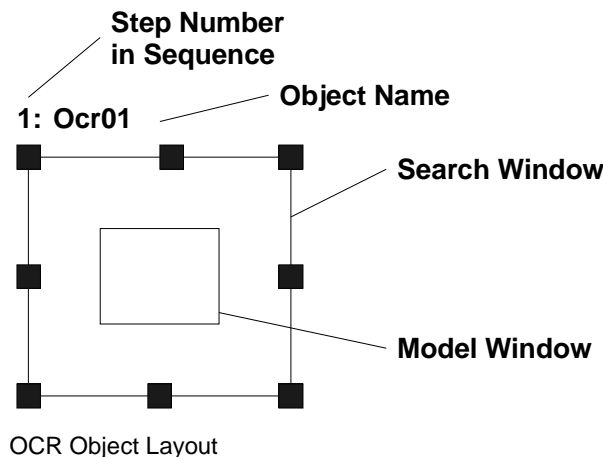
Create a font

To use OCR objects, a font need to be created according to the image to be scanned. You can import a font from other projects by using the ImportFont property. When you import a font, you don't need to create a font.
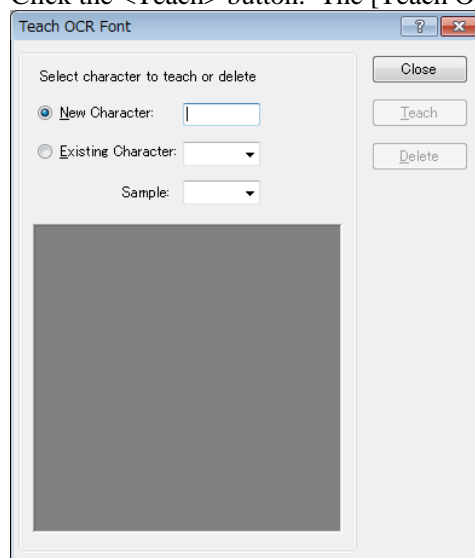
Also, common Western and Japanese fonts are embedded in the system. A font may need to be created depending on the size of the input image.

A user-defined font can be created from the image file.

Follow these steps to create a font:

(1) Click the Object tab. Adjust the model window to the character.

(2) Click the <Teach> button. The [Teach OCR Font] dialog will be displayed.

(3) To register a new character, click the <New Character> button. Then, click the
<Teach> button. You can register multiple images to one character.
You can register Japanese and alphanumeric characters, and symbols.
To delete the registered character from the font, click the <Existing Character>
button and select the character to be deleted from the dropdown list. Then, click the
<Delete> button.

6.2.8 CodeReader Object

CodeReader Object Description

The CodeReader object is used to read bar codes or 2 dimensional codes.

Codes supported are:

EAN13	Digits 0 - 9, fixed length.
Code39	0 - 9, A - Z, ./+-%$Spc, variable length.
DataMatrix	2 dimensional code *1
Interleaved 2 of 5	Digits 0 - 9, fixed length.
Code128	Full ASCII, variable length.
Codabar	Digits 0-9, $, -, :, /, ., +.
PDF417	2 dimensional code
QR	2 dimensional code
EAN	8 Digits 0 - 9, fixed length
UPC A	Digits 0 - 9, fixed length.
UPC E	Digits 0 - 9, fixed length

*1: Supports only ECC200 codes.

CoderReader Object Layout

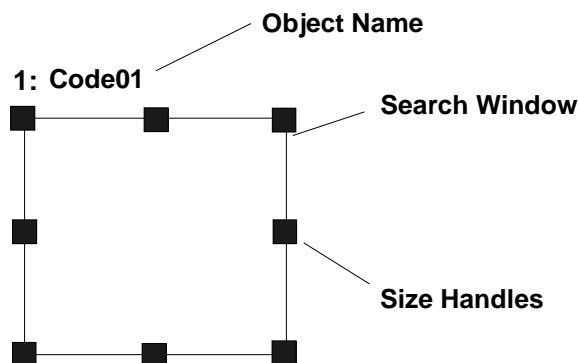The CodeReader object only has a search window, as shown below.

Figure 1: CodeReader Object Layout

CodeReader Object Properties

The following list is a summary of properties for the CodeReader object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed.
	Default: 0 - False
Caption	Used to assign a caption to the CodeReader object.
	Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the CodeReader Search. This property is normally set to base the center of the CodeReader Search on the XY position result from another vision object.
	Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject.
	If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results.
	Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0

Property	Description
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	Default: False
CodeType	Sets / returns which type of bar code or two-dimensional code to search for with the CodeReader object.
	Default: 0 – Auto
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color of an object when it is not accepted.
	Default: Red
Frame	Defines the current object searching position with respect to the specified frame.
	Default: none
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
Graphics	Specifies which graphics to display.
	Default: 1 – All
LabelBackColor	Sets the background color for the object label.
	Default: Transparent
Name	Used to assign a unique name to the CodeReader object.
	Default: Code01
NumberToFind	Defines the number of codes to find.
	Max: 8
	Default: 1
PassColor	Selects the color for an object when it is passed.
	Default: LightGreen
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound
Polarity	Defines the differentiation between objects and background. (Either "Dark Object on Light Background" or "Light Image on Dark Background").
	Default: 1 – DarkOnLight
SearchWin	Runtime only. Sets / returns the search window left, top, height, width parameters in one call.
SearchWinHeight	Defines the height of the area to be searched in pixels.
	Default: 100

Property	Description
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinWidth	Defines the width of the area to be searched in pixels. Default: 100

CodeReader Object Results

The following list is a summary of the CodeReader object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Results	Description
Angle	Returns the angle for inclination of the detected code.
CameraX	Returns the X coordinate of the detected object in the camera coordinate system. (Unit: mm)
CameraY	Returns the Y coordinate of the detected object in the camera coordinate system. (Unit: mm)
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a score which is above the Accept property's current setting.)
FoundCodeType	Returns the detected code type.
NumberFound	Returns the detected code.
Passed	Returns whether or not the object detection result was accepted.
PixelX	Returns the X coordinate of the detected object in pixels.
PixelY	Returns the Y coordinate of the detected object in pixels.
RobotX	Returns the X coordinate of the detected object in the robot coordinate system.
RobotY	Returns the Y coordinate of the detected object in the robot coordinate system.
ShowAllResults	Displays a dialog showing all results of the specified vision object in a table form.
Text	Returns the text found in a search operation.
Time	Returns the amount of time in milliseconds required to process the object.

Using CodeReader Objects

This next section will describe the steps required to use CodeReader objects as listed below:

- Create a new CodeReader object

- Position and Size the search window

- Configure properties associated with the CodeReader object

- Test the CodeReader object and examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab.

Step 1: Create a New CodeReader Object

1. Click on the New CodeReader toolbar button on the Vision Guide Toolbar. (In this case click means to click and release the left mouse button).

2. You will see a CodeReader icon appear above the CodeReader object toolbar button.

3. Click on the CodeReader icon and drag to the image display of the Vision Guide window.

4. Notice that a name for the object is automatically created. In the example, it is called "Code01" because this is the first CodeReader object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a CodeReader object similar to the one shown below:
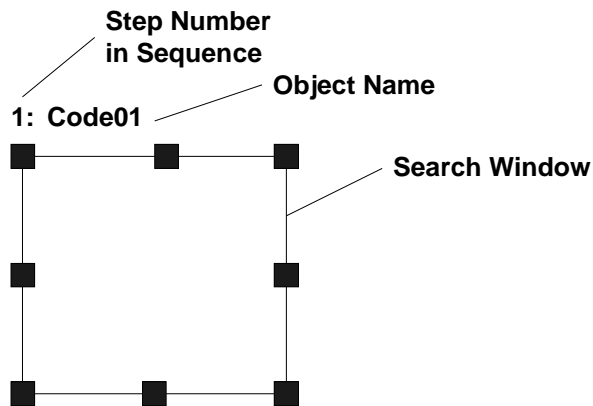

Figure 2: New CodeReader Object

1. Click on the name label of the CodeReader object and while holding the mouse down drag the CodeReader object to the position where you would like the top left position of the search window to reside.

2. Resize the CodeReader object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.) (The search window is the area within which we will search.)

NOTE
☞

It is important to allow space on both sides of a bar code (known as the quiet zone) or the search will fail. One cell or more blank space is necessary around it if it is a two demontional bar code.

6.2.9 ColorMatch Object

ColorMatch Object Description

The ColorMatch object is used to detect colors that match one or more color models.

ColorMatch Object Layout

The ColorMatch object has a rectangle, rotated rectangle, or circular layout and as such has a center point and radius. The position of the ColorMatch object can be moved by clicking on the name of the object or anywhere on the outer perimeter of the search window and then dragging the object to a new position.

ColorMatch Object Layout

NOTE

☞ The ColorMatch object center position (defined by the CenterPoint property) can also be based upon the position of another object. This means that even though you may reposition a ColorMatch object, once you run the object or Sequence the object center position may change.

The search area for a ColorMatch object can be a rectangle, rotated rectangle, or circle.

To resize the search window outer boundary for the ColorMatch object where the SearchWinType is Circle, click on one of the search window size handles and drag the ring inward or outward as desired to change the radius.

ColorMatch Properties

The following list is a summary of properties for the ColorMatch object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the score that a feature must equal or exceed to be considered found. If the value is small, it may result in false detection. Default: 700
Caption	Used to assign a caption to the ColorMatch object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point for the ColorMatch Search. This property is normally set to base the center of the ColorMatch Search on the XY position result from another vision object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. Default: 1
CenterPntOffsetX	Specifies the X offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject. If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result. Default: False

Property	Description
CenterX	Specifies the X coordinate position to be used as the center point for the object. This property is filled in automatically when the CenterPoint property is set to another vision object.
CenterY	Specifies the Y coordinate position to be used as the center point for the ColorMatch object. This property is filled in automatically when the CenterPoint property is set to another vision object.
ColorMode	Sets which color space (RGB/HSV) to use. Default: RGB
CurrentModel	Runtime only. This specifies which model to use for the ModelColor property and also for VTeach. CurrentModel values are 1 to the NumberOfModels. Default: 1
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one result. Default: 1
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color for an object when it is failed. Default: Red
Frame	Defines the current object searching position with respect to the specified frame. (Allows the object to be positioned with respect to a frame.) Default: None
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 – All
LabelBackColor	Sets the background color for the object label. Default: Transparent
ModelColor	Runtime only. This property is used at runtime to teach a model manually by setting the RGB color value directly. Default: RGB(0, 0, 0)
ModelColorTol	Runtime only. This property is used at runtime to set the color tolerance for a model color. If a pixel color is within the tolerance of a model color, then the pixel is unchanged. Default: 0 (ColorMode = RGB), 0,0,50 (ColorMode = HSV)
ModelName	Runtime only. This property is used at runtime to set the name of the current model.

Property	Description
ModelObject	Determines which model to use for searching.
	Default: Self
Name	Used to assign a unique name to the object.
	Default: ColorMatchxx
NumberOfModels	Runtime only.
	This is the number of color models used. At runtime, you can set the NumberOfModels, then use CurrentModel and VTeach to teach each color model.
	Default: 1
NumberToFind	Defines the number of features to find in the current search window.
	Default: 1
PassColor	Selects the color for an object when it is passed.
	Default: Light Green
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound
Radius	Defines the radius of the object circle.
	Default: 50
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched in pixels.
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinType	Defines the type of the area to be searched (i.e. Rectangle, RotatedRectangle, Circle).
SearchWinWidth	Defines the width of the area to be searched (the search window width)

ColorMatch Results

The following list is a summary of the ColorMatch object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Results	Description
CameraX	Returns the X coordinate position of the found part's position (referenced by model origin) in the camera coordinate system. Values are in millimeters.
CameraY	Returns the Y coordinate position of the found part's position (referenced by model origin) in the camera coordinate system. Values are in millimeters.
ColorIndex	Returns the index of the found color model.
ColorName	Returns the name of the found color model.
ColorValue	Returns the RGB or HSV value of the found color, depending on the ColorMode setting.
CameraXYU	Runtime only. Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system. Values are in millimeters.
Found	Returns whether or not a color was matched from one of the color models.
NumberFound	Returns the number of objects found.
PixelX	Returns the X coordinate position of the found parts position in pixels.
PixelY	Returns the Y coordinate position of the found parts position in pixels.
RobotX	Returns the X coordinate of the detected object in the robot coordinate system.
RobotY	Returns the Y coordinate of the detected object in the robot coordinate system.
RobotU	Returns the U coordinate of the detected object in the robot coordinate system.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the detected object in the robot coordinate system.
Score	Returns an INTEGER value from 0 to 1000 representing the degree to which the color of the model matches the object detected
ShowAllResults	Displays a dialog that allows you to see all results for a specified vision object in a table form. This makes it easy to compare results when the NumberToFind property is set to 0 or greater than 1.
Time	Returns the amount of time in milliseconds required to process the object.

Using ColorMatch Objects

Now that we've reviewed how ColorMatch and searching works we have set the foundation for understanding how to use Vision Guide ColorMatch objects. This next section will describe the steps required to use ColorMatch objects as listed below:
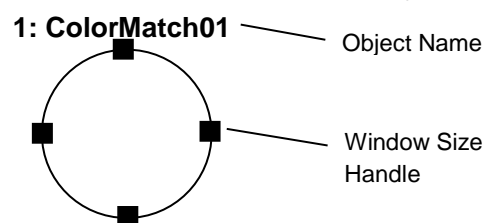
- Create a new ColorMatch object

- Position and size the search window

- Configure properties associated with the ColorMatch object

- Teach one or more color models

- Test the ColorMatch object and examine the results

- Make adjustments to properties and test again

- Working with Multiple Results from a single ColorMatch object

Step 1: Create a new ColorMatch object

(1) Click on the New ColorMatch toolbar button [icon] on the Vision Guide Toolbar. (In this case click means to click and release the left mouse button)

(2) The mouse cursor will change to a ColorMatch icon.

(3) Move the mouse cursor over the image display of the Vision Guide window and click the left mouse button to place the ColorMatch object on the image display

(4) Notice that a name for the object is automatically created. In the example, it is called "ColorMatch01" because this is the first ColorMatch object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a ColorMatch object similar to the one shown below:



New ColorMatch object layout

(1) Click on the name label of the ColorMatch object and while holding the mouse down drag the object to the position where you would like the search window to reside.

(2) Resize the ColorMatch object window as required using the window size handles. (This means click on a size handle and drag the mouse.) The window is the area within which we will match colors.

Step 3: Teach models for the ColorMatch object

Before you can use the ColorMatch object to detect colors, you must first teach one or more color models. When a color model is taught, the average color is determined from all of the pixels inside the ColorMatch object window. You can name each model.

(1) Make sure that the ColorMatch object is the currently activated object in the Object tab. You can look at the current object drop down list at the top of the Object tab to see which object is the object you are currently working on or you can look at the image display and see which object is highlighted in magenta.

(2) Click on the Teach button at the bottom right of the Object tab. A window will be opened as shown below:
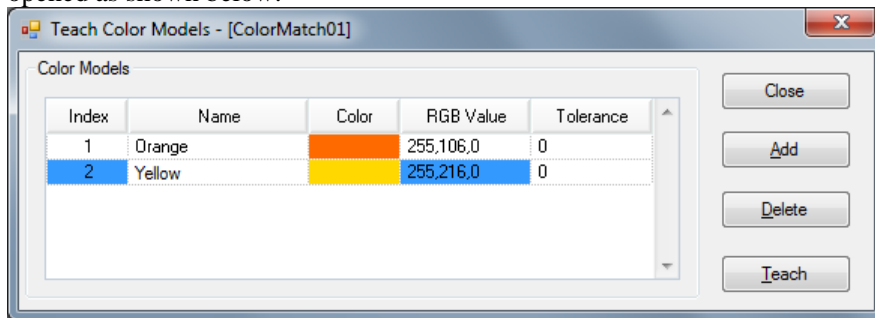
While the teach window is displayed, you can change the position of the ColorMatch object as needed to teach each color model.

(3) Position the ColorMatch object over the color you want to teach. Try to fill the entire window with the color.

(4) Click the Add button to add a new model.

(5) Select the model you want to teach by clicking in any field of the desired model's row.

(6) Click the Teach button to teach the color.

(7) Enter a meaningful name for the color. This name is used as the ColorName result.

(8) The Tolerance default value when ColorMode is RGB is 0, and the tolerance default values when ColorMode is HSV are 0, 0, 50. You can change the tolerance values to aid in matching colors with small variations or where lighting is not consistent.

(9) To add more models, repeat steps 3 – 8.

NOTE
☞
Although in most cases you will teach the color model using the ColorMatch object window, you can also enter the RBG (or HSV) values manually in the teach window.

6.2.10 ⊠ LineFinder Object

LineFinder Object Description

LineFinder objects are used to identify the position of a line in the image.

LineFinder objects process multiple Edge objects automatically to identify the edge position and obtain the line identified from each edge position.

The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels.

The LineFinder object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and defines that position as the edge position for a single edge. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is retured as the result.

LineFinder Object Layout

The LineFinder object has a different look than the Correlation and Blob objects. The LineFinder object's search window is the line along which the Edge object searches. The LineFinder object searches for a transition (light to dark or dark to light) somewhere along this line in the direction indicated by the Direction Indicator.
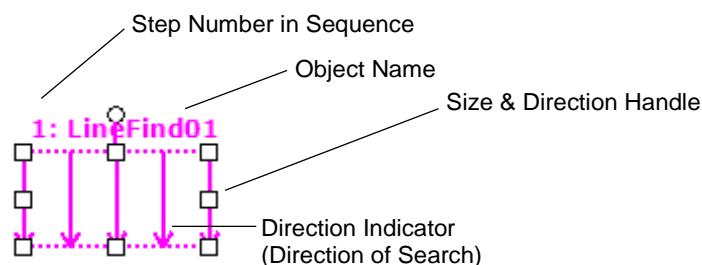
LineFinder Object Layout

The LineFinder object can be positioned to search in any direction (not just along the vertical and horizontal directions). This is done by using the size and direction handles of the LineFinder object to move the LineFinder object along the direction (and per a user-specified distance) required to find the edge you are interested in.

LineFinder Object Properties

The following list is a summary of properties for the LineFinder object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False

Property	Description
Accept	Specifies the shape score that a feature must equal or exceed to be considered found.
	Default: 100
Caption	Used to assign a caption to the Edge object.
	Default: Empty String
ContrastTarget	Sets the desired contrast for the edge search.
	Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget.
	Default: 0
EdgeThreshold	Sets the threshold at which edges below this value are ignored.
	Default: 2
EdgeType	Select the type of edge to search for: single or pair.
	Default: 1 – Single
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color for an object when it is failed.
	Default: Red
Frame	Specifies which positioning frame to use.
	Default: none
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
Graphics	Specifies which graphics to display.
	Default: 1 – All
LabelBackColor	Selects the background color for an object label.
	Default: Transparent
Name	Used to assign a unique name to the Edge object.
	Default: Edge01
NumberOfEdges	Defines the number of edges to find.
	Default: 5
PassColor	Selects the color for an object when it is passed.
	Default: Light Green
PassType	Selects the rule that determines if the object passed.
	Default: SomeFound
Polarity	Defines whether the LineFinder object should search for a LightToDark or DarkToLight transition.
	Default: 1 - LightToDark
ScoreWeightContrast	Sets the percentage of the score that depends on contrast.
	Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength.
	Default: 50

Property	Description
SearchWidth	Defines the width of the edge search. Range is from 3 - 99. Default: 3
StrengthTarget	Sets the desired edge strength to search for. Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget. Default: 0
X1	The X coordinate position of the start point of the edge.
X2	The X coordinate position of the end point of the edge.
Y1	The Y coordinate position of the start point of the edge.
Y2	The Y coordinate position of the end point of the edge.

LineFinder Object Results

The following list is a summary of the Edge object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
Angle	Returns the detection edge angle in the image coordinate system.
CameraX1	Returns the X coordinate for the start point of the detected edge line in the camera coordinate system.
CameraY1	Returns the Y coordinate for the start point of the detected edge line in the camera coordinate system.
CameraX2	Returns the X coordinate for the end point of the detected edge line in the camera coordinate system.
CameraY2	Returns the Y coordinate for the end point of the detected edge line in the camera coordinate system.
Contrast	Returns the average contrast of the detected edges.
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
Length	Returns the length of the detected edge line in millimeters.
PixelLength	Returns the length of the detected edge line in pixels.
PixelX1	Returns the X coordinate position of the start point of the detected edge line in the image coordinate system.
PixelY1	Returns the Y coordinate position of the start point of the detected edge line in the image coordinate system.
PixelX2	Returns the X coordinate position of the end point of the detected edge line in the image coordinate system.
PixelY2	Returns the Y coordinate position of the end point of the detected edge line in the image coordinate system.
RobotX1	Returns the X coordinate position of the start point of the detected edge line in the Robot coordinate system.

Results	Description
RobotY1	Returns the Y coordinate position of the start point of the detected edge line in the Robot coordinate system.
RobotX2	Returns the X coordinate position of the end point of the detected edge line in the Robot coordinate system.
RobotY2	Returns the Y coordinate position of the end point of the detected edge line in the Robot coordinate system.
RobotU	Returns the angle of the detected edge line in the Robot coordinate system.
Strength	Returns the average strength of the detected edges.
Time	Returns the amount of time in milliseconds required to process the object.

Using LineFinder Objects

The next few sections guide you through how to create and use a LineFinder object.

- How to create a new LineFinder object

- Position and Size the search window

- Configure the properties associated with the LineFinder object

- Test the LineFinder object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New LineFinder Object

(1) Click on the <New LineFinder ⊠> toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the LineFinder object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "LineFind01" because this is the first LineFinder object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the LineFinder Object

You should now see a LineFinder object similar to the one shown below:
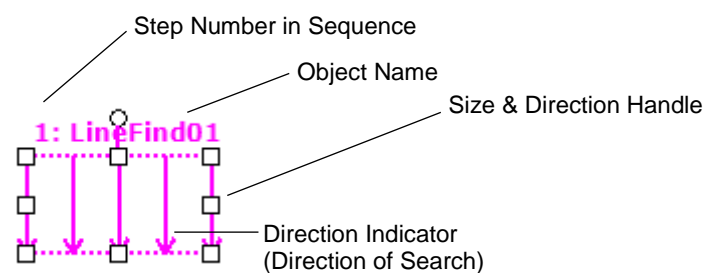
New LineFinder Object

1. Click on the name label of the LineFinder object and, while holding the mouse down, drag the LineFinder object to the position where you would like the top left position of the search window to reside.

2. Resize the LineFinder object search window as required using the search window size handles.

Step 3: Configuring Properties for the LineFinder Object

We can now set property values for the LineFinder object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the LineFinder object. Explanations for other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects, can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual*. Use the AngleMode property to set the angle output format. For more details, refer to *Line Object*.

EdgeType (Single)   Select the type of the edge to be searched.
For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.

Name property ("LineFindxx")	The default name given to a newly created LineFinder object is "LineFindxx" where xx is a number which is used to distinguish between multiple LineFinder objects within the same vision sequence. If this is the first LineFinder object for this vision sequence then the default name will be "LineFind01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the LineFinder object's name is displayed is updated to reflect the new name.
NumberOfEdges (1)	You can search for 1 or more edges along the search line.
Polarity (LightToDark)	If you are looking for a DarkToLight edge, change polarity.

Step 4: Running the LineFinder Object and Examining the Results

To run the LineFinder object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab.
Results for the LineFinder object will now be displayed. The primary results to examine at this time are:

Angle Result	Returns the angle of the detected edge line in the Image coordinate system.
Length Result	Returns the length of the detected edge line in the Camera coordinate system. Unit: mm
PixelLength Result	Returns the length of the detected edge line in the Image coordinate system. Unit: pixel
PixelX1 Result **PixelY1 Result**	Returns the XY coordinate position for the start point of the detected edge line in the Image coordinate system.
PixelX2 Result **PixelY2 Result**	Returns the XY coordinate position for the end point of the detected edge line in the Image coordinate system.
CameraX1 Result **CameraY1 Result**	Returns the XY coordinate position for the start point of the detected edge line in the Camera coordinate system. If the calibration is not performed, "no cal" will be returned.
CameraX2 Result **CameraY2 Result**	Returns the XY coordinate position for the end point of the detected edge line in the Camera coordinate system. If the calibration is not performed, "no cal" will be returned.
RobotX1 Result **RobotY1 Result**	Returns the XY coordinate position for the start point of the detected edge line in the Robot coordinate system. If the calibration is not performed, "no cal" will be returned.

RobotX2 Result
RobotY2 Result

Returns the XY coordinate position for the end point of the detected edge line in the Robot coordinate system.

If the calibration is not performed, "no cal" will be returned.

RobotU Result

Returns the angle of the detected edge line in the Robot coordinate system.

6.2.11 ⬚ LineInspector Object

LineInspector Object Description

LineInspector objects are used to inspect a line in the image.

LineInspctor objects process multiple Edge objects automatically to identify the defects in the line being inspected.

The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels.

The LineInspector object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and defines that position as the edge position for a single edge. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is returned as the result.

LineInspector Object Layout

The LineInspector object looks similar to the LineFinder tool. The LineInspector object's search window contains several edge search lines. The LineInspector object searches for a transition (light to dark or dark to light) somewhere along each search line in the direction indicated by the direction indicators. The data from the edge searches is used to determine defects along the line.
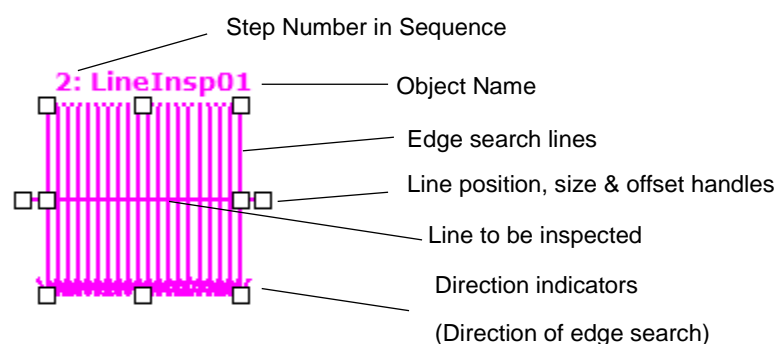

LineInspector Object Layout

The LineInspector object can be positioned and rotated to search in any direction (not just along the vertical and horizontal directions).

To rotate, drag one of the outer size handles in a clockwise or counter-clockwise direction.

The LineFinder result can be used for the line to be inspected. In this case, set the LineFInder to be specified to LineObject.

4: LineInsp01

Outer size handle (X1, Y1)

Outer size handle (X2, Y2)
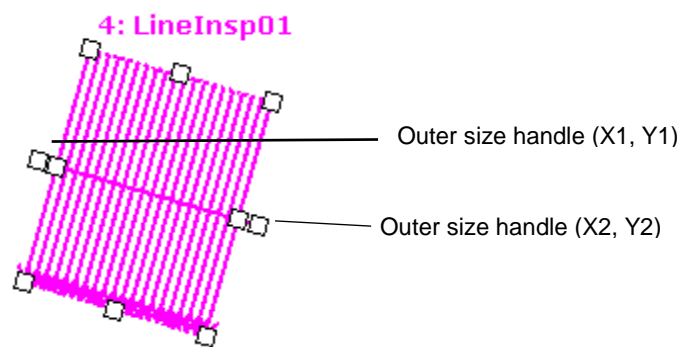
To change the width of the line to be inspected, drag one of the outer size handles away from or toward the center of the line.
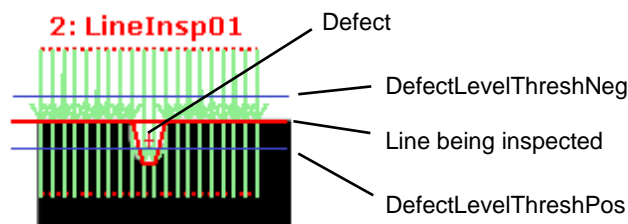
To change the size of the edge searches, drag one of the inner size handles.

LineInspector Search

The image below shows part of an object with a defect.

The LineInspector finds the defect as shown below. Note that each edge search position distance from the line being inspected must exceed either the DefectLevelThreshPos or DefectLevelThreshNeg property values for a defect to be found. Also, the defect area must be greater than MinArea and less than MaxArea.

2: LineInsp01   Defect

DefectLevelThreshNeg

Line being inspected

DefectLevelThreshPos

LineInspector Object Properties

The following list is a summary of properties for the LineInspector object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the score that an edge must equal or exceed to be considered found. Default: 100

Property	Description
Caption	Used to assign a caption to the Edge object. Default: Empty String
ContrastTarget	Sets the desired contrast for the edge search. Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget. Default: 0
DefectAreaExtended	Sets whether to extend the defect area interpolation. Default: False
DefectLevelThreshNeg	Sets defect threshold below the line. Default: 2
DefectLevelThreshPos	Sets defect threshold above the line. Default: 2
EdgeThreshold	Sets the threshold at which edges below this value are ignored. Default: 2
EdgeType	Select the type of edge to search for: single or pair. Default: 1 – Single
Enabled	Specifies whether to execute the object. Default: True
EndPntObjResult	Specifies which result to use from the EndPointObject. Default: 1
EndPointObject	Specifies which vision object to use to define the end point of the line to be inspected. Default: Screen
EndPointType	Defines the type of end point used to define the end point of a line. Default: 0 – Point
FailColor	Selects the color for an object when it is failed. Default: Red
Frame	Specifies which positioning frame to use. Default: none
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 – All
InspectEndOffset	Sets the offset from the end of the line where the inspection should stop. Default: 15
InspectStartOffset	Sets the offset from the start of the line where the inspection should start. Default: 15

Property	Description
LabelBackColor	Selects the background color for an object label. Default: Transparent
LineObject	Defines an object that will search for the line before it is inspected. Default: None
LineObjResult	Defines which result to use from the LineObject. Default: 1
MaxArea	Defines the upper Area limit for a defect. For a defect to be found it must have an Area result below the value set for MaxArea property. Default: 100,000
MinArea	Defines the lower Area limit for a defect. For a defect to be found it must have an Area result above the value set for the MinArea property. Default: 25
MinMaxArea	Runtime only. Sets or returns both MinArea and MaxArea in one statement.
MissingEdgeType	Defines how to handle a missing edge. Default: Interpolate
Name	Used to assign a unique name to the LineInspector object. Default: LineInsp01
NumberOfEdges	Defines the number of edges to use for the inspection. Default: 20
NumberToFind	Defines the number of defects to find in the search area. Default: 1
PassColor	Selects the color for an object when it is passed. Default: Light Green
PassType	Selects the rule that determines if the object passed. Default: AllNotFound
Polarity	Defines whether the LineInspector object should search for a LightToDark or DarkToLight transition. Default: 1 - LightToDark
ScoreWeightContrast	Sets the percentage of the score that depends on contrast. Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength. Default: 50
SearchWidth	Defines the width of the edge search. Range is from 3 - 99. Default: 3
SizeToFind	Selects which size of defects to find. Default: 1 - Largest
StartPntObjResult	Specifies which result to use from the StartPointObject.

Vision Guide 7.0 (Ver.7.4) Software Rev.1

Property	Description
	Default: 1
StartPointObject	Specifies which vision object to use to define the start point of the Line.
	Default: Screen
StartPointType	Defines the type of start point used to define the start point of a line.
	Default: 0 – Point
StrengthTarget	Sets the desired edge strength to search for.
	Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget.
	Default: 0
X1	The X coordinate position of the start point of the edge.
X2	The X coordinate position of the end point of the edge.
Y1	The Y coordinate position of the start point of the edge.
Y2	The Y coordinate position of the end point of the edge.

LineInspector Object Results

The following list is a summary of the LineInspector object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
Area	Returns the area of the defect in pixels.
CameraX	Returns the X coordinate of the defect in the Camera coordinate system.
CameraY	Returns the Y coordinate of the defect in the Camera coordinate system.
Contrast	Returns the average contrast of the detected edges.
DefectLevel	Returns the level of the defect.
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
Length	Returns the length of the defect in millimeters.
PixelLength	Returns the length of the defect in pixels.
NumberFound	Returns the number of defects found.
PixelX	Returns the X coordinate position of the defect.
PixelY	Returns the Y coordinate position of the defect.
RobotX	Returns the X coordinate position of the defect in the Robot coordinate system.
RobotY	Returns the Y coordinate position of the defect in the Robot coordinate system.

Strength	Returns the average strength of the detected edges.
Time	Returns the amount of time in milliseconds required to process the object.
TotalArea	Returns the total area of all defects found in pixels.

Using LineInspector Objects

The next few sections guide you through how to create and use a LineInspector object.

- How to create a new LineInspector object

- Position and Size the search window

- Configure the properties associated with the LineInspector object

- Test the LineInspector object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence [icon] toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.
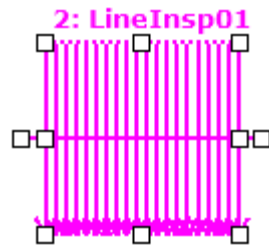
Step 1: Create a New LineInspector Object

(1) Click on the <New LineInspector [icon] > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the LineInspector object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "LineInsp01" because this is the first LineInspector object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the LineInspector Object

You should now see a LineInspector object similar to the one shown below:

New LineInspector Object

1. Click on the name label of the LineInspector object and, while holding the mouse down, drag the LineInspector object to the position where you would like the top left position of the search window to reside.

2. Resize the LineInspector object search window as required using the search window size handles.

Step 3: Configuring Properties for the LineInspector Object

We can now set property values for the LineInspector object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the LineInspector object. Explanations for other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects, can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual*.

EdgeType (Single)	Select the type of the edge to be searched. For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.
Name property ("LineInspxx")	The default name given to a newly created LineInspector object is "LineInspxx" where xx is a number which is used to distinguish between multiple LineInspector objects within the same vision sequence. If this is the first LineInspector object for this vision sequence then the default name will be "LineInsp01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the LineInspector object's name is displayed is updated to reflect the new name.
NumberToFind (1)	You can search for 1 or more defects along the search line.
Polarity (LightToDark)	If you are looking for a DarkToLight edge, change polarity.

Step 4: Running the LineInspector Object and Examining the Results

To run the LineInspector object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab.

Results for the LineInspector object will now be displayed. The primary results to examine at this time are:

Area result   The area (in pixels) of the defect found.

PixelX Result   Returns the XY coordinates of the defect in the image coordinate
PixelY Result   system.

CameraX Result   Returns the XY coordinates of the defect in the Camera
CameraY Result   coordinate system.

                         If the calibration is not performed, "no cal" will be returned.

RobotX Result   Returns the XY coordinates of the defect in the Robot coordinate
RobotY Result   system.

                         If the calibration is not performed, "no cal" will be returned.

6.2.12 ArcFinder Object

ArcFinder Object Description

ArcFinder objects are used to identify the position of an arc of circle / ellipse in the image.

To find the arc of circle / ellipse, a series of edge searches are executed to determine radius and center point of the arc, and major/minor axes and angle of the ellipse.

The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels.

Each edge search of the ArcFinder object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and defines that position as the edge position for a single edge. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is returned as the result.

The type of edges (circle/ellipse) can be specified by the ArcSearchType property.

ArcFinder Object Layout

The ArcFinder object has a different look than the Correlation and Blob objects. The ArcFinder object's search window is circular and is defined by a start angle, end angle, outer radius, and inner radius. The edge search lines are evenly spanned between the start angle and end angle. The number of edge search lines is specified with the NumberOfEdges property. The Direction property can specify search from the inner radius to the outer radius, or vice versa.
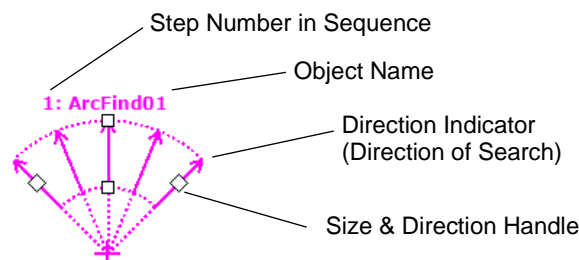
ArcFinder Object Layout

The ArcFinder is positioned to search for an arc by moving the center point to the approximate center of the arc to be found, and then adjusting the RadiusInner and RadiusOuter properties so that the arc to be found is somewhere within the search area.

ArcFinder Object Properties

The following list is a summary of properties for the ArcFinder object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. Default: 100
AngleEnd	Specified the end angle of the range to perform an arc of circle/ellipse search Default: 135
AngleStart	Specified the start angle of the range to perform an arc of circle/ellipse search Default: 45
ArcSearchType	Specifies the type of edges (circle/ellipse) to be searched for.
Caption	Used to assign a caption to the Edge object. Default: Empty String

Property	Description
CenterPointObject	Specifies the position to be used as the center point of the object.
	If the property is set to "Screen", the object can be placed to anywhere in the screen. If other vision objects are specified, the center point will be set to the PixelX and PixelY results of the object.
	Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject.
	Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result.
	Default: False
CenterX	Specifies the X coordinate position to be used as the center point for the object. This property is filled in automatically when the CenterPointObjet property is set to another vision object.
CenterY	Specifies the Y coordinate position to be used as the center point for the object. This property is filled in automatically when the CenterPointObjet property is set to another vision object.
ContrastTarget	Sets the desired contrast for the edge search.
	Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget.
	Default: 0
Direction	Sets the direction for the edge search.
	Default: InsideOut
EdgeThreshold	Sets the threshold at which edges below this value are ignored.
	Default: 2
EdgeType	Select the type of edge to search for: single or pair.
	Default: 1 – Single
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color of an object when it is not accepted.
	Default: Red
Frame	Specifies which positioning frame to use.
	Default: none

Property	Description
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 – All
LabelBackColor	Selects the background color for an object label. Default: Transparent
Name	Used to assign a unique name to the Edge object. Default: Edge01
NumberOfEdges	Specified the number of edges to be detected. Default: 5
PassColor	Selects the color for an object when it is passed. Default: LightGreen
PassType	Selects the rule that determines if the object passed. Default:SomeFound
Polarity	Defines whether the ArcFinder object should search for a LightToDark or DarkToLight transition. Default: 1 - LightToDark
ScoreWeightContrast	Sets the percentage of the score that depends on contrast. Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength. Default: 50
SearchWidth	Defines the width of the edge search. Range is from 3 - 99. Default: 3
ShowExtension	Defines whether or not to display the edge line with the both ends extended.
StrengthTarget	Sets the desired edge strength to search for. Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget. Default: 0

ArcFinder Object Results

The following list is a summary of the Edge object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
Angle	Returns the angle of the detected ellipse.
Angle1	Returns the start point angle of the detected edge in the Image coordinate system.
Angle2	Returns the end point angle of the detected edge in the Image coordinate system.
CameraX	Returns the central X coordinate of the detected circular/elliptic edge in the Camera coordinate system.
CameraY	Returns the central Y coordinate of the detected circular/elliptic edge in the Camera coordinate system.
Contrast	Returns the contrast of the detected circular/elliptic edge.
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
FoundMajorDiam	Length of the major axis of the detected elliptic edge.
FoundMinorDiam	Length of the minor axis of the detected elliptic edge.
FoundRadius	Radius of the detected circular edge.
MaxError	Returns the maximum difference from the detected circular/elliptic edge in pixel length.
Passed	Returns whether or not the object detection result was accepted.
PixelX	Returns the X coordinate at the center of the detected circular /elliptic edge in the image coordinate system.
PixelY	Returns the Y coordinate at the center of the detected circular /elliptic edge in the image coordinate system.
RobotX	Returns the X coordinate at the center of the detected circular /elliptic edge in the Robot coordinate system.
RobotY	Returns the Y coordinate at the center of the detected circular /elliptic edge in the Robot coordinate system.
Strength	Returns the strength of the detected edge.
Time	Returns the amount of time in milliseconds required to process the object.

Using ArcFinder Objects

The next few sections guide you through how to create and use an ArcFinder object.

- How to create a new ArcFinder object

- Position and Size the search window

- Configure the properties associated with the ArcFinder object

- Test the ArcFinder object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence [ ] toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New ArcFinder Object

(1) Click on the <New ArcFinder [ ] > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the ArcFinder object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "ArcFind01" because this is the first ArcFinder object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the ArcFinder Object

You should now see an ArcFinder object similar to the one shown below:
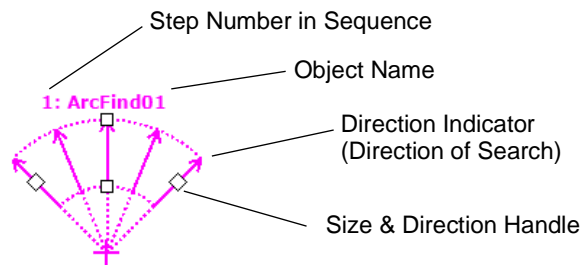
New ArcFinderObject

1. Click on the name label of the ArcFinder object and, while holding the mouse down, drag the ArcFinder object to the position where you would like the top left position of the search window to reside.

2. Resize the ArcFinder object search window as required using the search window size handles.

Step 3: Configuring Properties for the ArcFinder Object

We can now set property values for the ArcFinder object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the ArcFinder object. Explanations for other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects, can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual*.

EdgeType (Single)	Select the type of the edge to be searched. For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.
Name property ("ArcFindxx")	The default name given to a newly created ArcFinder object is "ArcFindxx" where xx is a number which is used to distinguish between multiple ArcFinder objects within the same vision sequence. If this is the first ArcFinder object for this vision sequence then the default name will be "ArcFind01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the ArcFinder object's name is displayed is updated to reflect the new name.
NumberOfEdges(5)	You can search for five edges to find circular edges.
Polarity (LightToDark)	If you are looking for a DarkToLight edge, change polarity.

Step 4: Running the ArcFinder Object and Examining the Results

To run the ArcFinder object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab. Results for the ArcFinder object will now be displayed. The primary results to examine at this time are:

Angle1 Result Returns the start point angle of the detected circular edge in the Image coordinate system.

Angle2 Result Returns the end point angle of the detected circular edge in the Image coordinate system.

FoundRadius Result Returns the radius of the detected circular edge in the pixel length.

Unit: pixel

MaxError Result Returns the maximum difference from the detected circular edge in pixel length.

PixelX Result Returns the XY coordinates at the center of the detected circular
PixelY Result edge in the image coordinate system.

CameraX Result Returns the central XY coordinates of the detection circular edge
CameraY Result in the Camera coordinate system.

If the calibration is not performed, "no cal" will be returned.

RobotX Result Returns the central XY coordinates of the detection circular edge
RobotY Result in the Robot coordinate system.

If the calibration is not performed, "no cal" will be returned.

6.2.13 ArcInspector Object

ArcInspector Object Description

ArcInspector objects are used to search for defects along an arc of circle/ellipse.

To find defects, a series of edge searches are executed to determine abnormalities in the arc of circle/ellipse being inspected.

The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels.

Each edge search of the ArcInspector object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and defines that position as the edge position for a single edge. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is retured as the result.

ArcInspector Object Layout

The ArcInspector object looks similar to the ArcFinder object. It searches for edges along lines from the center of the arc of circle/ellipse to the outer radius of the inspection area. Each edge search line searches for a transition (light to dark or dark to light) somewhere along this line in the direction indicated by the Direction of Search Indicator. The number of edges uses for inspection is set by the NumberOfEdges property. The AngleStart property specifies the start angle of the arc to be inspected. The AngleEnd property specifies the end angle of the arc to be inspected. The edge searches are spanned evenly between AngleStart + InspectStartOffset and AngleEnd – InspectEndOffset.
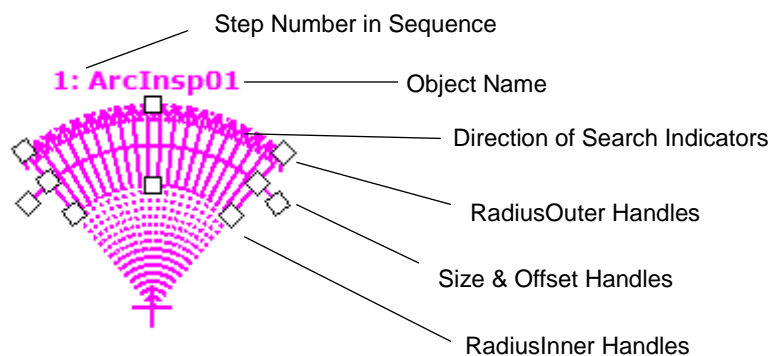
ArcInspector Object Layout

The ArcInspector is positioned to search for defects on an arc by aligning CenterX and CenterY to the center of the arc, and then adjusting RadiusInner and RadiusOuter to position the search area so that the radius of the arc to be inspected is within the search area. The ArcInspector object can search for defects from the inner radius to the outer radius (default), or vice versa, depending on the Direction setting.
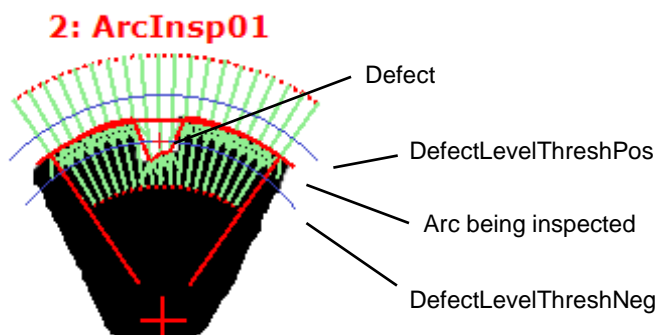
You can also use an ArcFinder object to first find an arc, and then inspect that arc with ArcInspector. Use the ArcObject property to specify the ArcFinder to use.

ArcInspector Search

The image below shows part of a round object with a defect.

The ArcInspector finds the defect as shown below. Note that each edge search position distance from the arc being inspected must exceed either the DefectLevelThreshPos or DefectLevelThreshNeg property values for a defect to be found. Also, the defect area must be greater then MinArea and less then MaxArea.

ArcInspector Object Properties

The following list is a summary of properties for the ArcInspector object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the score that an edge must equal or exceed to be considered found. Default: 100
AngleEnd	Specified the end angle of the range to perform a circular/elliptic search Default: 135
AngleStart	Specified the start angle of the range to perform a circular/elliptic search Default: 45
ArcObject	Specifies an ArcFinder object that will search for the arc of circle/ellipse to be inspectede. Default: None
ArcObjResult	Defines which result to use from the ArcObject. Default: 1

Property	Description
ArcSearchType	Specifies the type of edges (circle/ellipse) to be searched for.
	When specifying the ArcObject property, match this property with ArcType of the specified ArcFinder.
Caption	Used to assign a caption to use for the ArcInspector object label.
	Default: Empty String
CenterPointObject	Specifies the position to be used as the center point of the object.
	If the property is set to "Screen", the object can be placed to anywhere in the screen. If a vision objects is specified, the center point will be set to the PixelX and PixelY results of the specified center point object.
	Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject.
	Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result.
	Default: False
CenterX	Specifies the X coordinate position to be used as the center point for the object. This property is filled in automatically when the CenterPointObjet property is set to another vision object.
CenterY	Specifies the Y coordinate position to be used as the center point for the object. This property is filled in automatically when the CenterPointObjet property is set to another vision object.
ContrastTarget	Sets the desired contrast for the edge search.
	Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget.
	Default: 0
DefectAreaExtended	Sets whether to extend the defect area interpolation.
	Default: False

Property	Description
DefectLevelThreshNeg	Sets defect threshold below the line. Default: 2
DefectLevelThreshPos	Sets defect threshold above the line. Default: 2
Direction	Sets the direction for the edge search. Default: InsideOut
EdgeThreshold	Sets the threshold at which edges below this value are ignored. Default: 2
EdgeType	Select the type of edge to search for: single or pair. Default: 1 – Single
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color of an object when it is not accepted. Default: Red
Frame	Specifies which positioning frame to use. Default: none
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 – All
InspectStartOffset	Sets the offset from the start of the arc where the inspection should start. Default: 5
InspectEndOffset	Sets the offset from the end of the arc where the inspection should stop. Default: 5
LabelBackColor	Selects the background color for an object label. Default: Transparent
MaxArea	Defines the upper Area limit for a defect. For a defect to be found it must have an Area result below the value set for MaxArea property. Default: 100,000
MinArea	Defines the lower Area limit for a defect. For a defect to be found it must have an Area result above the value set for MinArea property. Default: 25
MinMaxArea	Runtime only. Sets or returns both MinArea and MaxArea in one statement.
MissingEdgeType	Defines how to handle a missing edge. Default: Interpolate
Name	Used to assign a unique name to the ArcInspector object. Default: ArcInsp01

Property	Description
NumberOfEdges	Specified the number of edges to be detected. Default: 20
NumberToFind	Defines the number of defects to find in the search area. Default: 1
PassColor	Selects the color for an object when it is passed. Default: LightGreen
PassType	Selects the rule that determines if the object passed. Default:AllNotFound
Polarity	Defines whether the ArcInspector object should search for a LightToDark or DarkToLight transition. Default: 1 - LightToDark
ScoreWeightContrast	Sets the percentage of the score that depends on contrast. Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength. Default: 50
SearchWidth	Defines the width of the edge search. Range is from 3 - 99. Default: 3
SizeToFind	Selects which size of defects to find. Default: 1 - Largest
StrengthTarget	Sets the desired edge strength to search for. Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget. Default: 0

ArcInspector Object Results

The following list is a summary of the ArcInspector object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
Area	Returns the area of the defect in pixels.
CameraX	Returns the X coordinate of the defect in the Camera coordinate system.
CameraY	Returns the Y coordinate of the defect in the Camera coordinate system.
Contrast	Returns the average contrast of the detected circular edges.
DefectLevel	Returns the level of the defect.
Length	Returns the length of the defect in millimeters.
PixelLength	Returns the length of the defect in pixels.
NumberFound	Returns the number of defects found.

Results	Description
Passed	Returns whether or not the object detection result was accepted.
PixelX	Returns the X coordinate of the defect in the image coordinate system.
PixelY	Returns the Y coordinate of the defect in the image coordinate system.
RobotX	Returns the X coordinate of the defect in the Robot coordinate system.
RobotY	Returns the Y coordinate of the defect in the Robot coordinate system.
Strength	Returns the average strength of the detected edges.
Time	Returns the amount of time in milliseconds required to process the object.
TotalArea	Returns the sum of areas of all defects found in pixels.

Using ArcInspector Objects

The next few sections guide you through how to create and use an ArcInspector object.

- How to create a new ArcInspector object

- Position and Size the search window

- Configure the properties associated with the ArcInspector object

- Test the ArcInspector object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New ArcInspector Object

(1) Click on the <New ArcInspector ⟨icon⟩ > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the ArcInspector object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "ArcInsp01" because this is the first ArcInspector object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the ArcInspector Object

You should now see an ArcInspector object similar to the one shown below:
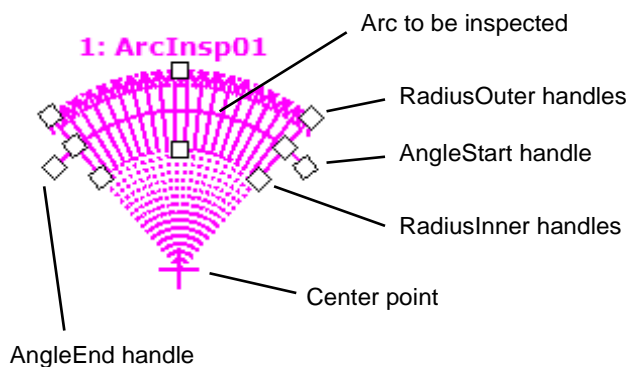
New ArcInspectorObject

1. Click on the name label of the ArcInspector object and, while holding the mouse down, drag the ArcInspector object to position the center point to be near the center of the arc to be inspected.

2. Resize the ArcInspector object search window as required using the RadiusOuter, RadiusInner, AngleStart, and AngleEnd size handles.

Step 3: Configuring Properties for the ArcInspector Object

We can now set property values for the ArcInspector object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the ArcInspector object. Explanations for other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects, can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual*.

EdgeType (Single)   Select the type of the edge to be searched.
        For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.

Name property ("ArcInspxx")	The default name given to a newly created ArcInspector object is "ArcInspxx" where xx is a number which is used to distinguish between multiple ArcInspector objects within the same vision sequence. If this is the first ArcInspector object for this vision sequence then the default name will be ""ArcInsp01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the ArcInspector object's name is displayed is updated to reflect the new name.
NumberOfEdges	Specify how many edge searches to be used to find defects. The default is 15, and the maximum is 99.
Polarity	The default edge search polarity is LightToDark. If you are looking for a DarkToLight edges, change the polarity.
Direction	Specify whether the edge searches should be InsideOut (from RadiusInner to RadiusOuter), or OutsideIn (from RadiusOuter to RadiusInner).
DefectLevelThreshPos	Specify the minimum distance above the arc being inspected from edges found to the arc.
DefectLevelThreshNeg	Specify the minimum distance below the arc being inspected from edges found to the arc.
MinArea	Specify the minimum area of a defect in pixels.
MaxArea	Specify the maximum area of a defect in pixels.

Step 4: Running the ArcInspector Object and Examining the Results

To run the ArcInspector object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab. Results for the ArcFinder object will now be displayed. The primary results to examine at this time are:

Area result	The area (in pixels) of the defect found.
PixelX Result **PixelY Result**	Returns the XY coordinates of the defect in the image coordinate system.
CameraX Result **CameraY Result**	Returns the XY coordinates of the defect in the Camera coordinate system.
	If the calibration is not performed, "no cal" will be returned.
RobotX Result **RobotY Result**	Returns the XY coordinates of the defect in the Robot coordinate system.
	If the calibration is not performed, "no cal" will be returned.

6.2.14 ⊞ DefectFinder Object

DefectFinder Object Description

DefectFinder objects are used to identify differences between a template image and an input image.

During the search for defects, first the absolute difference image between the search area and the template is computed. Next, blob analyisis is performed on the difference image to find the defects.

The following defect features are obtained:

- Area and perimeter

- Center of mass

- Principal axes and moments

- Connectivity

- Extrema

- Coordinate positions of the center of mass in pixel, camera world, and robot world Coordinate systems

- Holes, roughness, and compactness of defects (blobs)

DefectFinder Object Layout

The DefectFinder object layout is rectangular similar to the Blob object. However, DefectFinder object requires teaching a model (the template). For model teaching, the entire area defined by the search window is used. There is no separate model window as used for the Correlation object.

The search window defines the area within which DefectFinder searches for defects (image differences). Also, it defines the area of the template image. An example of the DefectFinder object is shown below:

DefectFinder Object Layout
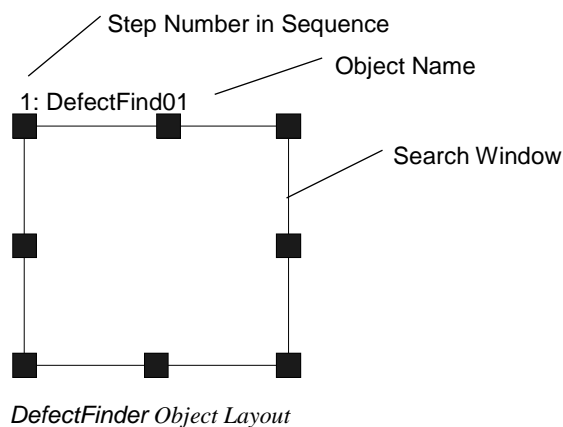
DefectFinder Object Properties

The following list is a summary of properties for the DefectFinder object. The details for each property are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Caption	Assigns a caption to the Blob object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point of the object. This property is normally set to base the center on the XY position result from another vision object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject. Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject. Default: False
CurrentResult	Defines which result to display in the Results list (on the Object tab) or which result to return data for when the system searches more than one result. Default: 1
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color of an object when it is not accepted. Default: Red
Frame	Specifies which positioning frame to use. Default: None
FrameResult	Specifies which number of the Frame results to be used. Default: 1
Graphics	Specifies which graphics to display. Default: 1 – All
KernelHeight	Allowable amount of pixel difference when computing the image difference from the registered image. (in vertical direction)

Property	Description
KernelWidth	Allowable amount of pixel difference when computing the image difference from the registered image. (in horizontal direction)
LabelBackColor	Selects the background color for an object label. Default: Transparent
LuminanceCorrection	Sets the use of luminance correction preprocessing. Default: None
MaxArea	Defines the upper Area limit for a defect. For a defect to be found it must have an Area result below the value set for MaxArea property. Default: 100,000
MinArea	Defines the lower Area limit for a defect. For a defect to be found it must have an Area result above the value set for MinArea property. Default: 25
MinMaxArea	Runtime only. Sets or returns both MinArea and MaxArea in one statement.
Name	Used to assign a unique name to the Blob object. Default: DefectFind01
NumberToFind	Defines the number of objects to find in the search window. Default: 1
PassColor	Selects the color for an object when it is passed. Default: Light Green
PassType	Selects the rule that determines if the object passed. Default: AllNotFound
Polarity	Sets the polarity of defects to detect. Default: Both
RejectOnEdge	If the property is set to True, the system ignores defectives detected on the edge of the search window. Default: 0 - False
SearchWin	Runtime only. Sets or returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched in pixels. Default: 100
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.

Vision Guide 7.0 (Ver.7.4) Software Rev.1

Property	Description
SearchWinType	Defines the type of the area to be searched (i.e . Rectangle, RotatedRectangle, Circle).
SearchWinWidth	Defines the width of the area to be searched in pixels. Default: 100
ShowModel	Displays the registered image.
SizeToFind	Selects which size of defects to find. Default: 1 - Largest
Sort	Selects the sort order used for the results of an object. Default: 0 – None
ThresholdHigh	Works with the ThresholdLow property to define the gray level regions that represent the feature (or object), the background and the edges of the image. The ThresholdHigh property defines the upper bound of the gray level region for the feature area of the image. Any part of the image that falls within gray level region defined between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is part of the feature.) Default: 128
ThresholdLow	Works with the ThresholdHigh property to define the gray level regions that represent the feature (or object), the background and the edges of the image. The ThresholdLow property defines the upper bound of the gray level region for the feature area of the image. Any part of the image that falls within gray level region defined between ThresholdLow and ThresholdHigh will be assigned a pixel weight of 1. (i.e. it is part of the feature.) Default: 0

DefectFinder Object Results

The following list is a summary of the DefectFinder object results with brief descriptions. The details for each result are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Results	Description
Angle	Returns the amount of part rotation in degrees.
Area	Returns the area of the blob in pixels.
CameraX	Returns the X coordinate position of the found part's position in the camera coordinate system.
CameraY	Returns the Y coordinate position of the found part's position in the camera coordinate system.
CameraXYU	Runtime only. Returns the CameraX, CameraY, and CameraU coordinates of the found part's position in the camera coordinate system.
Compactness	Returns the compactness of a blob.
Extrema	Runtime only. Returns MinX, MaxX, MinY, MaxY pixel coordiates of the blob Extrema.

Results	Description
Found	Returns whether or not the object was found. (i.e. was a Connected Blob found which has an Area result that falls between the MinArea and MaxArea properties.)
FoundOnEdge	Returns TRUE when a Blob object is found too close to the edge of the search window.
Holes	Returns the number of holes found in the blob.
MaxX	Returns the maximum X pixel coordinate of the blob Extrema in pixels.
MaxY	Returns the maximum Y pixel coordinate of the blob Extrema in pixels.
MinX	Returns the minimum X pixel coordinate of the blob Extrema in pixels.
MinY	Returns the minimum Y pixel coordinate of the blob Extrema in pixels.
NumberFound	Returns the number of blobs found within the search window. (This number can be anywhere from 0 up to the number of blobs you requested the Blob object to find with the NumberToFind property.)
Perimeter	The number of pixels along the outer edge of the found blob.
PixelX	Returns the X coordinate position of the found part's position in pixels.
PixelY	Returns the Y coordinate position of the found part's position in pixels.
PixelXYU	Runtime only. Returns the PixelX, PixelY, and PixelU coordinates of the found part's position in pixels.
RobotX	Returns the X coordinate of the detected object in the robot coordinate system.
RobotY	Returns the Y coordinate of the detected object in the robot coordinate system.
RobotU	Returns the U coordinate of the detected object in the robot coordinate system.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the detected object in the robot coordinate system.
Roughness	Returns the roughness of a blob.
ShowAllResults	Displays a dialog which allows you to see all results for a specified vision object in a table form. This makes it easy to compare results when the NumberToFind property is set to 0 or greater than 1.
Time	Returns the amount of time in milliseconds required to process the object.
TotalArea	Returns the sum of areas of all results found.

DefectFinder objects perform the Blob processing to the image difference of the template image and returns results. For details on results, refer to *Blob Object*.

Using DefectFinder Objects

Now that we've reviewed how blob analysis works we have set the foundation for understanding how to use Vision Guide 7.0 DefectFinder objects. This next section will describe the steps required to use DefectFinder objects as listed below:

- How to create a new DefectFinder object
- Position and Size the search window
- Configure the properties associated with the DefectFinder object
- Register the template image
- Test the Blob object & examine the results
- Make adjustments to properties and test again

Prior to starting the steps described above, you have to create a new vision sequence or selecte a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence ☐ > toolbar button.

You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab.

Refer to *7. Vision Sequences* for more details on how to create a new vision sequence or select one that was previously defined.

Step 1: Create a New DefectFinder Object

(1) Click on the <NewDefectFinder ⊡ > toolbar button on the Vision Guide 7.0 toolbar. Move the mouse over the image display. You will see the mouse pointer change to the DefectFinder icon.

(2) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(3) A name for the object is automatically created. In the example, it is called "DefectFind01" because this is the first DefectFinder object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a DefectFinder object similar to the one shown below:

Step Number in Sequence

1: DefectFind01 ——— Object Name

Search Window

New DefectFinder Object Layout
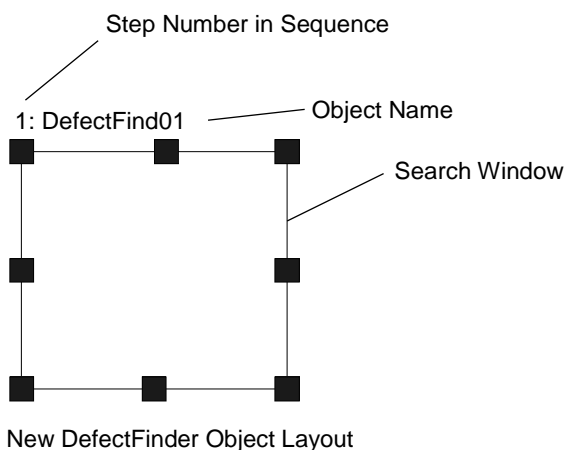
(1) Click on the name label of the DefectFinder object and, while holding the mouse down, drag the DefectFinder object to the position where you would like the top left position of the search window to reside.

(2) Resize the DefectFinder object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.) (The search window is the area within which we will search for Blobs.)

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Step 3: Configure the DefectFinder Object Properties

We can now set property values for the DefectFinder object. Shown below are some of the more commonly used properties that are specific to the DefectFinder object. Explanations for other properties such as AbortSeqOnFail, Graphics, etc. which are used on many of the different vision objects can be seen in the Vision Guide 7.0 Properties and Results Reference Manual or in the "DefectFinder Object Properties" list earlier in this chapter.

⚠ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Name property	The default name given to a newly created DefectFinder object is "DefectFindxx" where xx is a number which is used to distinguish between multiple DefectFinder objects within the same vision sequence. If this is the first DefectFinder object for this vision sequence then the default name will be "DefectFind01". To change the name, click on the Value field of the Name property and type a new name and hit return. You will notice that every place where the DefectFinder object's name is displayed is changed to reflect the new name.
KernelWidth, KernelHeight properties	Specifies an allowable amount of pixel difference when computing the image difference from the registered image. Large setting value can make the search resistant to ambient lighting or the gap between the input image and the template image. However, the search will not be able to find small defects. Set this value according to the defect size and the gap amount of the input image.
MinArea, MaxArea properties	These properties define the area limit for a DefectFinder object to be considered "Found". (i.e. the Found result returned as True) The default range is set as 25-100,000 (MinArea, MaxArea) which is a very broad range. This means that most blobs will be reported as Found when you first run a new DefectFinder object before adjusting the MinArea and MaxArea properties. Normally, you will want to modify these properties to reflect a reasonable range for the blob you are trying to find. This way if you find a blob which is outside of the range you will know it isn't the blob you wanted to find.
RejectOnEdge property	Excludes the parts touching the boundary of the search window.
PassType property	Selects the acceptance condition for DefectFinder object detection. Normally, no defects found means "Passed". Set the property to "AllNotFound".

Now, the DefectFinder object can be tested. Other necessary properties will be set after the test.

Step 4: Register and Confirm the Template Image

To register the DefectFinder object template image, click the <Teach> button at the bottom right of the Object tab. To view the registered template image, click ShowModel property in the Object tab.

Step 5: Test the DefectFinder Object and Examine the Results

To run the DefectFinder object, click on the <Run Object> button located at the bottom left of the object Tab. Results for the DefectFinder object will now be displayed. The primary results to examine at this time are shown below. There are others that you will find useful in the future as well though.

Found result	Returns whether or not the blob was found.
	If the blob that was found does not meet the area constraints defined by the MinArea and MaxArea properties then the Found result will return as False.
Passed result	Returns whether the detection result of the the DefectFinder object was accepted or not.
Area result	The area (in pixels) of the blob found.
Angle result	The angle at which the Blob is oriented. This is computed from the angle of the minor axis and will be a value between +/- 90 degrees.
Time result	The amount of time it took for the DefectFinder object to execute.
PixelX, PixelY	The XY position (in pixels) of the center of mass of the found blob.
MinX, MinY, MaxX, MaxY	Combined these 4 values return circumscribed rectangle of the blob

NOTE

☞

The RobotXYU, RobotX, RobotY, RobotU and CameraX, CameraY, CameraXYU results will return "no cal" at this time. This means that no calibration was performed so it is impossible for the vision system to calculate the coordinate results with respect to the Robot World or Camera World. Refer to *7. Vision Calibration* for more information.

Step 6: Make Adjustments to Properties and Test Again

After running the DefectFinder object a few times, you may have encountered problems with finding a blob or just want to fine-tune some of the property settings. Some common problems and fine tuning techniques are described below:

Problems : If the DefectFinder object returns a Found result of False, there are a few places to immediately examine.

- Look at the value defined for the Polarity property. Are you looking for a light object on a dark background or a dark object on a light background? Make sure that the Polarity property coincides with what you are looking for and with what you see within the search window.

- Look at the Area result and compare this area with the values defined in the MinArea and MaxArea properties. If the Area result does not fall between the limits defined by the MinArea and MaxArea properties, then you may want to adjust these properties and run the Blob object again.

- Use Histograms to examine the distribution of gray-scale values in an image. The Histogram tool is excellent for setting the ThresholdHigh and ThresholdLow properties. Histograms are described in detail in *8. Histograms Tools*.

Fine Tuning : Fine-tuning of the DefectFinder object may be required for some applications. The primary properties associated with fine-tuning of a DefectFinder object are described below:

- MinArea, MaxArea - After you have run the DefectFinder object a few times, you will become familiar with the approximate values returned for the Area result. Use these values when determining new values to enter to the MinArea and MaxArea properties. It is generally a good idea to have MinArea And MaxArea properties set to values which constrain the Found result such that only blobs which you are interested in are returned with the Found result equal to True. (This helps eliminate unwanted blobs that are different in area from the desired blob.)

- ThresholdHigh, ThresholdLow - These properties adjust parameters for the setting the gray levels thresholds for distinguishing between what is background and what is part of the blob. These properties are best set through using the Histogram tool. Please refer to the descriptions of the ThresholdHigh and ThresholdLow properties in the *Vision Properties and Results Reference Manual*. Histograms are described in detail in *Histograms and Statistics Tools*.

Once you have completed making adjustments to properties and have tested the DefectFinder until you are satisfied with the results you are finished with creating this vision object and can go on to creating other vision objects or configuring and testing an entire vision sequence.

6.2.15 ⌐ Frame Object

Frame Object Description

Frame objects provide a type of dynamic positioning reference for vision objects. Once a Frame object is defined other vision objects can be positioned with respect to that Frame. This proves useful for a variety of situations. It also helps reduce application cycle time because once a rough position is found and a Frame defined, the other vision objects that are based on that Frame object need not have large search windows. (Reducing the size of search windows helps reduce vision-processing time.)

Frame objects are best used when there is some type of common reference pattern on a part (such as fiducials on a printed circuit board) which can then be used as a base position on which other vision objects search window locations are based.

Frame Object Layout

The Frame object looks like 2 Line objects that intersect. The user can adjust the position of the Frame object by clicking on the Frame object's name and then dragging the object to a new position. However, in most cases the Frame object position and orientation will be based on other vision object's position.

3: Frame 01 — Object Name
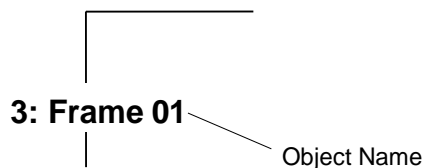
Frame Object Layout

Frame Object Properties

The following list is a summary of properties for the Frame object. The details for each property are explained in the Vision Guide 7.0 Properties and Results Reference Manual.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Caption	Used to assign a caption to the Frame object. Default: Empty String
CurrentResult	Defines which result to display in the Results list (on the Object tab) or which result to return data for when the system searches more than one result. Default: 1
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color of an object when it is not accepted. Default: Red
Graphics	Specifies which graphics to display. Default:1 – All
LabelBackColor	Sets the background color for the object's label. Default: Transparent
Name	Used to assign a unique name to the Frame object. Default: Framexx
OriginAngleEnabled	Enables single point frame usage which causes the frame to rotate with the angle of the origin object instead of based on the rotation of the vector between the OriginPoint and the YaxisPoint as in two point Frames. Default: 0 – False

Property	Description
OriginPntObjResult	Specifies the result to use from the vision object specified in the OriginPoint property.
	If "All" is specified, the Frame object will be applied to all the specified vision objects.
	Default: 1 (Use the first result)
OriginPoint	Defines the vision object to be used as the origin point for the Frame object.
	Default: Screen
PassColor	Selects the color for passed objects.
	Default: LightGreen
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound
ShowExtensions	Specifies whether to display extensions of the frame.
	Default: False
YAxisPoint	Defines vision object to be used as the Y-axis point for the frame. (Defines the direction of the Frame object.)
	Default: Screen
YAxisPntObjResult	Specifies the result to use from the vision object specified in the YAxisPoint property.
	Default: 1 (Use the first result)

Frame Object Results

The details for each result used with Frame objects are explained in the Vision Guide 7.0 Properties and Results Reference Manual. However, for convenience we have provided a list of the Frame object results and a brief explanation of each below:

Results	Description
Angle	Returns the amount of rotation of the Frame.
Found	Returns whether or not the frame could be defined.

Two Point Frames

Two point Frame objects require an origin (defined by the OriginPoint property) and a Y-Axis direction (defined by the YAxisPoint property). The combination of Origin and Y Axis direction define what can be thought of as a local coordinate system within which other vision objects can be based. The power of the Frame object is that when the Frame object moves, all the vision objects which are defined within that frame move with it. (i.e. Their search windows adjust based on the XY change of the vision object defined as the OriginPoint and the rotation created with the movement of the YaxisPoint property.) This allows you to keep your search windows small that in turn improves reliability and-processing time.

Defining a Frame Object

Once a new Frame object is created, it requires 2 vision objects to be used as reference positions for the OriginPoint and YAxisPoint of the Frame. These are defined with the OriginPoint property and YAxisPoint property. Any vision object which has XY position results can be used to define a Frame Origin or YAxisPoint. This means that Blob, Correlation, Edge, Polar, and Point objects can all be used to define the Origin or YAxisPoint property for a Frame object.

Single Point Frames

Single point frames are an optional usage method for the Frame object. With this usage the OriginPoint property specifies a vision object to be used as an XY positional origin reference. When the OriginAngleEnabled property is set to FALSE, the frame adjusts position based on the XY position change of the vision object used as the OriginPoint. No rotation is taken into account. This is useful for simple XY shifts where one object (such as a blob or correlation) finds the XY position of a part, and then the rest of the objects in the frame adjust in X and Y accordingly.

In some cases, you may also need to account for rotation within your frame. Assume that a Blob object is used to find a part's X, Y, and U position (XY coordinate + rotation). Then let's assume that a variety of other vision objects are required to find features on the part. The Blob object can be used to define a single point frame including rotation of the frame, and the other objects within the frame will then shift in X,Y and rotate based on the rotation returned by the Blob object. Hence, only one vision object was required to define both the XY shift and rotation of the part. This shows why the YaxisPoint property is not required for Single Point Frames.

Using Frame Objects

The next few sections guide you through how to create and use a Frame object.

- How to create a new Frame object

- Position and Size the search window

- Configure the properties associated with the Frame object

- Test the Frame object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence ☐ toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. See Vision Sequences for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New Frame Object

(1) Click on the New Frame toolbar button [icon] on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the Frame object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "Frame01" because this is the first Frame object created for this sequence. (We will explain how to change the name later.).

(5)

Step 2: Positioning the Frame Object

You should now see a Frame object similar to the one shown below:

3: Frame01

Object Name
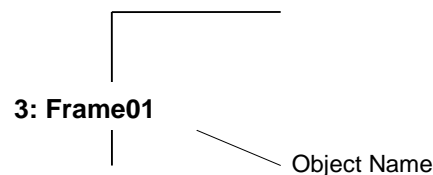
New Frame Object

Frame objects do not have a sizeable window. Click on the name label of the Frame object or anywhere along one of its axes and while holding the mouse down drag the entire Frame object to a new location on the screen. When you find the position you like, release the mouse and the Frame object will stay in this new position on the screen.

Step 3: Configuring Properties for the Frame Object

We can now set property values for the Frame object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the Frame object. Explanations for other properties such as AbortSeqOnFail, and Graphics, which are used on many of the different vision objects, can be seen in the Vision Guide 7.0 Properties and Results Reference Manual or in the *Frame Object Properties* previously described in this section.

Name property - The default name given to a newly created Frame object is "Framexx" where xx is a number which is used to distinguish between multiple Frame objects within the same vision sequence. If this is the first Frame object for this vision sequence then the default name will be "Frame01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Frame object's name is displayed is updated to reflect the new name.

OriginPoint property - Typically you will set this property to one of the objects that occur previously in the sequence. This will determine the origin of the frame at runtime.

YAxisPoint property - Typically you will set this property to one of the objects that occur previously in the sequence. This will determine the direction of the Y axis of the frame at runtime.

Step 4: Running the Frame Object and Examining the Results

To run the Frame object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab. If either the OriginPoint or YAxisPoint properties are not Screen, then the respective objects will be run first. For example, if the OriginPoint is a Blob object, then the blob will be run first to determine the position of the origin of the frame.

Results for the Frame object will now be displayed. The primary results to examine at this time are:

Angle - The angle of the frame.

6.2.16　 Line Object

Line Object Description

Line objects are used to define a line between 2 points. The Points can be either based on positions on the screen or on other vision object positions. For example, shown below are just some of the situations where a Line object can be created:

- between 2 Blob objects
- between 2 Correlation objects
- between 2 Point objects
- between a Blob object and a Correlation object
- between 2 results on a single Blob object
- between result 1 on a Blob object and result 3 on a Correlation object
- between a Point object and a Correlation object
- any other variety of combinations between objects that have an XY position associated with them.

Line objects are useful for the following situations:

- To measure the distance between two vision objects (or vision object results when multiple results are used) The distance can also be checked to make sure it falls between a minimum and maximum distance as per your application requirements.

- To calculate the amount of rotation between 2 vision objects (use the angle of the line returned in Robot Coordinates called the RobotU result)

- To create a building block for computing the midpoint of a line or intersection point between 2 lines

Line Object Layout

The Line object layout appears just as you would expect it to. It's just a line with a starting point, an ending point, and an object name. To reposition the Line object, simply click on the name of the Line object (or anywhere on the line) and then drag the line to a new position. To resize a Line object, click on either the starting or ending point (shown by the sizing handles) of the line and then drag it to a new position.
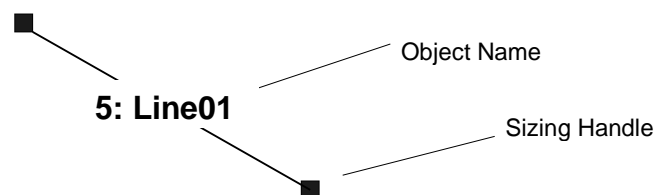
Line Object Layout

Line Object Properties

The following list is a summary of properties for the Line object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 – False
AngleBase	Sets the reference angle Default: 0
AngleMode	Sets the angle output format. Default: 1–Default
LabelBackColor	Sets the background color for the object's label. Default:Transparent
Caption	Used to assign a caption to the object. Default: Empty String
Directed	Specifies whether to set the angle using the line direction Default: True
Enabled	Specifies whether to execute the object. Default: True
EndPntObjResult	Specifies which result to use from the EndPointObject. Default: 1
EndPointObject	Specifies which vision object to use to define the end point of the Line. Default: Screen
EndPointType	Defines the type of end point used to define the end point of a line. Default: 0 – Point
FailColor	Selects the color for an object when it is failed. Default: Red
Frame	Specifies which positioning frame to use.

Property	Description
	Default: none
Graphics	Specifies which graphics to display.
	Default: 1 – All
MaxLength	Defines the upper length limit for the Line object. For a Line to be found it must have a Length result below the value set for MaxLength property.
	Default: 9999
MaxPixelLength	Defines the upper pixel length limit for the Line object. For a Line to be found it must have a PixelLength result below the value set for MaxPixelLength property.
	Default: 9999
MinLength	Defines the lower length limit for the Line object. For a Line to be found it must have a Length result above the value set for MinLength property.
	Default: 0
MinPixelLength	Defines the lower length limit for the Line object. For a Line to be found it must have a PixelLength result above the value set for MinPixelLength property.
	Default: 0
Name	Used to assign a unique name to the Line object.
	Default: Linexx
PassColor	Selects the color for passed objects.
	Default: LightGreen
PassType	Selects the rule that determines if the object passed.
	Default:SomeFound
ShowExtensions	When set to True, this causes the graphics display of the line to be extended to the end of the image display.
	Default: 0 – False
StartPntObjResult	Specifies which result to use from the StartPointObject.
	Default: 1
StartPointObject	Specifies which vision object to use to define the start point of the Line.
	Default: Screen
StartPointType	Defines the type of start point used to define the start point of a line.
X1	The X coordinate position of the start point of the line.
X2	The X coordinate position of the end point of the line.
Y1	The Y coordinate position of the start point of the line.
Y2	The Y coordinate position of the end point of the line.

Line Object Results

The following list is a summary of the Line object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual.*

Results	Description
Angle	Returns the angle of the Line formed from StartPoint to EndPoint with respect to the 0 degrees position at 3 O'clock.
CameraX1	Returns the X coordinate position of the starting point of the line in the camera coordinate system.
CameraX2	Returns the X coordinate position of the ending point of the line in the camera coordinate system.
CameraY1	Returns the Y coordinate position of the starting point of the line in the camera coordinate system.
CameraY2	Returns the Y coordinate position of the ending point of the line in the camera coordinate system.
Found	Returns whether or not the object was found. (i.e. If the vision objects used to define the Line object are not found, then the Line object is not found.)
Length	Returns the length of the line in millimeter units. (The camera must be calibrated or "no cal" will be returned as the length.) If the Line object fails due to the MaxLength and MinLength constraints, the Length result is shown in red in the Results list.
PixelLength	Returns the length of the line in pixels. If the Line object fails due to the MaxPixelLength and MinPixelLength constraints, the PixelLength result is shown in red in the Results list.
PixelLine	Runtime only. Returns the four line coordinates X1, Y1, X2, Y2 in pixels.
PixelX1	Returns the X coordinate position of the starting point of the line in pixels.
PixelX2	Returns the X coordinate position of the ending point of the line in pixels.
PixelY1	Returns the Y coordinate position of the starting point of the line in pixels.
PixelY2	Returns the Y coordinate position of the ending point of the line in pixels.
RobotU	Returns the angle of the Line formed from StartPoint to EndPoint with respect to the Robot Coordinate System.

Using Line Objects

The next few sections guide you through how to create and use an Line object.

- How to create a new Line object
- Position and Size the search window
- Configure the properties associated with the Line object
- Test the Line object & examine the results
- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence [ ] toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. Refer to *5. Vision Sequences* for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New Line Object

(1) Click on the <New Line [ ]> toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button).

(2) Move the mouse over the image display. You will see the mouse pointer change to the Line object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display, then click the left mouse button to create the object.

(4) Notice that a name for the object is automatically created. In the example, it is called "Line01" because this is the first Line object created for this sequence. (We will explain how to change the name later.).

Step 2: Positioning the Line Object

You should now see a Line object similar to the one shown below:
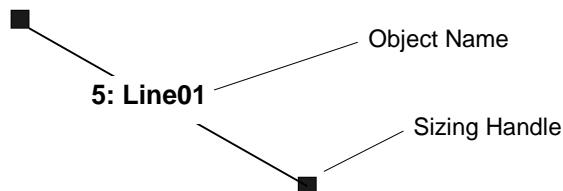
New Line Object

Line objects do not have a window. You can change the length and rotation by clicking down on either size handle, and then dragging that end of the line to a new position. You can also click on the name label of the Line object or anywhere along the line and while holding the mouse down drag the entire Line object to a new location on the screen. When you find the position you like, release the mouse and the Line object will stay in this new position on the screen.

Step 3: Configuring Properties for the Line Object

We can now set property values for the Line object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the Line object. Explanations for other properties such as AbortSeqOnFail and Graphics, which are used on many of the different vision objects, can be seen in the *Vision Guide 7.0 Properties and Results Reference Manual* or in the *Line Object Properties* found previously in this section.

Name property - The default name given to a newly created Line object is "Linexx" where xx is a number which is used to distinguish between multiple Line objects within the same vision sequence. If this is the first Line object for this vision sequence then the default name will be "Line01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Line object's name is displayed is updated to reflect the new name.

StartPointObject property - Typically you will set this property to one of the objects that occur previously in the sequence. This will determine the starting point of the line at runtime.

EndPointObject property - Typically you will set this property to one of the objects that occur previously in the sequence. This will determine the end point of the line at runtime.

Set the AngleMode property to "2-UseAngleBase" to specify the angle output format using Directed and AngleBase property settings. The Directed property specifies whether angles are dependent on the orientation of Line objects. Angles are output using the angle set with AngleBase as a reference.

For example, when performing measurements based on the angle set when teaching, the angle output will be the angle measured around the angle set to the AngleBase property as the center. Set the AngleMode property to "1-Default" to output angles from 0 to 360 degrees.

(When doing so, the following configuration will result in the same action being performed.

 AngleMode property: 2-UseAngleBase
 AngleBase: 0
 Directed: True)

Step 4: Running the Line Object and Examining the Results

To run the Line object, simply do the following:

Click on the <Run Object> button located at the bottom left of the Object tab. If either the StartPointObject or EndPointObject properties are not Screen, then the respective objects will be run first. For example, if the StartPointObject is a Blob object, then the blob will be run first to determine the position of the starting point of the line.

Results for the Line object will now be displayed. The primary results to examine at this time are:

Length results   - The length of the line in millimeters. There must be a calibration associated with the sequence that contains the line for Length to be determined.

PixelX1, PixelY1, PixelX2, PixelY2 results

         - The XY position (in pixels) for both ends of the line.

PixelLength results   - The length of the line in pixels.

6.2.17 ⊞ Point Object

Point Object Description

Point objects can be thought of as a type of utility object that is normally used with other vision objects.

Point objects are most useful in defining a position references for Polar and Line objects as follows:

Polar Objects: Point objects can be used to define the CenterPoint property which is used as the center of the Polar object.

Line Objects: Point objects can be used to define the start point, midpoint, or end point of a single line or the intersection point of 2 lines.

Point Object Layout

The Point object layout appears as a cross hair on the screen. The only real manipulation for a Point object is to change the position. This is done by clicking on the Point object and dragging it to a new position. In most cases Point objects are attached to other objects so the position is calculated based on the position of the associated object.
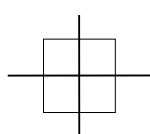
3: Point01 —————— Object Name

Point Object Layout*:*

Point Object Properties

The following list is a summary of properties for the Point object. The details for each property are explained in the *Vision Guide 7.0 Properties and Results Reference Manual.*

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
AngleObject	Specifies objects with an output angle. Default: Screen
Caption	Used to assign a caption to the object. Default: Empty String
Enabled	Specifies whether to execute the object. Default: True
FailColor	Selects the color for an object when it is failed. Default: 1 –Red
Frame	Specifies which positioning frame to use. Default: none
Graphics	Specifies which graphics to display. Default: 1 - All
LabelBackColor	Sets the background color for the object's label. Default: Transparent
LineObject1	Defines a Line object used to define the position of the point object as the midpoint of the line or as the intersection of 2 lines. Default: none
LineObject2	Defines the 2nd Line object used to define the position of the point as the intersection of 2 lines. Default: none
Name	Used to assign a unique name to the Point object. Default: Pointxx
PassColor	Selects the color for passed objects. Default: LightGreen
PassType	Selects the rule that determines if the object passed. Default: SomeFound
PointType	Defines the position type for the point. Default: Screen
X	The X coordinate position of the Point object in pixels.
Y	The Y coordinate position of the Point object in Pixels.

Point Object Results

The following list is a summary of the Point object results with brief descriptions. The details for each result are explained in the *Vision Guide 7.0 Properties and Results Reference Manual*.

Results	Description
CameraX	Returns the X coordinate position of the Point object's position in the camera coordinate system.
CameraY	Returns the Y coordinate position of the Point object's position in the camera coordinate system.
CameraXYU	Runtime only. Returns the CameraX, CameraY, and CameraU coordinates of the Point object's position in the camera coordinate system.
ColorValue	Returns the grayscale value or color value of the pixel at the current location.
Found	Returns whether or not the Point object was found. (i.e. If the vision objects used to define the position of the Point object are not found, then the Line object is not found.)
PixelX	Returns the X coordinate position of the Point object's position in pixels.
PixelY	Returns the Y coordinate position of the Point object's position in pixels.
PixelXYU	Runtime only. Returns the PixelX, PixelY, and PixelU coordinates of the found part's position in pixels.
RobotX	Returns the X coordinate position of the found part's position (referenced by model origin) with respect to the Robot's Coordinate System.
RobotY	Returns the Y coordinate position of the found part's position (referenced by model origin) with respect to the Robot's Coordinate System.
RobotXYU	Runtime only. Returns the RobotX, RobotY, and RobotU coordinates of the Point object's position with respect to the Robot Coordinate System. (Keep in mind that the U coordinate value has no meaning since a Point object has no rotation.)

Understanding Point Objects

Point objects were created to give users a method to mark a position on the screen or with respect to Line objects so that these positions could be used by other vision objects as reference positions. There are 2 fundamental parts to understanding Point objects:

1. Defining the position of the Point object

2. Using the position of the Point object as a reference position for other objects

Defining the Position of the Point Object

The position of a Point object can either be based upon the position on the screen where you place the Point object, the midpoint of a Line object, or the intersection between 2 Line objects. The PointType property is used to define which item the Point object's position is based upon. The PointType property can be set to Screen, MidPoint, or Intersection.

When the PointType property is Set to Screen

When a Point object is first created the default PointType is Screen which means that the position of the Point object is based upon the position on the screen where the Point object was placed. As long as the PointType is set to Screen, the Point object's position only changes when you move it manually with the mouse or change the values of the X property or Y property.

Setting the PointType property to MidPoint

To define a Point object's position as the Midpoint of a line can be done by setting the PointType property to MidPoint. However, it is important to note that you must first define the line to be used prior to setting PointType to MidPoint. Otherwise, Vision Guide 7.0 would have no idea about which line to use for computing the midpoint. The LineObject1 property is used to define the Line object to for which the midpoint will be computed and the position of the Point object then set to. Any valid Line object that is ahead of the Point object in the vision sequence Step list can be selected from the drop down list of the LineObject1 property. In fact Vision Guide automatically checks which Line objects are ahead of the Point object in the sequence Step list and only displays these lines in the LineObject1 drop down list. This makes the system easier to use.

If you try to set the PointType property to MidPoint without first specifying the Line object to be used with the LineObject1 property, then an error message will appear telling you that you must first select a line for the LineObject1 property.

Setting the PointType property to Intersection

To define a Point object's position as the intersection point between 2 lines requires that the 2 lines first be defined. This is done with the LineObject1 and LineObject2 properties. The LineObject1 and LineObject2 properties must each define a different Line object. Once a Line object is defined for each, the PointType property can be set to Intersection which indicates that the position of the Point object will be the intersection point of the 2 lines defined by the LineObject1 and LineObject2 properties.

Any valid Line object which is ahead of the Point object in the vision sequence Step list can be selected from the drop down list of the LineObject1 property to be used as the 1st line required for the intersection. Then any valid remaining Line object which is ahead of the Point object in the vision sequence Step list can be selected from the drop down list of the LineObject2 property to be used as the 2nd line required for the intersection. Vision Guide automatically takes care of displaying only those Line objects that are valid Line objects for use as LineObject1 or LineObject2 in the associated drop down lists.

If you try to set the PointType property to Intersection without first specifying the Line objects to use for both LineObject1 and LineObject2, then an error message will appear. The error message will tell you that you must first define the line for LineObject1 or LineObject2 prior to setting the PointType property to Intersection. Depending upon whichever Line object is not defined.

<table>
<tr><td>

NOTE

☞

</td><td>

Using the intersection of 2 lines to define a position is useful for computing things such as the center of an object. For example, consider a rectangular object. Once you find the 4 corners, you can create 2 diagonal lines that intersect at the center of the rectangle. A Point object positioned on this intersection point is then positioned at the center of the rectangle.

</td></tr>
</table>

Using the Position of the Point Object as a Reference Position for Other Objects

The Point object's primary purpose is to act as a reference position for other vision objects such as the Line and Polar objects. This means that the position of the Point object is used as a base position for a Line or Polar object. This is powerful because it allows you to create such situations as a Point object which is defined as the intersection of 2 lines to calculate the center of an object, which can then be used as an end point for a Line object which calculates the distance between the centers of 2 objects.

Point Objects used as a Reference Position for Polar Objects

The Polar object requires a CenterPoint position to base its Polar Search upon. When first using the Polar object, you may try using the screen as the reference point for the CenterPoint property but you will soon find that as the feature you are searching moves the Polar object must also move with respect to the CenterPosition of the feature. This is where being able to apply the XY position from another vision object (such as the Point object) to the Polar object becomes very powerful.

There may be times where you want to apply the position of the Point object as the CenterPoint of a Polar object. There are 2 primary situations for which Point objects can be used as the CenterPoint of a Polar object.

1. CenterPoint defined as the midpoint of a line

2. CenterPoint defined as the intersection point of 2 lines

For example, after you find the midpoint of a line, you may want to do a Polar search from this midpoint. You may also want to base a Polar search CenterPoint on the intersection point of 2 lines. This is the more common use for Polar objects with Point objects.

Point Objects used as a Reference Position for Line Objects

A line requires a starting and ending position. Many times the starting and ending positions of lines are based on the XY position results from other vision objects such as the Blob or Correlation object. However, you can also use a Point object position as the reference position of a line. A line can be defined with both its starting and ending positions being based on Point objects or it can have just one of the endpoints of the line be based on a Point object.

One of the more common uses of the Point object is to use it to reference the intersection point of 2 lines. The figure below shows an example of 2 line objects (Line1 and Line2) which intersect at various heights. Line 3 is used to calculate the height of the intersection point. The Point object is shown at the intersection point between Line1 and Line2.)
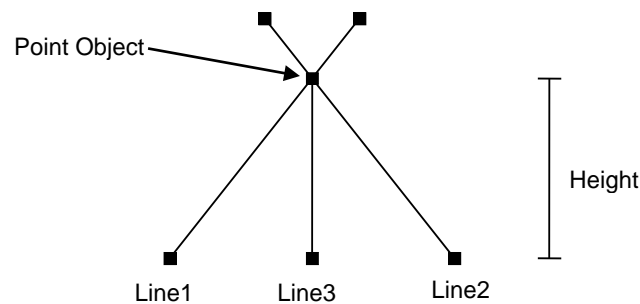
Point object defined as intersection point of Lines 1 and 2

Using Point Objects

The next few pages take you through how to create and use a Point object. We will review the following items

- Create a new Point object
- Positioning the Point object on the screen
- Selecting the Object tab
- Configuring properties associated with the Point object
- Running the Point object and Examining the results

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the New Sequence [ ] toolbar button. You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab. See Vision Sequences for more details on how to create a new vision sequence or select one that was previously defined.

Step 1: Create a New Point Object

(1) Click on the <New Point [+] > toolbar button on the Vision Guide 7.0 Toolbar. (In this case click means to click and release the left mouse button)

(2) You will see a point icon appear above the Point object toolbar button

(3) Click on the point icon and drag to the image display of the Vision Guide window

(4) Notice that a name for the object is automatically created. In the example, it is called "Pnt01" because this is the first Point object created for this sequence. (We will explain how to change the name later.)

Step 2: Positioning the Point Object

You should now see a Point object similar to the one shown below:

3: Point01 ⎯⎯⎯ Object Name
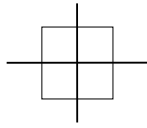
New Point Object Layout

Point objects cannot be resized since they are just a point and have no height or thickness. However, they can be positioned with the mouse or by setting position values into the X and Y properties. Since Point objects are created with a PointType property setting of "0 – Screen", we can move the Point object around a little with the mouse as described below:

(1) Click on the name label of the Point object and while holding the mouse down drag the Point object to a new location on the screen. When you find the position you like, release the mouse and the Point object will stay in this new position on the screen.

Step 3: Configuring Properties for the Point Object

We can now set property values for the Point object. To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed click on one of the items in the list.

Shown below are some of the more commonly used properties for the Point object. Explanations for other properties such as Allows the user to specify that if the specified object fails (not found), then the entire sequence is aborted at that point and no further objects in the sequence are processed., Graphics, etc. which are used on many of the different vision objects can be seen in the Vision Properties and Results Reference Manual or in the Point Object Layout.

The Point object layout appears as a cross hair on the screen. The only real manipulation for a Point object is to change the position. This is done by clicking on the Point object and dragging it to a new position. In most cases Point objects are attached to other objects so the position is calculated based on the position of the associated object.

Point Object Layout

The Point object layout appears as a cross hair on the screen. The only real manipulation for a Point object is to change the position. This is done by clicking on the Point object and dragging it to a new position. In most cases Point objects are attached to other objects so the position is calculated based on the position of the associated object.

3: Point01 ⎯⎯⎯ Object Name
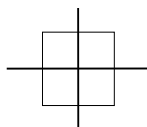
Point Object Layout

Point Object Properties

Point Object Properties list on page 210 earlier in this chapter. We shouldn't have to set any of these properties to test our Point object because the default values are fine since we are not positioning the Point object at the midpoint of a single line or intersection point of 2 lines. However, you may want to read through this information if you are working with Point objects for the first time.

Name property ("Pntxx") - The default name given to a newly created Point object is "Pntxx" where xx is a number which is used to distinguish between multiple Point objects within the same vision sequence. If this is the first Point object for this vision sequence then the default name will be "Pnt01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Point object's name is displayed is updated to reflect the new name.

LineObject1 (None) - If you will set the Point object's PointType property to Midpoint (specifying the midpoint of a line), then this property will specify which line to use. (It is also used to specify the 1st of 2 lines required if the PointType will be an intersection point.)

LineObject2 (None) - If you will set the Point object's PointType property to Intersection (specifying the intersection point of 2 lines), then this property will specify the 2nd line to use. (The 1st line for the intersection is specified by the LineObject1 property.)

PointType (Screen) - This property is used to define the position of the Point object. It can be based on the Screen position, midpoint of a line specified by the LineObject1 property, or the intersection point of 2 lines specified by the LineObject1 and LineObject2 properties.

Since there are no Line objects specified in this example, the LineObject1, LineObject2, and PointType properties cannot be changed from their default state. Normally, we would select a Line Object for the LineObject1 property if we want to make this Point the midpoint of a line. Or we would select a Line object for the LineObject1 property and a 2nd Line object for the LineObject2 property if we want to make this Point an intersection point between two lines. For more details, refer to *Defining the Position of the Point Object* earlier in the section.

Object angles specified using the AngleObject property can be set as the angle output from the Point object. This not only allows Point object output to include the XY position, but for U angle information to be obtained together with the XY position, such as in the form of RobotXYU, by setting the angle of the object preceding the Point object as the Point object angle.

Step 4: Running the Point Object and Examining the Results

To run the Point object, simply do the following:

(1) Click on the <Run Object> button located at the bottom left of the Object tab.

Results for the Point object will now be displayed. The primary results to examine at this time are:

PixelX, PixelY results   - The XY position (in pixels) of the Point object. If the PointType property for the Point object was set to midpoint, then the PixelX and PixelY results would return the XY position (in Pixels) of the midpoint of the Line object specified by LineObject1.

CameraX, CameraY results

       - These define the XY position of the Point object in the Camera's World Coordinate system. The CameraX and CameraY results will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.

RobotX, RobotY results - These define the XY position of the Point object in Robot World Coordinates. The robot can be told to move to this XY position. (No other transformation or other steps are required.) The RobotX and RobotY results will only return a value if the camera has been calibrated. If it has not then [No Cal] will be returned.

6.2.18   BoxFinder Object

BoxFinder Object Description

BoxFinder objects are used to identify the position of rectangle edges (including squares) in an image.

BoxFinder objects process multiple Edge objects automatically to identify the edge position and obtain the rectangle identified from each edge position.

The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels.

Each edge search of the BoxFinder object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and specifies the best line between the detected edge positions. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is returned as the result.

BoxFinder Object Layout

Similar to LineFinder, BoxFinder objects have a direction indicator showing the edge search direction within the search window. This differs from LineFinder in that a direction indicator covers each direction for all four sides of the search window. The number of edge search lines is specified with the NumberOfEdges property. You can specify the search direction using the Direction property.
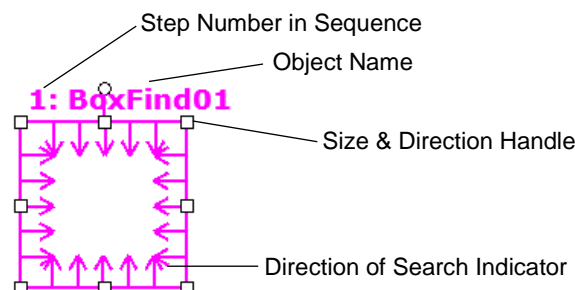
BoxFinder Object Layout

The BoxFinder object can be positioned to search in any direction (not just along the vertical and horizontal directions). Similar to SearchWinType=AngledRectangle for Blob objects, use the handle for rotating the BoxFinder object search window to move the BoxFinder object in the intended edge detection direction.

BoxFinder Object Properties

The following list is a summary of properties for the BoxFinder object. For details on each property, refer to *Vision Guide 7.0 Properties and Results Reference*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. If the value is small, it may result in false detection. Default: 100
Caption	Used to assign a caption to the Edge object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point of the object. If the property is set to "Screen", the object can be placed to anywhere in the screen. If other vision objects are specified, the center point will be set to the PixelX and PixelY results of the object. Default: Screen

Property	Description
CenterPntObjResult	Specifies which result to use from the CenterPointObject.
	Default: 1
CenterPntOffsetX	Specifies the X offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntOffsetY	Specifies the Y offset of the center of the search window after it is positioned with the CenterPointObject.
	Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject.
	If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result.
	Default: False
ContrastTarget	Sets the desired contrast for the edge search.
	Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget.
	Default: 0
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one of a like feature within a single search window.
Direction	Sets the direction for the edge search.
	Default: InsideOut
EdgeSort	Sets the method of sorting detected edge results
	Default: Score
EdgeThreshold	Sets the threshold at which edges below this value are ignored.
	Default: 2
EdgeType	Select the type of edge to search for: single or pair.
	Default: 1 - Single
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color of an object when it is not accepted.
	Default: Red
FittingThreshold	Specifies the edge results to use for linear fittings.
	Default: 10
Frame	Specifies which positioning frame to use.
	Default: none
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1

Property	Description
Graphics	Specifies which graphics to display. Default: 1 - All
LabelBackColor	Selects the background color for an object label. Default: Transparent
MissingEdgeType	Defines how to handle a missing edge. Default: Interpolate
Name	Used to assign a unique name to the Edge object. Default: Edge01
NumberOfEdges	Specified the number of edges to be detected. Default: 5
PassColor	Selects the color for an object when it is passed. Default: LightGreen
PassType	Selects the rule that determines if the object passed. Default: SomeFound
Polarity	Specifies whether the ArcFinder object should search for a LightToDark or DarkToLight transition. Default: 1 - LightToDark
ScoreWeightContrast	Sets the percentage of the score that depends on contrast. Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength. Default: 50
SearchLength	Defines the length of the edge search range. The following SearchLength1 - 4 values can be set together.
SearchLength1	Sets the SearchLength1 length in the figure "BoxFinder object properties, positional relationship of results".
SearchLength2	Sets the SearchLength2 length in the figure "BoxFinder object properties, positional relationship of results".
SearchLength3	Sets the SearchLength3 length in the figure "BoxFinder object properties, positional relationship of results".
SearchLength4	Sets the SearchLength4 length in the figure "BoxFinder object properties, positional relationship of results".
SearchWidth	Defines the width of the edge search. Range is from 3 - 99. Default: 3
SearchWin	Runtime only. Sets / returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.

Property	Description
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched in pixels.
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinWidth	Defines the width of the area to be searched in pixels.
StrengthTarget	Sets the desired edge strength to search for. Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget. Default: 0

BoxFinder object properties, positional relationship of results

BoxFinder Object Results

The following list is a summary of the BoxFinder object results with brief descriptions. For details on each result, refer to *Vision Guide 7.0 Properties and Results Reference*.

Results	Description
Angle	Returns the detected rectangle angle.
CameraX	Returns the central X coordinate of the detected rectangular edge in the Camera coordinate system.
CameraY	Returns the central Y coordinate of the detected rectangular edge in the Camera coordinate system.
CameraX1	Returns the Corner1 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Camera coordinate system.
CameraY1	Returns the Corner1 Y coordinate in the figure "BoxFinder object properties, positional relationship of results"in the Camera coordinate system.
CameraX2	Returns the Corner2 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Camera coordinate system.

Results	Description
CameraY2	Returns the Corner2 Y coordinate in the figure "BoxFinder object properties, positional relationship of results"in the Camera coordinate system.
CameraX3	Returns the Corner3 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Camera coordinate system.
CameraY3	Returns the Corner3 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Camera coordinate system.
CameraX4	Returns the Corner4 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Camera coordinate system.
CameraY4	Returns the Corner4 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Camera coordinate system.
Contrast	Returns the contrast of the detected rectangular edge.
FitError	Returns the distance between each edge point and the rectangle detected as the root mean square (RMS).
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
MaxError	Returns the maximum difference from the detected rectangular edge in pixel length.
Passed	Returns whether or not the object detection result was accepted.
Perimeter	Returns the number of pixels of the perimeter of the detected rectangle.
PixelX	Returns the central X coordinate of the detected rectangular edge in the Image coordinate system.
PixelY	Returns the central Y coordinate of the detected rectangular edge in the Image coordinate system.
PixelX1	Returns the Corner1 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
PixelY1	Returns the Corner1 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
PixelX2	Returns the Corner2 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
PixelY2	Returns the Corner2 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.

Results	Description
PixelX3	Returns the Corner3 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
PixelY3	Returns the Corner3 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
PixelX4	Returns the Corner4 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
PixelY4	Returns the Corner4 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Image coordinate system.
RobotX	Returns the central X coordinate of the detected rectangular edge in the Robot coordinate system.
RobotY	Returns the central Y coordinate of the detected rectangular edge in the Robot coordinate system.
RobotU	Returns the central U coordinate of the detected rectangular edge in the Robot coordinate system.
RobotX1	Returns the Corner1 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotY1	Returns the Corner1 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotX2	Returns the Corner2 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotY2	Returns the Corner2 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotX3	Returns the Corner3 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotY3	Returns the Corner3 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotX4	Returns the Corner1 X coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.
RobotY4	Returns the Corner1 Y coordinate in the figure "BoxFinder object properties, positional relationship of results" in the Robot coordinate system.

Results	Description
ShowAllResults	Displays a dialog showing all results of the specified vision object in a table form.
Strength	Returns the strength of the found edge.
Time	Returns the amount of time in milliseconds required to process the object.

Using BoxFinder Objects

The next few sections guide you through how to create and use a BoxFinder object.

- How to create a new BoxFinder object

- Position and Size the search window

- Configure the properties associated with the BoxFinder object

- Test the BoxFinder object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence▢> toolbar button.

You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab.

Refer to "*Vision Sequences*" for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New BoxFinder Object

(1) Click on the <BoxFinder ⬚> toolbar button on the Vision Guide 7.0 Toolbar.

(2) Move the mouse over the image display.
You will see the mouse pointer change to the BoxFinder object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display.

(4) Cick the left mouse button to create the object.

(5) Notice that a name for the object is automatically created.
In the example, it is called "BoxFind01" because this is the first BoxFinder object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a BoxFinder object similar to the one shown below:
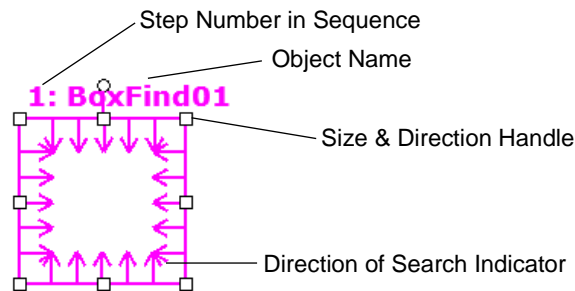
New BoxFinder Object Layout

(1) Click on the name label of the BoxFinder object and, while holding the mouse down, drag the BoxFinder object to the position where you would like the search window to reside.

(2) Resize the BoxFinder object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.)

Step 3: Configuring Properties for the BoxFinder Object

We can now set property values for the BoxFinder object.
To set any of the properties, simply click on the associated property's value field and then
either enter a new value or if a drop down list is displayed select one of the items in the list.

Shown below are some of the more commonly used properties for the BoxFinder object.
For details on other properties such as AbortSeqOnFail and Graphics which are used on
many of the different vision objects, refer to *Vision Guide 7.0 Properties and Results
Reference*.

EdgeType (Single)	Select the type of the edge to be searched.
	For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.
NumberOfEdges(5)	To find edges, you can search for five edges in each corner of the search window.
Polarity (LightToDark)	Search for edges using "LightToDark" polarity.
	If you are looking for a DarkToLight edge, change polarity.

Step 4: Running the BoxFinder Object and Examining the Results

The next few sections guide you through how to run a BoxFinder object.

Click on the <Run Object> button located at the bottom left of the Object tab.
Results for the BoxFinder object will now be displayed. The primary results to examine at
this time are:

Angle Result	Returns the angle of the detected rectangular edge in the Image coordinate system.
MaxError Result	Returns the maximum difference from the detected rectangular edge in pixel length.
PixelX Result PixelY Result	Returns the XY coordinate positions at the center of the detected rectangular edge in the Image coordinate system.
CameraX Result CameraY Result	Returns the XY coordinate positions at the center of the detected rectangular edge in the Camera coordinate system. If the calibration is not performed for the XY coordinate positions, "no cal" will be returned.
RobotX Result RobotY Result	Returns the XY coordinate positions at the center of the detected rectangular edge in the Robot coordinate system. If the calibration is not performed for the XY coordinate positions, "no cal" will be returned.

6.2.19 CornerFinder Object

CornerFinder Object Description

CornerFinder objects are used to identify the position of the corner of two lines in the image.

CornerFinder objects process multiple Edge objects automatically to identify the edge position and obtain the corner of two lines identified from each edge position.

The edge of an object in an image is a change in gray value from dark to light or light to dark. This change may span several pixels.

Each edge search of the CornerFinder object finds the transition from Light to Dark or Dark to Light as defined by the Polarity property and specifies the best line between the detected edge positions. You can also search for edge pairs by changing the EdgeType property. With edge pairs, two opposing edges are searched for, and the midpoint is returned as the result.

CornerFinder Object Layout

Similar to LineFinder, CornerFinder objects have a direction indicator showing the edge search direction within the search window. This differs from LineFinder in that a direction indicator covers each direction for two sides of the search window. The number of edge search lines is specified with the NumberOfEdges property. You can specify the search direction using the Direction property.
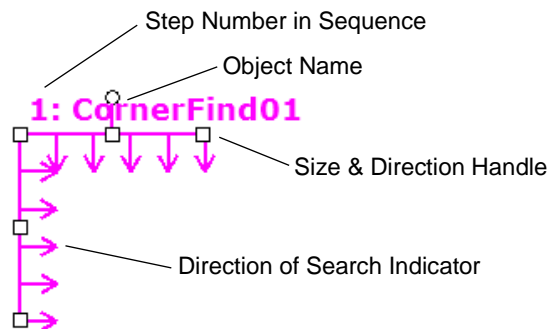
CornerFinder Object Layout

The CornerFinder object can be positioned to search in any direction (not just along the vertical and horizontal directions). Similar to SearchWinType=AngledRectangle for Blob objects, use the handle for rotating the CornerFinder object search window to move the CornerFinder object in the intended edge detection direction.

CornerFinder Object Properties

The following list is a summary of properties for the CornerFinder object. For details on each property, refer to *Vision Guide 7.0 Properties and Results Reference*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. Default: 100
Caption	Used to assign a caption to the Edge object. Default: Empty String
CenterPointObject	Specifies the position to be used as the center point of the object. If the property is set to "Screen", the object can be placed to anywhere in the screen. If other vision objects are specified, the center point will be set to the PixelX and PixelY results of the object. Default: Screen
CenterPntObjResult	Specifies which result to use from the CenterPointObject. Default: 1
CenterPntOffsetX	Sets or returns the X offset after the center of the search window is positioned with the CenterPointObject property. Default: 0
CenterPntOffsetY	Sets or returns the Y offset after the center of the search window is positioned with the CenterPointObject property. Default: 0
CenterPntRotOffset	Specifies whether to rotate the XY offset value of the center (CenterPntOffsetX, CenterPntOffsetY) based on the Angle result of CenterPointObject. If SearchWinType is set to RotatedRectangle, the search window rotates based on the Angle result. Default: False
ContrastTarget	Sets the desired contrast for the edge search. Default: 0 (best contrast)
ContrastVariation	Selects the allowed contrast variation for ContrastTarget. Default: 0
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one of a like feature within a single search window.

Property	Description
Direction	Sets the direction for the edge search.
	Default: InsideOut
EdgeSort	Sets the method of sorting detected edge results.
	Default: Score
EdgeThreshold	Sets the threshold at which edges below this value are ignored.
	Default: 2
EdgeType	Select the type of edge to search for: single or pair.
	Default: 1 - Single
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color of an object when it is not accepted.
	Default: Red
FittingThreshold	Specifies the edge results to use for linear fittings.
	Default: 10
Frame	Specifies which positioning frame to use.
	Default: none
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
Graphics	Specifies which graphics to display.
	Default: 1 - All
LabelBackColor	Selects the background color for an object label.
	Default: Transparent
MissingEdgeType	Defines how to handle a missing edge.
	Default: Interpolate
Name	Used to assign a unique name to the Edge object.
	Default: Edge01
NumberOfEdges	Specified the number of edges to be detected.
	Default: 5
PassColor	Selects the color for an object when it is passed.
	Default: LightGreen
PassType	Selects the rule that determines if the object passed.
	Default: SomeFound
Polarity	Specifies whether the ArcFinder object should search for a LightToDark or DarkToLight transition.
	Default: 1 - LightToDark
ScoreWeightContrast	Sets the percentage of the score that depends on contrast.
	Default: 50
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength.
	Default: 50

Property	Description
SearchLength	Defines the length of the edge search range.
	The following SearchLength1, 2 values can be set together.
SearchLength1	Sets the SearchLength1 length in the figure "Positional relationship of CornerFinder object properties".
SearchLength2	Sets the SearchLength2 length in the figure "Positional relationship of CornerFinder object properties".
SearchWidth	Defines the width of the edge search. Range is from 3 - 99.
	Default: 3
SearchWin	Runtime only.
	Sets / returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched in pixels.
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
SearchWinWidth	Defines the width of the area to be searched in pixels.
StrengthTarget	Sets the desired edge strength to search for. Default: 0
StrengthVariation	Sets the amount of variation for StrengthTarget. Default: 0

Positional relationship of CornerFinder object properties

CornerFinder Object Results

The following list is a summary of the CornerFinder object results with brief descriptions. For details on each result, refer to *Vision Guide 7.0 Properties and Results Reference*.

Results	Description
Angle	Returns the angle of the detected corner.
CameraX	Returns the X coordinate of the detected corner in the Camera coordinate system.
CameraY	Returns the Y coordinate of the detected corner in the Camera coordinate system.
Contrast	Returns the contrast of the found Edge.
FitError	Returns the distance between each edge point and straight lines detected as the root mean square (RMS).
Found	Returns whether or not the object was found. (i.e. did the feature or part you are looking at have a shape score that is above the Accept property's current setting.)
MaxError	Returns the maximum difference from the detected line edge in pixel length.
Passed	Returns whether or not the object detection result was accepted.
PixelX	Returns the X coordinate of the detected corner in the Image coordinate system.
PixelY	Returns the Y coordinate of the detected corner in the Image coordinate system.
RobotX	Returns the X coordinate of the detected corner in the Robot coordinate system.
RobotY	Returns the Y coordinate of the detected corner in the Robot coordinate system.
RobotU	Returns the U coordinate of the detected corner in the Robot coordinate system.
ShowAllResults	Displays a dialog showing all results of the specified vision object in a table form.
Strength	Returns the strength of the found edge.
Time	Returns the amount of time in milliseconds required to process the object.

Using CornerFinder Objects

The next few sections guide you through how to create and use a CornerFinder object.

- How to create a new CornerFinder object

- Position and Size the search window

- Configure the properties associated with the CornerFinder object

- Test the CornerFinder object & examine the results

- Make adjustments to properties and test again

Prior to starting the steps shown below, you should have already created a new vision sequence or selected a vision sequence to use. If you have no vision sequence to work with, you can create a new vision sequence by clicking on the <New Sequence [□] > toolbar button.

You can also select a sequence which was created previously by clicking on the Sequence tab in the Vision Guide window and clicking on the drop down list box which is located towards the top of the Sequence tab.

Refer to "*Vision Sequences*" for details on how to create a new vision sequence or select one which was previously defined.

Step 1: Create a New CornerFinder Object

(1) Click on the <CornerFinder [icon] > toolbar button on the Vision Guide 7.0 Toolbar.

(2) Move the mouse over the image display.
      You will see the mouse pointer change to the CornerFinder object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display.

(4) Click the left mouse button to create the object.

(5) Notice that a name for the object is automatically created.
      In the example, it is called "CornerFind01" because this is the first CornerFinder object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a CornerFinder object similar to the one shown below:
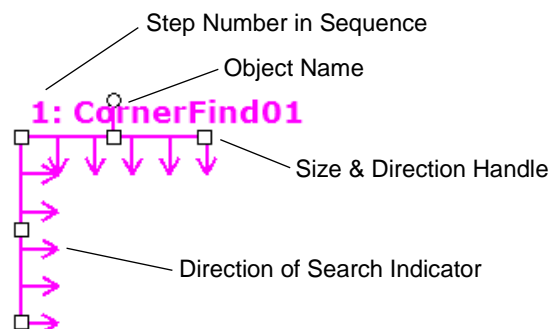
New CornerFinder Object Layout

(1) Click on the name label of the CornerFinder object and, while holding the mouse down, drag the CornerFinder object to the position where you would like the search window to reside.

(2) Resize the CornerFinder object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.)

Step 3: Configuring Properties for the CornerFinder Object

We can now set property values for the CornerFinder object.
To set any of the properties simply click on the associated property's value field and then either enter a new value or if a drop down list is displayed select one of the items in the list.

Shown below are some of the more commonly used properties for the CornerFinder object. For details on other properties such as AbortSeqOnFail and Graphics which are used on many of the different vision objects, refer to *Vision Guide 7.0 Properties and Results Reference*.

EdgeType (Single)	Select the type of the edge to be searched.
	For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.
NumberOfEdges(5)	To find edges, you can search for five edges in each corner of the search window.
Polarity (LightToDark)	Search for edges using "LightToDark" polarity.
	If you are looking for a DarkToLight edge, change "Polarity".

Step 4: Running the CornerFinder Object and Examining the Results

The next few sections guide you through how to run a CornerFinder object.

Click on the <Run Object> button located at the bottom left of the Object tab.
Results for the CornerFinder object will now be displayed. The primary results to
examine at this time are:

Angle Result	Returns the angle of the detected corner in the Image coordinate system.
MaxError Result	Returns the maximum difference from the detected line in pixel length.
PixelX Result **PixelY Result**	Returns the XY coordinate positions of the detected corner in the Image coordinate system.
CameraX Result **CameraY Result**	Returns the XY coordinate positions of the detected corner in the Camera coordinate system. If the calibration is not performed for the XY coordinate positions, "no cal" will be returned.
RobotX Result **RobotY Result**	Returns the XY coordinate positions of the detected corner in the Robot coordinate system. If the calibration is not performed for the XY coordinate positions, "no cal" will be returned.

6.2.20 Contour Object

Contour Object Description

Contour objects outputs the trajectory a workpiece contour is to follow. You can easily obtain the travel route from image information when you wish to move the robot hand along the workpiece contour.

The trajectory of Contour objects can be obtained in three ways. Use each according to the requirements of the application. The means in which the trajectory is acquired can be changed by changing the ContourMode property, with a different GUI and properties available to each mode.

The distinctive features of each mode are explained in brief below.

1. Blob format

    Detects target workpieces in the search window as a blob, acquiring the contour of such. This is used to acquire the contour from complex-shaped workpieces.
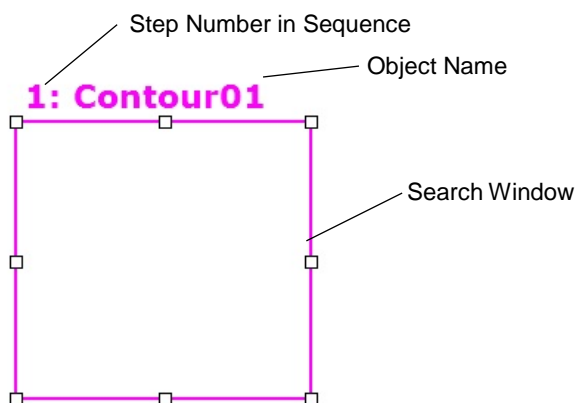
2. Line format

    Obtains the contour using multiple edge search lines arranged horizontally. This is convenient as an easy means of acquiring a contour with little in the way of undulations from a part of the target workpiece.
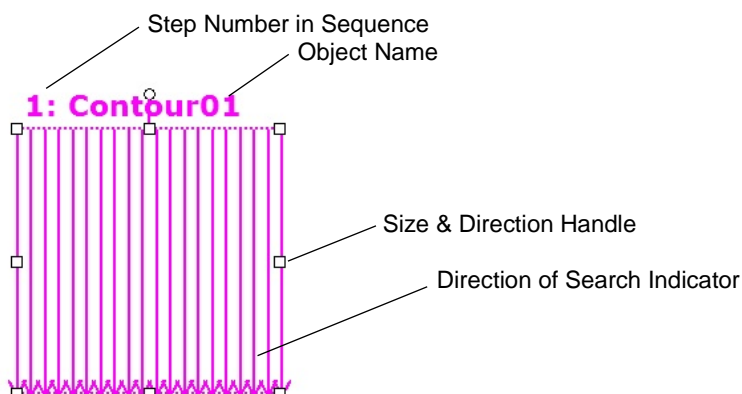
3. Arc format

Obtains the contour using multiple edge search lines arranged radially. This is convenient as an easy means of acquiring an arc-shape contour with little in the way of undulations from a part of the target workpiece.
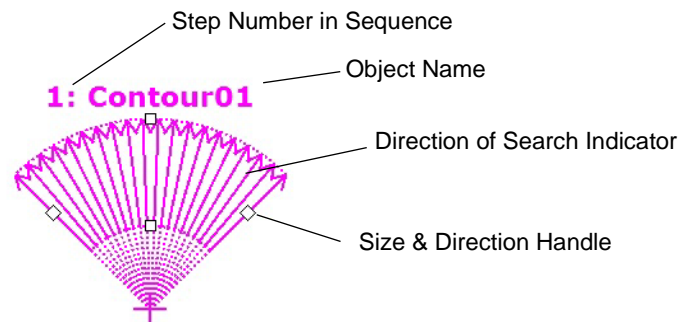
Contour Object Properties

The following list is a summary of the Contour object results with brief descriptions. For details on each result, refer to *Vision Guide 7.0 Properties and Results Reference*.

Further, the properties available will vary based on the ContourMode value. The list shows the ContourMode value required to apply each property.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False Format supported: Blob, Line, Arc
Accept	Specifies the shape score that a feature must equal or exceed to be considered found. Default: 100 Format supported: Line, Arc
AngleEnd	Specifies the end angle of the range to perform a circular search Default: 135 Format supported: Arc
AngleStart	Specifies the start angle of the range to perform a circular search Default: 45 Format supported: Arc
ArcSearchType	Specifies the type of circle/ellipse to be searched for. Format supported: Arc
Caption	Used to assign a caption to the Contour object. Default: Empty String Format supported: Blob, Line, Arc

Property	Description
CenterPointObject	Specifies the position to be used as the center point of the Contour Search.
	This property is normally set to base the center of the Contour Search on the XY position result from another vision object.
	Format supported: Blob, Line, Arc
CenterPntObjResult	Specifies which result to use from the CenterPointObject.
	If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results.
	Default: 1
	Format supported: Blob, Line, Arc
CenterPntOffsetX	Sets or returns the X offset after the center of the search window is positioned with the CenterPointObject property.
	Format supported: Blob, Line, Arc
CenterPntOffsetY	Sets or returns the Y offset after the center of the search window is positioned with the CenterPointObject property.
	Format supported: Blob, Line, Arc
CenterX	Specifies the X coordinate of the position to be used as the center point for the object.
	This property is filled in automatically when the CenterPoint property is set to another vision object.
	Format supported: Arc
CenterY	Specifies the Y coordinate of the position to be used as the center point for the object.
	This property is filled in automatically when the CenterPoint property is set to another vision object.
	Format supported: Arc
ContrastTarget	Sets the desired contrast for the edge search.
	Default: 0 (best contrast)
	Format supported: Line, Arc
ContrastVariation	Selects the allowed contrast variation for ContrastTarget.
	Default: 0
	Format supported: Line, Arc
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one of a like feature within a single search window.
	Format supported: Blob, Line, Arc
Direction	Sets the direction for the edge search.
	Default: InsideOut
	Format supported: Arc

Property	Description
EdgeThreshold	Sets the threshold at which edges below this value are ignored.
	Default: 2
	Format supported: Line, Arc
EdgeType	Select the type of edge to search for: single or pair.
	Default: 1 - single
	Format supported: Line, Arc
Enabled	Specifies whether to execute the object.
	Default: True
	Format supported: Blob, Line, Arc
EndPntObjResult	Specifies which result to use from the EndPointObject.
	Default: 1
	Format supported: Blob, Line, Arc
EndPointObject	Specifies which vision object to use to define the end point of the line to be inspected.
	Default: Screen
	Format supported: Blob, Line, Arc
EndPointType	Defines the type of end point used to define the end point of a line.
	Default: 0 - Point
	Format supported: Blob, Line, Arc
FailColor	Selects the color of an object when it is not accepted.
	Default: Red
	Format supported: Blob, Line, Arc
FillHoles	Specifies whether to fill the holes in a binary image.
	Default: False
	Format supported: Blob
FittingThreshold	Defines the fitting threshold for straight lines and circular arcs.
	Format supported: Line, Arc
Frame	Specifies which positioning frame to use.
	Default: none
	Format supported: Blob, Line, Arc
FrameResult	Specifies which number of the Frame results to be used.
	Default: 1
	Format supported: Blob, Line, Arc
Graphics	Specifies which graphics to display.
	Default: 1 - All
	Format supported: Blob, Line, Arc

Property	Description
LabelBackColor	Selects the background color for an object label.
	Default: Transparent
	Format supported: Blob, Line, Arc
LineDirection	Defines the contour point output direction.
	Default: LeftToRight
	Format supported: Line
MaxArea	Defines the upper Area limit for a defect.
	Default: 100,000
	Format supported: Blob
MinArea	Defines the lower Area limit for a defect.
	Default: 25
	Format supported: Blob
Name	Used to assign a unique name to the Contour object.
	Default: Contour01
	Format supported: Blob, Line, Arc
NumberOfEdges	Specified the number of edges to be detected.
	Default: 20
	Format supported: Line, Arc
NumberToFind	Defines the maximum number of contour points to output.
	Default: 1
	Format supported: Blob, Line, Arc
PassColor	Selects the color for an object when it is passed.
	Default: LightGreen
	Format supported: Blob, Line, Arc
PassType	Selects the rule that determines if the object passed.
	Default: AllNotFound
	Format supported: Blob
Polarity	Specifies whether the Contour object should search for a LightToDark or DarkToLight transition.
	Default_Blob: 1 - DarkToLight
	Default_Line /Arc: 1 - LightToDark
	Format supported: Blob, Line, Arc
RadiusInner	Specifies the inner diameter of the detection range.
	Format supported: Arc
RadiusOuter	Specifies the outline form of the detection range.
	Format supported: Arc

Property	Description
RejectOnEdge	If the property is set to True, the system ignores blobs detected on the edge of the search window.
	Default: 0 - False
	Format supported: Blob
RotationDirection	Specifies the direction of rotation for contour points.
	Default: 0 - CW
	Format supported: Blob, Arc
RuntimeContour	Specifies whether to detect contour points at object runtime.
	Default: True (Detected at runtime)
	Format supported: Blob, Line, Arc
SamplingPitch	Sets the extent to which contour points are reduced.
	Default: 0 (Not reduced)
	Format supported: Blob, Line, Arc
ScoreWeightContrast	Sets the percentage of the score that depends on contrast.
	Default: 50
	Format supported: Line, Arc
ScoreWeightStrength	Sets the percentage of the score that depends on edge strength.
	Default: 50
	Format supported: Line, Arc
SearchWidth	Defines the width of the edge search.
	Range is from 3 - 99.
	Default: 3
	Format supported: Line, Arc
SearchWin	Runtime only.
	Sets / returns the search window left, top, height, width parameters in one call.
	Format supported: Blob, Line
SearchWinAngle	Defines the angle of the area to be searched.
	Format supported: Blob, Line
SearchWinCenterX	Defines the X coordinate value of the center of the area to be searched.
	Format supported: Blob, Line
SearchWinCenterY	Defines the Y coordinate value of the center of the area to be searched.
	Format supported: Blob, Line
SearchWinHeight	Defines the height of the area to be searched in pixels.
	Default: 100
	Format supported: Blob, Line

Property	Description
SearchWinLeft	Defines the left most position of the area to be searched in pixels.
	Format supported: Blob, Line
SearchWinTop	Defines the upper most position of the area to be searched in pixels.
	Format supported: Blob, Line
SearchWinType	Defines the type of the area to be searched (i.e. Rectangle, RotatedRectangle, Circle).
	Format supported: Blob
SearchWinWidth	Defines the width of the area to be searched in pixels.
	Default: 100
	Format supported: Blob
SizeToFind	Selects which size of defects to find.
	Default: 1 - Largest
	Format supported: Blob
ShowModel	Displays the contour teaching model.
	Format supported: Blob
Sort	Selects the sort order used for the results of an object.
	Format supported: Blob
StartPntObjResult	Specifies which result to use from the StartPointObject.
	Default: 1
	Format supported: Blob, Line, Arc
StartPointObject	Specifies which vision object to use to define the start point of the line to be inspected.
	Default: Screen
	Format supported: Blob, Line, Arc
StartPointType	Defines the type of start point used to define the start point of a line.
	Default: "0 - Point"
	Format supported: Blob, Line, Arc
StrengthTarget	Sets the desired edge strength to search for.
	Default: 0
	Format supported: Line, Arc
StrengthVariation	Sets the amount of variation for StrengthTarget.
	Default: 0
	Format supported: Line, Arc

Property	Description
ThresholdAuto	Specifies whether or not to automatically set the threshold value of the gray level that represents the feature (or object), the background, and the edges of the image.
	Default: Disables
	Format supported: Blob
ThresholdColor	Defines the color assigned to pixels within the thresholds.
	Default: Black
	Format supported: Blob
ThresholdHigh	Works with the ThresholdLow property to define the gray level regions that represent the feature (or object), the background, and the edges of the image.
	The ThresholdHigh property defines the upper bound of the gray level region for the feature area of the image.
	Any part of the image that falls within gray level region defined between ThresholdLow and ThresholdHigh will <u>be assigned a pixel weight of</u> 1. (i.e. it is part of the feature.)
	If the ThresholdAuto property is "True" and the ThresholdColor property is "White", this property value will be set to 255 and cannot be changed.
	Default: 128
	Format supported: Blob
ThresholdLow	Works with the ThresholdHigh property to define the gray level regions that represent the feature (or object), the background, and the edges of the image.
	The ThresholdLow property defines the lower bound of the gray level region for the feature area of the image.
	Any part of the image that falls within gray level region defined between ThresholdLow and ThresholdHigh will <u>be assigned a pixel weight of</u> 1. (i.e. it is part of the feature.)
	If the ThresholdAuto property is "True" and the ThresholdColor property is "Black", this property value will be set to 0 and cannot be changed.
	Default: 0
	Format supported: Blob
ContourTolerance	Sets the tolerance when reducing contour points.
	Default: 0
	Format supported: Blob, Line, Arc

Contour Object Results

The following list is a summary of the Contour object results with brief descriptions. For details on each result, refer to *Vision Guide 7.0 Properties and Results Reference*.

Results	Description
CameraX	Returns the X coordinate position of the contour point in the Camera coordinate system.
CameraY	Returns the Y coordinate position of the contour point in the Camera coordinate system.
Found	Returns whether or not the object was found.
NumberFound	Returns the total number of contour points.
PixelX	Returns the X coordinate of the contour point in pixels.
PixelY	Returns the Y coordinate of the contour point in pixels.
RobotX	Returns the X coordinate position of the contour point in the Robot coordinate system.
RobotY	Returns the Y coordinate position of the contour point in the Robot coordinate system.
ShowAllResults	Displays a dialog showing all results of the specified vision object in a table form. This makes it easy to compare results.
Time	Returns the amount of time in milliseconds required to process the object.

Contour Extraction Principle

Broadly, contour extraction can be performed in two different ways. The details of each method are as described below.

1. Create initial contour points

   Detect blob (Blob)

   Contour (Tracing) (Blob)

   Detect edges (Line/Arc)

2. Edit contour points

   Reduce contour points

   Acquire contour

Detect blob (Blob)

If ContourMode is set to "Blob", blobs inside the search window will be detected using the same functionality as that provided by a Blob object. For more details on the principle behind blob detection, refer to the following.

6.2.4 Blob Object - How Blob Analysis Works

For Contour objects, all blobs within the search window will be detected before proceeding to the next step.

Contour (Tracing) (Blob)

If ContourMode is set to "Blob", the step of Contour (Tracing) will be performed once the 'detect blob' step ends. In this step, the contour of the first resulting blob detected in the 'detect blob' step is traced to produce the initial contour points. As such, search window position alignment, and Sort, MinArea, MaxArea and other property configurations need to be setup to ensure the workpiece you wish to acquire the contour of is detected as the first result.

The initial contour points are output as a continuous trajectory without gaps. Note that the initial contour points start from the closest point to the coordinate position specified as StartPointObject, and end at the closest point to the coordinate position specified as EndPointObject.

Detect edges (Line/Arc)

If ContourMode is set to "Line" or "Arc", results of edge search line detection are set as the initial contour points. The initial contour points start from the closest point to the coordinate position specified as StartPointObject, and end at the closest point to the coordinate position specified as EndPointObject.

Reduce contour points

Once the initial contour points are created, unnecessary points are removed. The degree to which contour points are reduced is based on the SamplingPitch and ContourTolerance values set. The SamplingPitch setting is used to determine the number of initial contour points required to acquire a single contour point. For example, if this is set to "10", a single contour point will be extracted from a maximum of 10 initial contour points. Further, the ContourTolerance setting is used to determine the permissible difference when reducing initial contour points. Increase this value to delete a greater number of contour points that do not conform to the outline of the workpiece.

Acquire contour

This step acquires the resulting contour point outline that has been refined in the previous step as result data. When acquiring this data, the RotationDirection / LineDirection settings are used to determine the trajectory direction.

Additionally, when in Blob mode, setting RejectOnEdge to "False" may sometimes result in the workpiece protruding outside the search window. If this happens, the border of the search window is not acquired as the contour. The trajectory will end at the point it touches the search window border frame. If a trajectory is split into multiple acquirable trajectories by the search window, the longest trajectory will be acquired unless StartPointObject and EndPointObject settings have been configured.

Step 1: Create a New Contour Object

(1) Click on the <Contour [icon]> toolbar button on the Vision Guide 7.0 Toolbar.

(2) Move the mouse over the image display.
You will see the mouse pointer change to the Contour object icon.

(3) Continue moving the mouse until the icon is at the desired position in the image display.

(4) Click the left mouse button to create the object.

(5) Notice that a name for the object is automatically created.
In the example, it is called "Contour01" because this is the first Contour object created for this sequence. (We will explain how to change the name later.)

Change ContourMode based on the type of trajectory you wish to acquire. For more information on types of ContourMode available for selection, refer to the table below.

ContourMode setting	Judgment criteria
Blob	Used to acquire trajectories that completely circle the workpiece. Used to acquire the trajectory of complex shapes.
Line	Used to acquire the trajectory of part of a workpiece. Used to acquire simple line trajectories with little undulation.
Arc	Used to acquire the trajectory of part of a workpiece. Used to acquire circular arc trajectories with little undulation.

Step 2: Set the Contour Detection Position (If ContourMode:Blob)

If ContourMode is Blob, you should now see a Contour object similar to the one shown below:
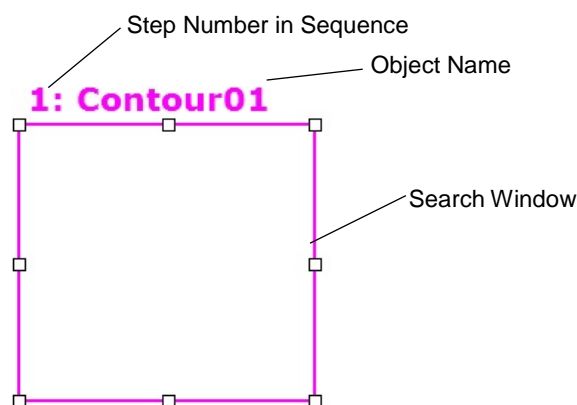
New Contour Object Layout (ContourMode:Blob)

(1) Click on the name label of the Contour object and while holding the mouse down drag the Contour object to the position where you would like the top left position of the search window to reside.

(2) Resize the Contour object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.) (The search window is the area within which we will search for Blobs.)

Step 2: Set the Contour Detection Position (If ContourMode: Line)

If ContourMode is Line, you should now see a LineFinder object similar to the one shown below:
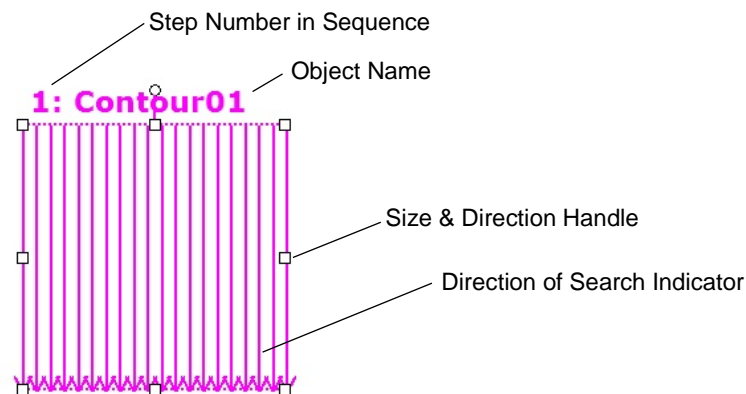
New Contour Object Layout (ContourMode:Line)

(1) Click on the name label of the Contour object and while holding the mouse down drag the Contour object to the position where you would like the top left position of the search window to reside.

(2) Resize the Contour object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.)
Edge object will now be displayed.

Step 2: Set the Contour Detection Position (If ContourMode:Arc)

If ContourMode is Arc, you should now see the ArcFinder object similar to the one shown below:
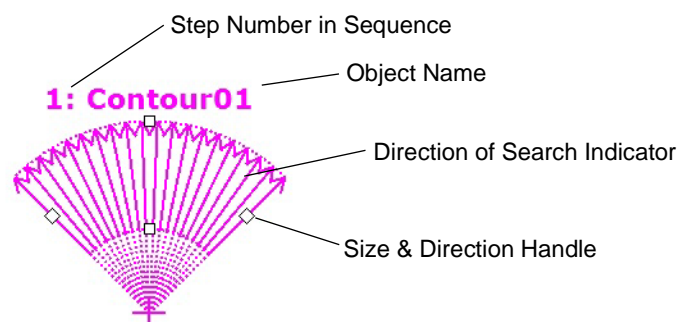
New Contour Object Layout (ContourMode:Arc)

(1) Click on the name label of the Contour object and, while holding the mouse down, drag the Contour object to the position where you would like the search window to reside.

(2) Resize the Contour object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.)

Step 3: Configuring Properties for the Contour Object (ContourMode: Blob)

We can now set property values for the Contour object (ContourMode: Blob).
Shown below are some frequently used properties when ContourMode is set to "Blob".

For details on other properties such as AbortSeqOnFail and Graphics which are used on many of the different vision objects, refer to *Vision Guide 7.0 Properties and Results Reference* or the list of "Correlation Object Properties".

⚠️ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Properly configure MaxArea, MinArea, RejectOnEdge and other properties to reduce the risk of detection errors.

Name property	The default name given to a newly created Contour object is "Contour xx" where xx is a number which is used to distinguish between multiple Contour objects within the same vision sequence. If this is the first Contour object for this vision sequence then the default name will be "Contour 01". To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Contour object's name is displayed is updated to reflect the new name.

Polarity property	Select either one of the following in Polarity property:
	  - detect a dark object on a light background (DarkOnLight)
	  - detect a light object on a dark background (LightOnDark)
	The default setting is DarkOnLight (a dark object on a light background).
	If you want to change it, click on the Value field of the Polarity property and you will see a drop down list with two choices: "DarkOnLight" or "LightOnDark". Click on the choice you want to use.

MinArea, MaxArea	Defines the area of the blob covering the contour extraction area. The default range is set as 25 to 100,000 (MinArea to MaxArea) which is a very broad range.
	This means that most blobs will be reported as Found when you first run a new Blob object before adjusting the MinArea and MaxArea properties. Normally, you will want to modify these properties to reflect a reasonable range for the blob you are trying to find. This way if you find a blob which is outside of the range you will know it isn't the blob you wanted to find.
RejectOnEddge property	Excludes the parts touching the boundary of the search window. Normally, this should be set to True.
RuntimeContour	Defines whether to extract the contour at object runtime.
	Set this to "True" when the shape of the workpiece changes. If the shape of the workpiece does not change, set this to "False" to teach contour information before object runtime. You can check the shape taught by the ShowModel property.

You can test the Blob object now and then come back and set the any other properties as required later.

Step 3: Configuring Properties for the Contour Object (ContourMode: Line)

We can now set property values for the Contour object (ContourMode: Line).
Shown below are some frequently used properties when ContourMode is set to "Line".
For details on other properties such as AbortSeqOnFail and Graphicswhich are used on many of the different vision objects, refer to *Vision Guide 7.0 Properties and Results Reference*.

EdgeType (Single)	Select the type of the edge to be searched.
	For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.
Name property ("Contourxx")	The default name given to a newly created Contour object is "Contour xx" where xx is a number which is used to distinguish between multiple Contour objects within the same vision sequence.
	If this is the first Contour object for this vision sequence then the default name will be "Contour01".
	To change the name, click on the Value field of the Name property, type a new name and hit the return key. You will notice that once the name property is modified, every place where the Contour object's name is displayed is updated to reflect the new name.
NumberOfEdges(1)	You can search for 1 or more edges along the search line.
Polarity (LightToDark)	Search for edges using "LightToDark" polarity.
	If you are looking for a DarkToLight edge, change polarity.

Step 3: Configuring Properties for the Contour Object (ContourMode: Arc)

We can now set property values for the Contour object (ContourMode: Arc).
Shown below are some frequently used properties when ContourMode is set to "Arc". For details on other properties such as AbortSeqOnFail and Graphics which are used on many of the different vision objects, refer to*Vision Guide 7.0 Properties and Results Reference*.

EdgeType (Single)	Select the type of the edge to be searched.
	For edge pairs, an edge is found from each direction and the center of the pair is reported as the position.
Name property ("Contourxx")	The default name given to a newly created Contour object is "Contourxx" where xx is a number which is used to distinguish between multiple Contour objects within the same vision sequence.
	If this is the first Contour object for this vision sequence then the default name will be "Contour01".
	To change the name, click on the Value field of the Name property, type a new name and hit return. You will notice that once the name property is modified, every place where the Contour object's name is displayed is updated to reflect the new name.
NumberOfEdges(5)	You can search for five edges to find circular edges.
Polarity (LightToDark)	Search for edges using "LightToDark" polarity.
	If you are looking for a DarkToLight edge, change polarity.

Step 4: Configure Contour Point Properties

Configure properties to adjust trajectory accuracy, the number of contour points and other settings.

Shown below are properties used. For details on each property, refer to *Vision Guide 7.0 Properties and Results Reference*.

SamplingPitch	Defines the maximum number of contour points reduced.
	When set to "5", the number of contour points is, at most, reduced to around 1/5 of the number of contour points initially detected.
Tolerance	Defines the tolerance when reducing contour points.
	Reduce this value to restrict the number of contour points that run counter to the work shape from being deleted. Check object runtime results to adjust this to the level of accuracy required.
StartPointObject, EndPointObject	Specifies the starting and ending position of a trajectory.
	When set, create Point objects and adjust the vision sequence order to have these run before the Contour object. Place Point objects near the points where you want to start/end the trajectory.
RotationDirection, LineDirection	Specifies the trajectory direction. If ContourMode is set to Blob or Arc, this sets the direction of rotation of the trajectory (CW/CCW). If ContourMode is set to Line, this specifies the direction left or right (LeftToRight/RightToLeft) from the edge search line when facing vertically downwards.

Step 5: Test the Contour Object and Examine the Results

To run testing the Contour object, click on the <Run Object> button located at the bottom left of the object Tab. Results for the Contour object will now be displayed. The primary results to examine at this time are:

PixelX, PixelY	The contour point position detected in pixels.
CameraX, CameraY	The contour point position according to the camera coordinates in millimeters.
RobotX, RobotY	The contour point position according to the robot coordinates in millimeters.
Time result	The amount of time it took for the Contour object to execute.

NOTE ☞ The RobotX, RobotY and CameraX, CameraY results will return "no cal" at this time. This means it is impossible for the vision system to calculate the coordinate results with respect to the Robot coordinate system or Camera coordinate system since the calibration is not executed. For details, refer to *7. Vision Calibration.*

6.2.21 ☐ Text Object

Text Object Description

Text objects display vision object execution results in text form on the screen.

Text Object Layout

The Text object has an image processing window, as shown below.

Text Object Properties

The following list is a summary of the Text object properties with brief descriptions. For details on each property, refer to *Vision Guide 7.0 Properties and Results Reference*.

Property	Description
AbortSeqOnFail	Allows the user to specify that if the object fails (not passed), then the entire sequence is aborted at that point and no further objects in the sequence are processed. Default: 0 - False
CenterPointObject	Specifies the position to be used as the center point for the Geometric Search. This property is normally set to base the center of the Geometric Search on the XY position result from another vision object.
CenterPntObjResult	Specifies which result to use from the CenterPointObject. If All is specified, Frame object will be applied to all of the (NumberFound) for specified vision object results. Default: 1
CenterPntOffsetX	Sets or returns the X offset of the center of the search window after it is positioned with the CenterPointObject.
CenterPntOffsetY	Sets or returns the Y offset of the center of the search window after it is positioned with the CenterPointObject.
Caption	Used to assign a caption to the Text object. Default: Empty String
CurrentResult	Defines which result to display in the Results list on the Object tab or which result to return data for when the system is requested to find more than one of a like feature within a single search window.

Property	Description
Enabled	Specifies whether to execute the object.
	Default: True
FailColor	Selects the color of an object when it is not accepted.
Font	Sets the formatting of the text displayed.
FontBold	Displays the font in bold. (Only available from SPEL+ programs)
FontItalic	Displays the font in italics. (Only available from SPEL+ programs)
FontName	Sets the name of the font. (Only available from SPEL+ programs)
FontSize	Specifies the font point size.
	(Only available from SPEL+ programs)
Graphics	Specifies which graphics to display.
	Default: 1 - All
TextBackColor	Sets the background color for text.
	Default: Transparent
Name	Used to assign a unique name to the Text object.
	Default: Text01
PassColor	Selects the color of an object when it is not accepted.
	Default: LightGreen
ResultObject	Specifies the vision object rendered.
ResultText1	Specifies the results rendered.
ResultText2	Specifies the results rendered.
ResultText3	Specifies the results rendered.
ShowLabel	Specifies whether to add a label for character strings.
SearchWin	Runtime only.
	Sets / returns the search window left, top, height, width parameters in one call.
SearchWinAngle	Defines the angle of the area to be searched.
SearchWinHeight	Defines the height of the area to be searched. (i.e. the search window height)
SearchWinLeft	Defines the left most position of the area to be searched. (i.e. the search window left side)
SearchWinTop	Defines the upper most position of the area to be searched. (i.e. the search window top side)
SearchWinWidth	Defines the width of the area to be searched. (i.e. the search window width)
UserText	Sets a user-defined character string.

Text Object Results

The following list is a summary of the Text object results with brief descriptions. For details on each result, refer to *Vision Guide 7.0 Properties and Results Reference*.

Results	Description
Passed	Returns whether or not the object detection result was accepted.
Found	Returns whether or not the result was acquired.
PixelX	Returns the X coordinate position of the text displayed in pixels
PixelY	Returns the Y coordinate position of the text displayed in pixels

Using Text Objects

Now we have set the foundation for understanding how to use Vision Guide Text objects.

This next section will describe the steps required to use Text objects as listed below:

- How to create a new Text object

- Position and size the search window

- Configure character string settings

- Run a text object and examine the results

Step 1: Create a New Text Object

(1) Click on the <Text [ ] > toolbar button on the Vision Guide Toolbar.

(2) You will see a Text icon appear above the Text object toolbar button.

(3) Click on the Text icon and drag to the image display of the Vision Guide window. Notice that a name for the object is automatically created. In the example, it is called "Text01" because this is the first Text object created for this sequence. (We will explain how to change the name later.)

Step 2: Position and Size the Search Window

You should now see a Text object similar to the one shown below:

Step Number in Sequence

1: **Text01** ——— Object Name

Search Window
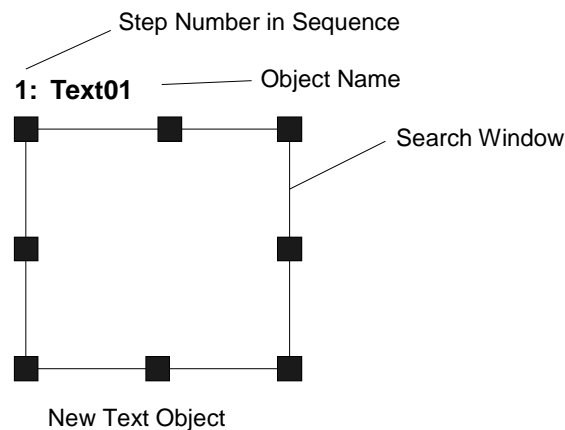
New Text Object

(1) Click on the name label of the Text object and, while holding the mouse down, drag the Text object to the position where you would like the text to appear in relation to the top left corner of the search window.

(2) Resize the Text object search window as required using the search window size handles. (This means click on a size handle and drag the mouse.)
(The character string will appear in relation to the top left corner of the search window)

Step 3: Configure Character String Settings

(1) Specify the vision object containing the result you wish to render in ResultObject. For a vision object to be available for selection in ResultObject, the vision object must be executed before the Text object in the vision sequence.

(2) Select the results you wish to render from ResultText 1 to 3.

(3) Edit the UserText property to display character text other than result strings.

(4) Edit the Font property to adjust the font.
Click the Font property entry field to display the following settings window. Here you can configure the formatting, style and size of the font.
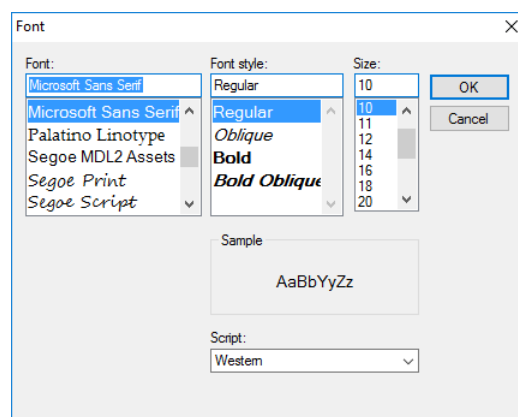
Step 4: Running the Text Object and Examining the Results

Click on the <Run Object> button located at the bottom left of the Object tab. Character strings set as Text objects will appear on the screen. Adjust the property settings if there is a problem with the text shown.

Character Strings Displayed as Text Objects

The formatting of character strings displayed as Text objects is as follows.
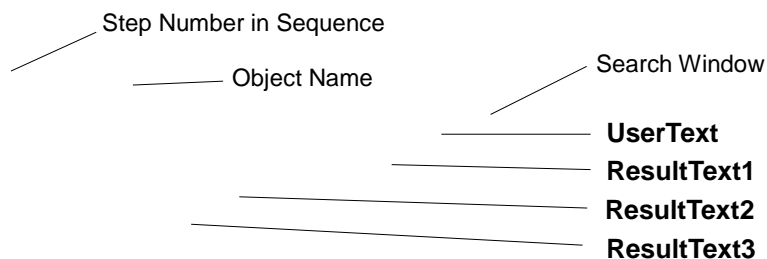
Character strings will appear in the order of UserText, ResultText1, ResultText2, ResultText3 from the top down. This will not appear if the UserText field is left blank, and ResultText 1 - 3 values are set to "None". If character text cannot fit within the search window, overlapping text will not be shown.

6.2.22 Working with Multiple Results from a Single Object

Blob, Geometric, Edge, Correlation, and DefectFinder objects can be used to find more than only 1 feature within a single search window. The properties and results that you will need to understand for working with multi-result vision objects are listed below:

- CurrentResult property

- NumberToFind property

- NumberFound result

- ShowAllResults result

The Default and Maximum Number of Features a Vision Object Can Find

The default property configurations for a Blob, Geometric, Edge, Correlation, or DefectFinder object cause it to find only 1 feature within a search window. This is because the NumberToFind property is set to 1. However, when you set the NumberToFind property to a number larger than 1, the vision object will attempt to find as many features as you specify.

If the NumberToFind property is set to All, the search will continue until the maximum detection number (100) of the object will be reached.

Set the NumberToFind property and run your vision object. You will see that several of the features in the current image that meet the acceptance criteria will be shown as found (green box with crosshair indicating position returned for found object). You can also see that the feature found which is considered the CurrentResult is highlighted in a lighter shade of green than the other features.

The Sorting Order of the Features that are Found

When a multi-result object is first run, the CurrentResult is automatically set to 1 indicating that the 1st result for the multi-result vision object should have its results displayed in the Results list on the Object tab. For a Correlation or Geometric object, the first result is the one with the highest score (as compared to all the features found for that Correlation object). The second result is the one with the second highest score and so on.

For a Blob and DefectFinder objects, the first result is the one returned based on the values for the SizeToFind and Sort properties. (For example, if SizeToFind is set to "Largest" then the first result will be the largest object found.) See the SizeToFind and Sort properties in the *Vision Guide 7.0 Properties and Results Manual* for more info regarding sorting results.

Examining the Vision Object's Multiple Results

If you look closely at the Results list heading you can see where it says something like "Result 1 of 10". (Assume for the sake of this discussion that we had set the NumberToFind property to 10 for a Blob object.) This means the CurrentResult is 1 out of 10 features that were searched for (as defined by the NumToFind property.) It is important to remember that the 2nd number in the Results list heading is the value of the NumToFind property and not the number of features that were actually found.

You can examine the results for any of the multiple blobs you tried to find by changing the CurrentResult property to reflect the blob you would like to see. This can be done by manually typing a number into the value field of the CurrentResult property or by moving the cursor to the value field of the CurrentResult property and then using the SHIFT+DnArrow or SHIFT+UpArrow keys to move through the results. You will see the results displayed in the Results list. You can always tell which result is displayed in the Results list by either looking at the CurrentResult property or by just looking at the heading of the Results list. It will display "Result 1 of 10" for the 1st result when NumToFind is set to 10 and "Result 2 of 10" for the 2nd result, and "Result 3 of 10" for the 3rd result, etc.

⚠️ WARNING	■ When the CurrentResult property is changed, this also changes the VGet result in a program. For example, if a SPEL program uses VGet to get the RobotXYU result from a vision object, and the CurrentResult is set to 3, then the VGet instruction will return the RobotXYU 3rd result.
	Be careful with this. As you may end up getting the wrong results because the CurrentResult was set wrong for your SPEL⁺ program.

Tips: To acquire multiple results, specify the result numbers explicitly by the result names specified in VGet command.
Example: VGet seqname.objname.RobotXYU(**1**), found, X, Y, U

Using the NumberFound Result

The NumberFound result is useful because it displays how many blobs or matching features for a Correlation Model were actually found. This result is also available from the SPEL$^+$ language so you can use a SPEL$^+$ program to make sure that the proper number of results were found, or to count how many were found, or a whole host of other things. See the examples below:

This shows a small section of code to check if the number of results found is less than 5.

```
VGet seqname.objname.NumberFound, numfound
If numfound < 5 Then
     'put code to handle this case here
```

Consider a situation where vision is used to find as many parts in one VRun as it can. The robot will then get each part until all the found parts are moved onto a conveyor. This example shows a small section of code to have the robot pick each of the parts found and then drop them off one by one onto the same position on a moving conveyor.
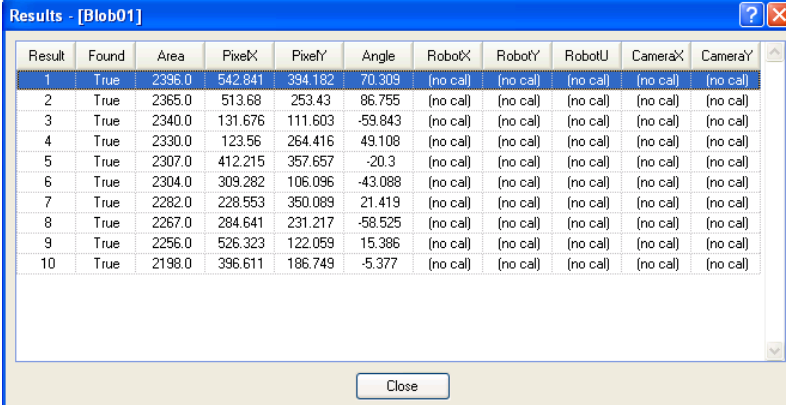
```
VRun seqname
VGet seqname.objname.NumberFound, numfound
For i = 1 to numfound
  VGet seqname.objname.RobotXYU(i), found, X, Y, U
  If found = True Then
   'Set coordinates found from vision
    VPick = XY(X, Y, -100.000, U)
  Else
    Print "Vision Error: part not found"
  EndIf

  Jump Vpick          'Jump to Vision Pickup position
  On Gripper          'Turn on vacuum
  Wait .1
  Jump Vconvey        'Jump to Conveyor Position to drop part
  Off Gripper         'Turn off vacuum
  Wait .1
Next count
```

Examining All the Multiple Results at Once

One of the nicest features built in with the multiple results feature for Blob, and Correlation objects is the ShowAllResults result. There may be cases where you want to compare the different results to see the top Score vs. the 2nd Score and so on. For example, maybe there is a big drop off in score after the 3rd feature is found. Using the ShowAllResults result makes is easy to see all results at once.

Clicking on the ShowAllResults result's value field will cause a button to appear. Click on the button and a dialog will be displayed which shows all the results for the current vision object. A sample ShowAllResults dialog is shown in the figure below

Result	Found	Area	PixelX	PixelY	Angle	RobotX	RobotY	RobotU	CameraX	CameraY
1	True	2396.0	542.841	394.182	70.309	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
2	True	2365.0	513.68	253.43	86.755	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
3	True	2340.0	131.676	111.603	-59.843	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
4	True	2330.0	123.56	264.416	49.108	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
5	True	2307.0	412.215	357.657	-20.3	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
6	True	2304.0	309.282	106.096	-43.088	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
7	True	2282.0	228.553	350.089	21.419	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
8	True	2267.0	284.641	231.217	-58.525	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
9	True	2256.0	526.323	122.059	15.386	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)
10	True	2198.0	396.611	186.749	-5.377	(no cal)	(no cal)	(no cal)	(no cal)	(no cal)

Using the Multiple Results Dialog to Debug Searching Problems

Sometimes the parts which you are working with vary considerably (even within the same production lot) and sometimes there are 2 or more features on a part which are similar. This can make it very difficult to determine a good Accept property value. Just when you think you have set the Accept property to a good value, another part will come in which fools the system. In these cases it can be very difficult to see what is going on.

The Show All Results dialog was created to help solve these and other problems. While you may only be interested in 1 feature on a part, requesting multiple results can help you see why a secondary feature is sometimes being returned by Vision Guide as the primary feature you are interested in. This generally happens a few different ways:

1. When 2 or more features within the search window are very similar and as such have very close Score results.

2. When the Confusion or Accept properties are not set high enough which allow other features with lower scores than the feature you are interested in to meet the Accept property setting .

Both of the situations above can be quite confusing for the beginning Vision Guide user when searching for a single feature within a search window.

If you have a situation where sometimes the feature you are searching for is found and sometimes another feature is found instead, use the Show All Results dialog to home in on the problem. Follow the following steps to get a better view of what is happening:

(1) Set your NumberToFind property to 3 or more.

(2) Run the vision object from the Vision Guide Development Environment.

(3) Click on the ShowAllResults property button to bring up the Show All Results dialog.

(4) Examine the scores of the top 3 or more features that were found.

(5) If only 1 or 2 features were found (*Vision Guide* will only set scores for those features that are considered found) reduce your Accept property so that more than 1 feature will be found and Run the vision object again. (You can change the Accept level back after examining the Show All Results dialog)

(6) Click on the ShowAllResults property button to bring up the Show All Results dialog.

(7) Examine the scores of the top 3 or more features that were found.

Once you examine the scores of the top 3 or more features that were found as described above, it should become clear to you what is happening. In most cases you will see one of these two situations.

1. Each of the features that were found has a score greater than the Accept property setting. If this is the case, simply adjust your Confusion property value up higher to force the best feature to always be found rather that allowing other features to be returned because they meet the Accept threshold. You may also want to adjust the Accept property setting.

2. Each of the features are very close in score. If this is the case, then you will need to do something to differentiate between the feature which you are primarily interested in such as:

    - Readjust the search window so the features that are randomly returning as the found feature are not contained inside.

    - Teach the Model again for the feature that you are most interested in.

    - Adjust the lighting for your application so the feature that you are most interested in gets a much higher score than the other features that are currently fooling the system.

Accessing Multiple Results from the SPEL+ Language

We already explained that the CurrentResult property is used to set which result will be displayed in the Results list. It is also used to determine what number of results to return results for. In other words if we want to get the Area results from the 3rd result returned from a Blob object, CurrentResult must be set to 3. You have already seen how this can be done from the Object tab Properties list. Now let's take a look at how to access multiple results from SPEL+.

Multiple result access from SPEL+ treats the result like a type of array where the CurrentResult is referenced with a subscript number next to the result to be obtained. The first example below shows how to get the third Area result and put it in a variable called area from the SPEL+ Language.

```
VGet seqname.objname.Area(3), area
```

The 2nd example below shows how to get that same 3rd Area result but this time assign it as the value of the 3rd element in an array called Area().

```
VGet seqname.objname.Area(3), area(3)
```

Variable names can also be used to represent the element in an array rather than fixed elements as in example #2 above. Notice that the variable called "var" is used as the subscript for the Area result.

```
VGet seqname.objname.Area(var), area(var)
```

The 4th example assumes that you have used a single vision object to find multiple like parts (let's say up to 10). You would now like to pick these parts (let's say they are pens) with a robot so you will need to store the X, Y, and U coordinates to variables to represent the coordinate values of each of the parts found. The following code could pull these coordinates out of the RobotXYU result and put them into X, Y and U arrays that could later be used for the robot to move to.

```
Function test
  Boolean found(10)
  Integer numfound, i
  Real X(10), Y(10), U(10)

  Jump camshot    'move camera into position snap shot

  VRun seq01      'run the vision sequence to find the pens

  VGet seq01.blob01.NumFound, numfound     'how many found

  For i = 1 to numfound    'get robot coords
    VGet seq01.blob01.RobotXYU(I), found(i), X(i), Y(i), Z(i)
  Next i
  'Add code for robot motion here……
Fend
```

6.2.23 Automatic Multiple Object Search

You can use automatic multiple object search for several vision objects. Objects for searching are created automatically when you specify to use All results from another object. This allows you to configure one or more objects to search for a feature, and at runtime, the objects are created and run automatically for all results of the parent object.

Automatic multiple object search can be used with CenterPointObject and Frame.

Here is a simple example using CenterPointObject:

1. Create a sequence and add a Blob object. Set NumberToFind to 10.

2. Create a Polar object. Set CenterPointObject to the Blob object, and set CenterPntObjResult to All.
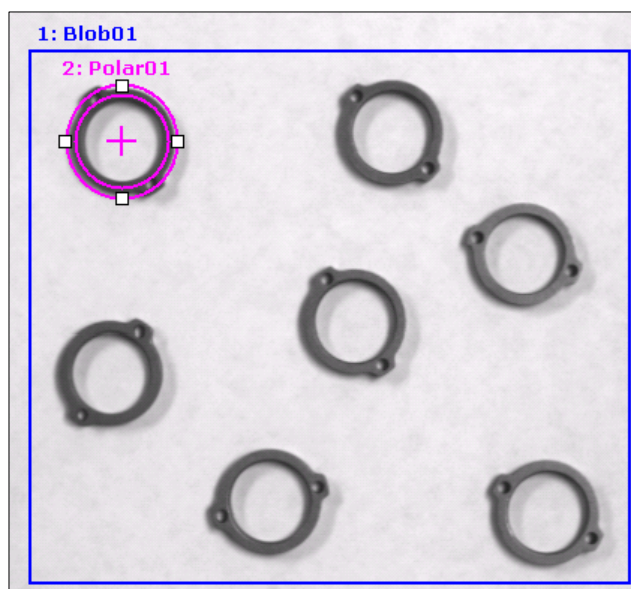
3. Teach the Polar object.

4. Run the sequence. For each blob found, an instance of the Polar object is created and run. If 10 blobs were found, then there will be 10 Polar objects, each centered on a result from the Blob object. In the picture below, you can see seven Polar objects, one for each blob found.
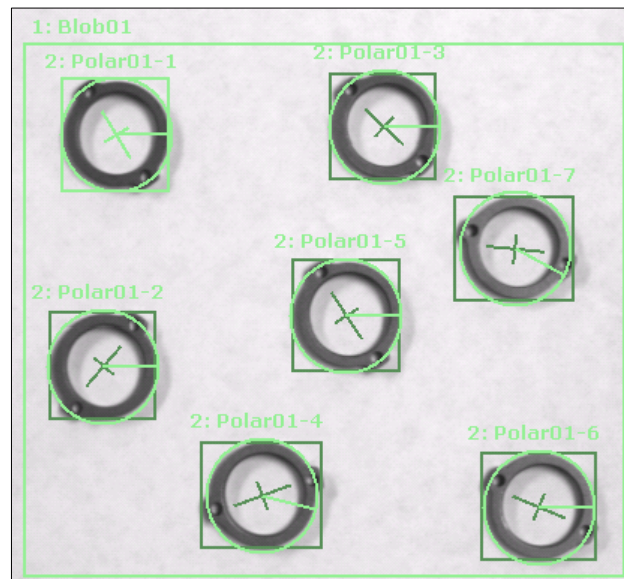
When using automatic multiple object search, if a child object can find multiple results, only one result can be found for each instance of the child object.

You can also use Line, Edge, and LineInspector objects with automatic multiple search. You must specify both the StartPointObject and EndPointObject, along with StartPntObjResult = All and EndPntObjResult = All.

For example:

1. Create a new sequence and select ImageFile with two horizontal rows of blobs (see image below).

2. Create Blob01, set NumberToFind = 3, and Sort = PixelX. Size and position it to find the first row of blobs.

3. Copy Blob01 and paste to create Blob02. Size and position it to find the second row of blobs.

4. Create a Line object, set StartPointObject = Blob01, and set StartPntObjResult = All. Set EndPointObject = Blob02, and set EndPntObjResult = All.
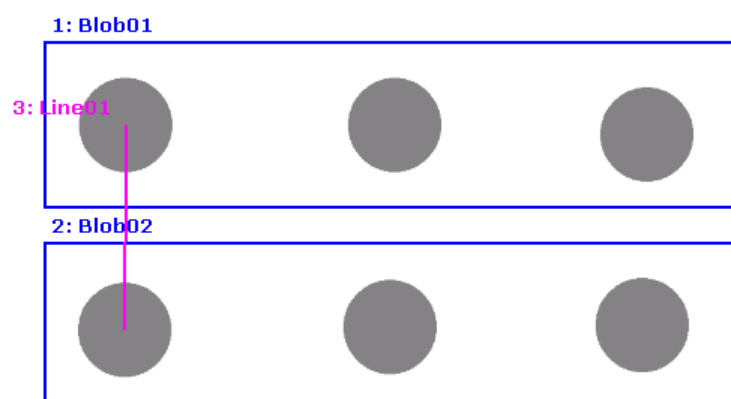
5. Run the sequence. A instance of the Line object will be created for each pair of results for StartPointObject and EndPointObject.
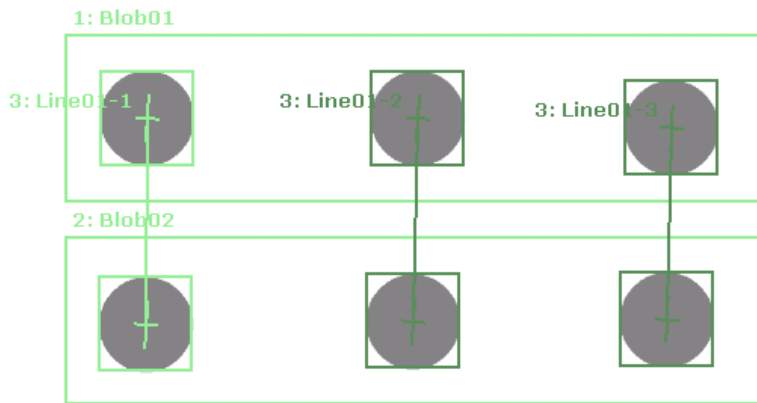
6.2.24 Turning All Vision Object Labels On and Off

Force Labels Off (Vision Toolbar Only)

The Force Labels Off feature is a useful method to reduce screen clutter when working with many vision objects in one Sequence. The Force Labels Off button on the Vision Guide 7.0 toolbar is a two-position button. When pressed in, the labels for all vision objects are turned off, thus making the objects which are displayed easier to see. When the Force Labels Off is in the out position (not pressed in), then labels are shown for each vision object that is displayed in the image display.

The Force Labels Off feature is sometimes used in conjunction with the Force All Graphics On toolbar button. When the Force all Graphics On toolbar button is pressed, you can still Force Labels Off. This means that even though the Force All Graphics On toolbar button is pressed in, the labels still will not be displayed because the Force Labels Off button is also pressed in.

NOTE

If you are working in a vision sequence where you just turned the Force Labels Off button Off (i.e. it is in the out position), and you still cannot see a specific vision object, chances are that the Graphics property for that vision object is set to None. This means don't display graphics at all for this object and may be your problem.

The Force Labels Off toolbar button is dimmed (made light gray in color and inaccessible) if there are no vision sequences for the current project.

6.2.25 Turning All Vision Object Graphics On

Force All Graphics On (Vision Toolbar Button Only)

The Force All Graphics On toolbar button provides a quick method to turn all graphics (search window, model origin, model window, Lines, and Labels) for all vision objects in the current vision sequence On with one button click. This button overrides the setting of the Graphics property for each individual vision object making it easy to quickly see all vision objects rather than modifying the Graphics property for each vision object individually.

NOTE

☞

The Force Labels Off feature is sometimes used in conjunction with the Force All Graphics On toolbar button. However, it is important to note which button has precedence. When the Force all Graphics On toolbar button is pressed, you can still Force Labels Off. This means that even though the Force All Graphics On toolbar button is pressed in, the labels still will not be displayed because the Force Labels Off button is also pressed in.

It should be noted that the Force All Graphics On toolbar button is dimmed (made light gray in color and inaccessible) if there are no vision sequences for the current project.

6.2.26 Showing Only the Current Object

Show Only Current Object (Vision Toolbar Button Only)

When there are many objects in a sequence, sometimes it is difficult to select and work with the desired object. By clicking the Show Only Current Object toolbar button, then only the current active object is displayed. To display all objects again, click the Show Only Current Object again. When only the current object is displayed, you can select which object to display by selecting it in the Objects list.

7. Vision Calibration

By using the Vision Calibration, you can register the positional relationship between the cameras and the robot system to the Robot Controller, and convert the image coordinate position of the object recognized by the image processing into the robot coordinate position.

Vision calibration is performed by the following steps:

- Camera installation

- Create vision sequences for calibration

- Create a Calibration using a calibration wizard.

- Point teaching

- Execute Calibration

In order to perform calibration with high accuracy, correction of lens distortion and camera tilt (difference between the work plane and the camera's optical axis) can be performed.

Correction of lens distortion and camera tilt is performed by the following steps:

1. Camera installation

2. Create vision sequence for lens distortion and camera tilt correction

3. Execute correction of lens distortion and camera tilt

4. Create vision sequences for calibration

5. Create a Calibration by calibration wizard

6. Point teaching

7. Execute Calibration
8.

Additional calibrations using a camera
- Local plane can be detected for the calibration plate placed on the work plane.
  (Only mobile camera mounted on Arm #6 (J6) on the vertical 6-axis robot.)

- The tool coordinates of the camera mount position for the mobile camera mounted on Joint #4 (J4) on the horizontal articulated (SCARA) robot or Arm #6 (J6) on the vertical 6-axis robot can be automatically detected.
  Also, the camera mount position of the mobile camera mounted on the Arm #2 (J2) on the horizontal articulated (SCARA) robot can be automatically detected as parameters for additional arm setting.

- The tool coordinates of a tool mounted on the end effector of the robot can be automatically detected using a fixed camera.

The wizards for these calibrations can be run by clicking on [Locals], [Tools], or [Arms] tabs on the [Robot Manager]. You can also execute these wizards from the camera calibration wizard.

In all cases, you need to create a required vision sequence in advance.

Configuration for local detection function using a camera

Hardware Configuration	Local detection function using a camera
Compact Vision CV1	No
Compact Vision CV2-S, CV2-H (Firmware 2.x.x.x)	No
Compact Vision CV2-SA, CV2-HA (Firmware 3.0.0.0 or later)	OK
PC VisionPV1	OK

Other vision calibrations other than the above are available for any hardware configuration (tool detection using by camera and arm detection are included).

An optional calibration plate is necessary when using the local detection function using a camera.

Refer to the specification table below to select the proper calibration plate according to the largeness of the work surface.

Calibration Plates

	Large (CP01-L)	Medium (CP01-M)	Small (CP01-S)	Extra Small (CP01-XS)
External dimension (Width×Depth×Thickness) [mm]	152×152×2	78×78×2	40×40×2	28×28×2
Usable dimension (Width×Depth) [mm]	128×128	64×64	32×32	21×21
Dot pitch (Width / Depth) [mm]	4	2	1	0.5
Number of dots	1089 (Width 33×Depth 33)			1849 (43×43)
Material	White glass			

7.1 Camera Installation

Select a camera installation for each calibration. Calibration methods vary (requirements for vision sequences, calibration procedure) depending on the camera installation methods.

Note that the wrong setting may result in improper calibration.

Vision Guide 7.0 supports the following camera installations:

Camera Installation	Description
Mobile J2	Camera is mounted on Joint #2 on SCARA robot or Cartesian robot.
Mobile J4	Camera is mounted on Joint #4 on SCARA robot or Cartesian robot.
Mobile J5	Camera is mounted on Joint #5 on 6-Axis robot.
Mobile J6	Camera is mounted on Joint #6 on 6-Axis robot.
Fixed downward	Camera and target objects do not move and are fixed in the work envelope of the robot. The vision system uses the camera to acquire position information in the robot coordinate system. If the camera is not installed vertically to the XY plane of the specified coordinate system, correction of lens distortion and camera tilt must be performed. Specified coordinate systems are robot base and local coordinate systems.
Fixed upward	Camera does not move and is looking up into a portion of the robot work envelope. For example, this installation method is used to check the position of the object which is carried by the robot. The calibration target is on the end effector or the object held by the robot.
Standalone	Camera can be installed anywhere. Camera does not have a relation with the robot. With this method, position information in the Robot coordinate system cannot be acquired. However, it can be converted from the image coordinate system to the camera coordinate system. That is, simple length measurement can be performed.

Tips:

Correction of lens distortion and camera tilt can be configured for all camera installation.

7.2 Correction of Lens Distortion and Camera Tilt

In order to increase the detection success of Geometric and Correlation objects, to operate the robot with higher accuracy, and to measure the dimensions with higher accuracy, lens distortion and camera tilt can be corrected.

Position the square lattice pattern like the sample shown below so that it fills the entire field of view, and then perform lens distortion calibration. The lattice pattern must be square. The optional calibration plate can also be used.

This square lattice pattern is essential for distortion correction.
Prepare a lattice pattern with minimum distortion as possible and place it on the work plane in the camera's field of view. When placing the lattice pattern, also consider its working distance (WD) from the camera. It is necessary to place the pattern at the same height as the work piece height where the actual work piece will be detected.

The angle of the lattice pattern is not necessary to be aligned to that of the camera's field of view.
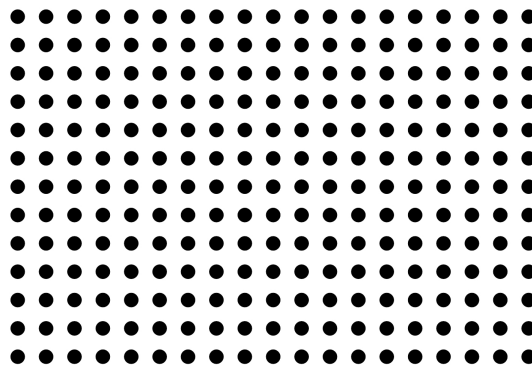
It is important to set the square lattice pattern so that it fills the entire field of view of the camera. If calibration is performed while the lattice pattern is in only a part of the field of view, distortion cannot be corrected properly since the area with no lattice pattern is not taken into account.

When executing calibration, make sure that at least 100 points of the lattice pattern are within the field of view.

After completing correction of lens distortion and camera tilt, subsequent camera images will be corrected when they are captured, and images with less distortion can be obtained.

Tips:

When executing calibration, an error may occur depending on how the lattice pattern is captured. In this case, calibration can be done properly by adjusting the lattice pattern and executing calibration again.
Also, set the camera so that the angle between camera's optical axis and the lattice pattern is 45 to 90 degrees. If the camera angle is 45 degrees or less, calibration will become an error.

7.3 Reference Points and Camera Points

Reference points are important points to be used to calibrate the relation of the image coordinates and the Camera or Robot coordinate systems.

Camera points are points used to capture an image for detecting a target.
Calibration associates the coordinates with the reference points. The reference points and camera points are configured by jogging the robot in the point teaching mode.

Required reference points and camera points vary depending on camera mount types.

The reference points specified by using a robot have the following three types.

TaughtPoints (Teach point)
EndEffector (End effector)
UpwardCamera (Upward camera)

The reference points and camera points can be selected according to the camera mount type as shown in the table below. For Standalone (fixed camera) calibration which does not use a camera, you manually enter the coordinate values (horizontal and vertical distance between reference points) of the reference points into the system.

Types of reference points by camera mount type

Camera mount type	Selectable reference point type	Reference point teach method	Target detecting method	Required vision sequence
Mobile Camera	TaughtPoints	One-point teaching	Nine-camera-point teaching*	Detect one target
	UpwardCamera	Unnecessary	Nine-camera-point teaching*	Detect one target
Fixed downward Camera	TaughtPoints	Nine-point teaching	One-shot nine-point detection	Detect nine targets
	EndEffector	Unnecessary	Nine-camera-point teaching*	Detect one target
Fixed upward Camera	EndEffector	Unnecessary	Nine-camera-point teaching*	Detect one target
Standalone	-	Nine-point coordinate specification	One-shot nine-point detection	Detect nine targets

* Automatic camera point generation is available. (Refer to the description below)

Reference points teaching method

When "TaughtPoints (teach point)" is set as a reference point type, jog the robot so that positions of the tool and the target match, and teach one or nine points.
How to set reference points for a mobile camera:
Refer: *7.3.1 Mobile camera reference points*

How to set reference points for Fixed Downward and Standalone camera:
Refer: *7.3.2 Fixed Downward and Standalone camera reference points*

Nine-camera-point teaching

When "nine-camera-point teaching" is used for detecting the target, a total of nine positions are detected with one point for each image capturing.

In the calibration, move the target to the specified nine areas in the field of view of the camera and capture images to detect the target.

In the point teaching mode, jog the robot to teach points while checking the images so that the nine camera points can be detected at a proper position in each image.

How to create a sequence:
    Refer: *7.4.1 Vision Sequence for Detecting One Target*

Automatic Camera point generation

When "nine-camera-point teaching" is used for detecting the target, the automatic camera point generation is available.
Instead of teaching nine camera points, eight camera points can be generated automatically by teaching only one point at the center of the field of view.

Note: When using the automatic camera point generation, the robot moves automatically to generate camera points during the calibration. Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during automatic camera point generation.

One-shot nine-point detection

When "one-shot nine-point detection" is used for the calibration, positions of nine targets are detected in one shot.

It is necessary to create a vision sequence for detecting nine targets by a camera in advance.

How to create a sequence:
    Reference: *7.4.2 Vision Sequence for Detecting Nine Targets*

7.3.1 Mobile Camera Reference Points

This scheme requires one reference point.

You can specify a taught point (reference point type="TaughtPoints") that you manually teach by jogging the robot, or you can specify a point (reference point type= "UpwardCamera") that is found with an upward looking camera. The later is preferred for greatest accuracy and automation.

Here are some examples of taught reference points:

- A part or calibration target in the robot work envelope.
- A hole somewhere in the work envelope that a tool mounted on the robot end effector can be slipped into.

When using an upward camera to find the reference point, there needs to be a thin plate with hole through it that can be seen by both the mobile camera and the upward looking camera. During calibration, the upward camera (which has been previously calibrated) locates the reference hole in the plate. Then the mobile camera is calibrated by searching for the reference hole in nine positions.

Using an upward camera to find the reference point for mobile camera calibration is more accurate because during calibration of the upward camera, the robot tool is rotated 180 degrees for each camera point so that the precise center of the U axis of the robot can be determined. This allows the calibrated upward camera to more accurately find the reference hole that the mobile camera will locate during calibration.
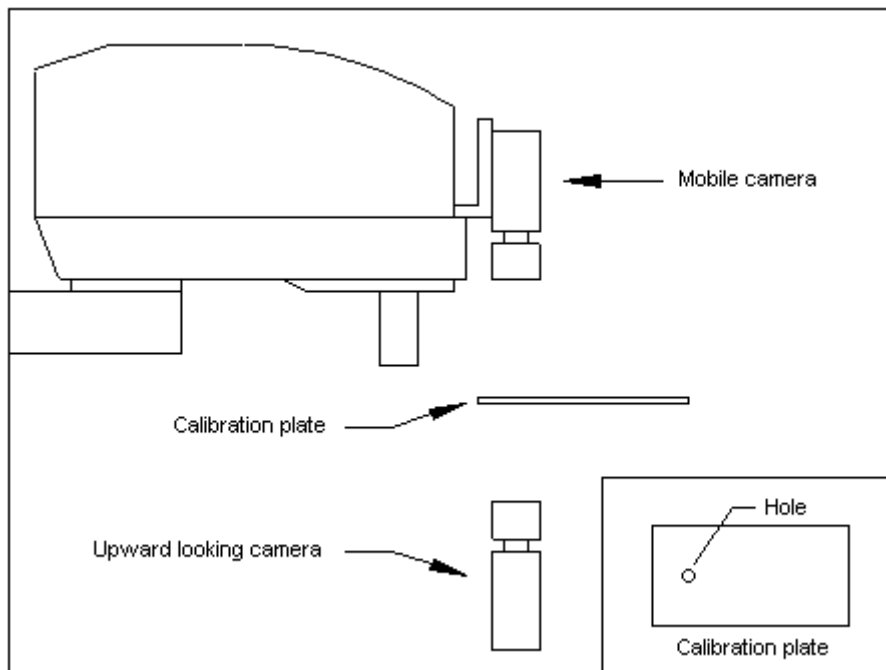
Mobile Camera Calibration using Upward Camera

7.3.2 Fixed Downward and Standalone Camera Reference Points

The "Fixed Downward" and "Standalone" calibration schemes for which the reference point type is set to "TaughtPoints (teach point)" require a calibration target plate or sheet that contains nine targets. The figure below shows the pattern for the nine calibration targets.
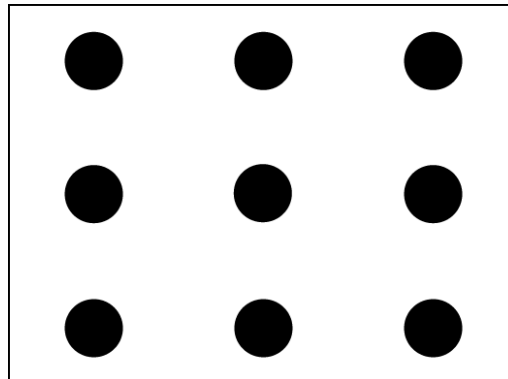
Fixed Camera Calibration Targets

For 'Fixed Downward' calibrations, the targets could be holes in a plate that a rod on the robot end effector can be slipped into. The distances between the targets do not have to be exact. When the points are taught, the calibration software will read the robot positions.

For Standalone cameras, a pattern on paper, mylar, etc. can be used. The horizontal and vertical distances between the targets must be known. These are entered when you teach the points from the [Calibration] dialog.

7.3.3 Teaching by TwoRefPoints

When teaching reference points for mobile and fixed downward calibration, you will be prompted to teach the point and then optionally teach it again at 180 degrees from the first position. This allows the system to determine more accurate location of the reference point in the robot world. However, if you are using a robot tool that has been accurately defined and specified in the calibration setup, you can skip teaching the point at 180 degrees. To skip the 180-degree steps, ensure that the TwoRefPoints calibration property is set to False. For details on defining tools, see Defining Tools in *Using Vision Guide 7.0 with SPEL+*.

7.4 Creating Vision Sequences for Calibration

Before you can calibrate a camera, you must create a vision sequence that can find the calibration target(s).

7.4.1 Vision Sequence for Detecting One Target

When the mobile camera or fixed downward/upward camera whose reference point type is "EndEffector" is used, the sequence will search for one calibration target in the entire field of view.

Be sure to define the size of the search window as large as the entire image display.

The X and Y results for the last object in the sequence will be used by the calibration software during calibration.
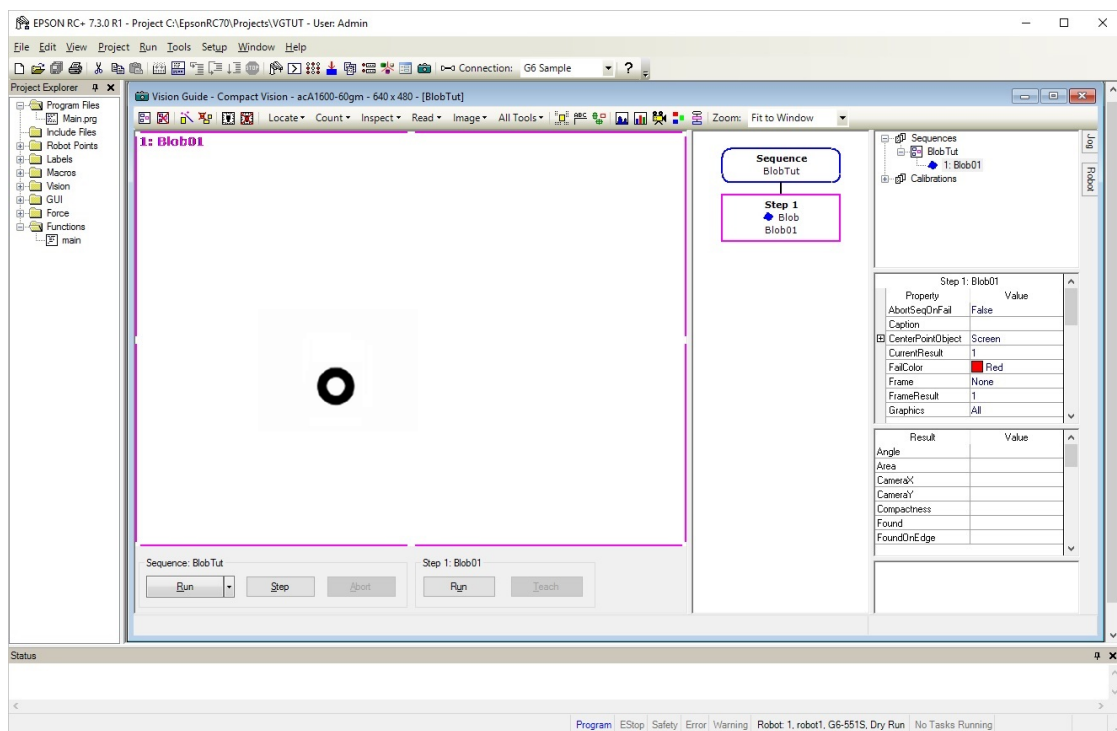
7.4.2 Vision Sequences for Detecting Nine Targets

Vision sequences necessary for calibration of fixed downward or standalone camera whose reference point type is "TaughtPoints (teach point)" must locate nine calibration targets. There are two methods to do this.

1.   Create an object for each target, for a total of nine objects.

OR

2.   Use one of the multi-result objects (Blob or Correlation objects, etc.) to return nine results. Set NumberToFind to "9", so that nine results will be returned.
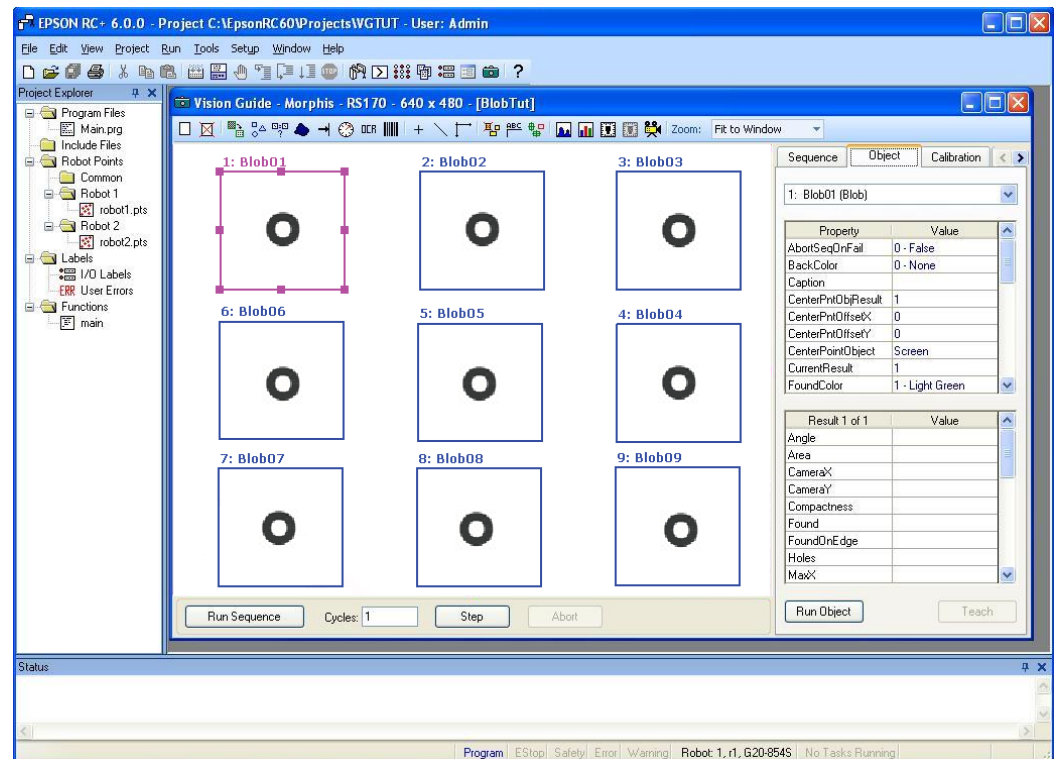
Vision objects placed in order for Standalone Calibration

With the first method described above, the objects must be placed on the image window in the same order that the targets will be taught. The first object will be in the upper left corner, the second at top center, the third in the upper right corner, etc.

For 'Fixed Downward' calibrations, the order for the middle row of objects is in reverse order: 6, 5, 4. This is because moving to the positions in the order 1, 2, 3, 6, 5,4, 7, 8, 9 produces the most efficient motion from the robot and helps make calibration complete faster.

7.4.3 Vision Sequence for Distortion Correction

To perform distortion correction, a vision sequence needs to be created in the following steps in advance.

1. Set a square grid pattern.

2. Create a sequence to detect the square grid pattern.

3. Select Lamp property as necessary.

4. Create a Blob object and expand a search area to the entire window.

5. Set NumberToFind property to "All".

6. Set RejectOnEdge property to "True".

7. Set ThresholdAuto, ThresholdHigh, ThresholdLow, MinArea, MaxArea and other properties to detect the dots.

8. Execute the created sequence. Check if more than 100 points can be detected.
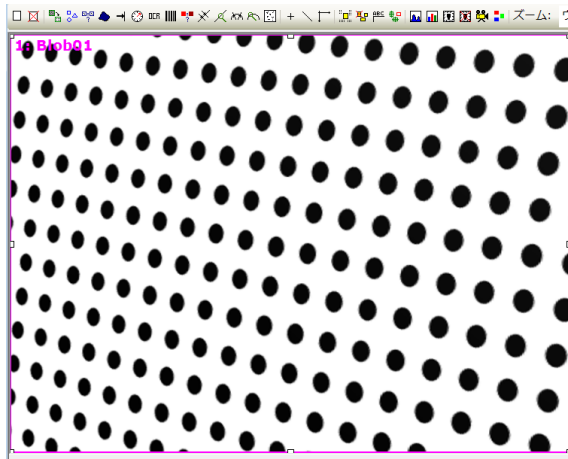
Image example of square grid pattern captured by camera with tilt angle

7.4.4 Vision Sequence for Local, Tool, and Arm Setting

To detect the local coordinate of the calibration plate by using a camera, a vision sequence to recognize the calibration plate is necessary.
Set properties for the sequence (ExposureTime, etc.) so that dots on the optional calibration plate can be recognized clearly. The vision object is not necessary.

To execute calibration of tool and arm settings using a camera, a vision sequence for detecting position of the calibration target is necessary. Create a sequence that has a vision object for detecting a position of one target in the same way as *7.4.1 Vision Sequence for Detecting One Target*.

7.5 Calibration GUI

This section describes the GUI used for calibration.

7.5.1 Create a New Calibration

To create a new calibration, click the <New calibration [icon] > button on the Vision Guide 7.0 toolbar, or select New Calibration from the Vision menu.

The following dialog will be displayed and the calibration wizard will start.
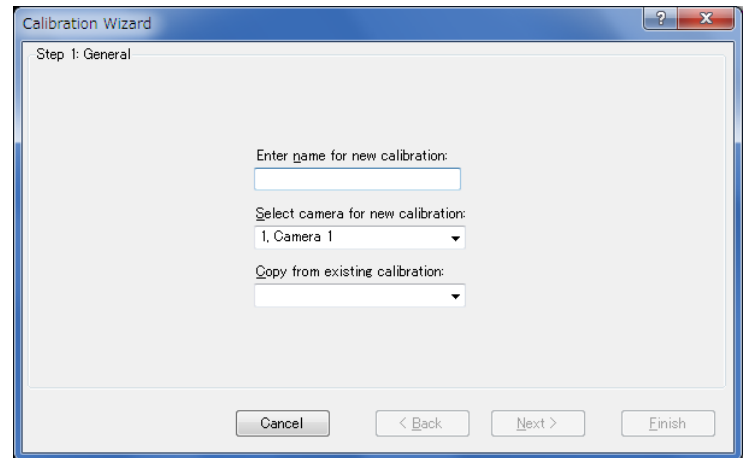
Enter the name for the new calibration. You can copy an existing calibration to specify the [Copy from existing calibration] after selecting the camera you will be calibrating.

After entering the calibration name, you can click the <Finish> button on any screen to exit the wizard. In this case, refer to *7.5.3 Calibration Properties and Results* and set required properties.
You can also click the <Back> button to revise the previous settings.

Click the <Next> button to move on to the next step.

If the name already exists for another calibration, an error message will be displayed.

Follow the instructions for each step to complete the wizard. Settings vary depending on the position or direction of camera. For details of the settings, refer to *7.6 Calibration Procedures*.

After completing the wizard, the calibration configured in the wizard will be added to the calibration tree. When the calibration is selected in the tree, the properties and results of calibration are displayed. The items configured in the wizard can be changed in the property panel anytime.

7.5.2 Delete a Calibration

To delete a calibration, click the <Delete calibration > button on the Vision Guide 7.0 toolbar, or select Delete Calibration from the Vision menu.

The following dialog will be displayed:
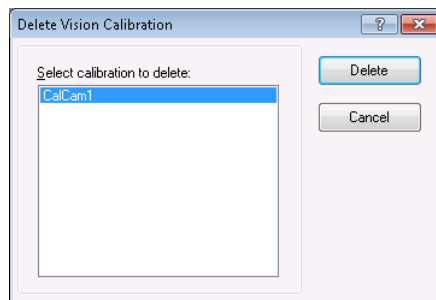
Select the calibration you want to delete and click the <Delete> button.

7.5.3 Calibration Properties and Results

Settings configured in the calibration wizard are reflected to the calibration properties. After creating a calibration, you can change the calibration properties in the calibration window on the Vision Guide window.

Property	Description
ApproachPoint	This property sets an approach point that is the start point of the robot for moving to each camera point in the calibration.
AutoCamPoints	This property enables the automatic camera point generation.
AutoReference	This property specifies whether or not to calculate the calibration reference point automatically when calibrating the mobile camera.
AutoRefFinalRotation	Sets the final angle of rotation for the tool when automatically calculating the reference point.
AutoRefInitRotation	Sets the initial angle of rotation for the arm and tool when automatically calculating the reference point.
AutoRefMode	This property specifies the auto calculate mode of the reference point as Rough, Fine, or Manual. Use Fine to move the arm largely and calculate at higher accuracy. Use Manual to manually enter the angle of rotation and tolerance values.
AutoRefTolerance	Sets the degree of tolerance for vision detection deviation when automatically calculating the reference point.
Camera	Specifies the camera for the currently selected calibration.

Property	Description
CameraOrientation	Specifies camera orientation. The choices are as follows:
	Mobile J2. The camera is looking downward and is mounted to Joint #2 on SCARA or Cartesian robots.
	Mobile J4. The camera is looking downward and is mounted to Joint #4 on SCARA or Cartesian robots.
	Mobile J5. The camera is looking downward and is mounted to Joint #5 on 6-Axis robots.
	Mobile J6. The camera is looking downward and is mounted to Joint #6 on 6-Axis robots.
	Fixed downward. The camera does not move and reports coordinates in the robot world.
	Fixed upward. The camera does not move and reports coordinates in the robot world.
	Standalone. The camera does not move and does not report coordinates in the robot world. It reports coordinates in the camera world.
DistCorrectCal	This property executes correction of lens distortion and camera tilt.
DistCorrectEnable	This property enables/disables correction of lens distortion and camera tilt.
DistCorrectTargetSeq	This property specifies a vision sequence to be used for correction of lens distortion and camera tilt.
	Any sequence in the current project can be used.
Lamp	This is an optional output channel that will be turned on automatically when calibration is started.
LampDelay	This is the amount of delay (in seconds) before a lamp is turned on. This allows for time required to turn on fluorescent lamps.
LJMMode	Sets a mode to control the posture flag for point data.
MaxMoveDist	Specifies a limit of move distance for the arm end.
MotionDelay	This is the amount of delay (in milliseconds) after each robot motion during the calibration cycle.
PointsTaught	This property defines whether calibration points have been taught.
ReferenceType	This property specifies the type of reference point used for calibration. There are two types of reference points for mobile camera calibrations: Taught and Upward camera.
RobotArm	This property specifies the arm number to use during calibration. Normally, this is set to zero.
RobotAccel	This is the acceleration that will be used during calibration.
RobotLimZ	This property specifies the LimZ value for the first motion command during a Scara mobile calibration cycle.
RobotLocal	This property specifies the local number to use for the current

Property	Description
	calibration. The local used must be previously defined.
RobotNumber	This property specifies the robot number to use for the current calibration.
RobotSpeed	This property specifies the speed that the robot will move during calibration. For higher speeds, you will need to set higher accelerations.
RobotTool	This property specifies the tool number to use during calibration. The tool number used must be previously defined.
RobotUOffset	This property sets the offset value for the U coordinate position of the detected part in robot coordinate system.
ShowConfirmation	This property sets whether to show the calibration results dialog for acceptance by the operator at runtime.
TargetSequence	This property specifies which vision sequence to use for calibration. This sequence can be any sequence in the current project.
TwoRefPoints	This property specifies whether or not to use two reference points in the calibration.
UpwardLamp	This is an optional output channel that will be turned on automatically when the upward camera sequence is run.
UpwardSequence	This property specifies which vision sequence to use for upward looking camera. This list is only active for mobile calibrations that use an upward camera for the reference point.

Result	Description
CalComplete	The completion status of a calibration.
CalImageSize	The size of the image when the calibration was executed.
DistCorrectCalComplete	The status whether or not correction of lens distortion and camera tilt is completed
FOVHeight	The height of the field of view in millimeters.
FOVWidth	The width of the field of view in millimeters.
XAvgError	The average calibration error along the camera X axis.
XMaxError	The maximum calibration error along the X axis.
XmmPerPixel	The X millimeters / pixel value.
XTilt	The calibration X tilt result.
YAvgError	The average calibration error along the camera Y axis.
YMaxError	The maximum calibration error along the Y axis.
YmmPerPixel	The Y millimeters / pixel value.
YTilt	The calibration Y tilt result.

7.5.4 Distortion Detection

To perform the distortion correction, detect distortion by manipulating the calibration properties. Distortion detection needs to be performed before executing the point teaching for other vision calibrations.

The steps to detect distortion are as below. When the distortion correction is configured in the calibration wizard, you can skip the step (1).
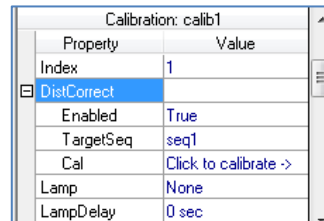
(1) Set [Enabled] under the DistCorrect property to "True" and select a sequence for detecting a square grid pattern for the TargetSeq property.

   How to create a sequence:
     Reference: *7.4.3 Vision Sequence for Distortion Correction*

(2) Select the Cal property under the DistCorrect property and execute detection for a square grid pattern.

You can check the distortion corrected image by specifying the calibration for which the distortion detection was performed to "Calibration" of the vision sequence and then executing the sequence.
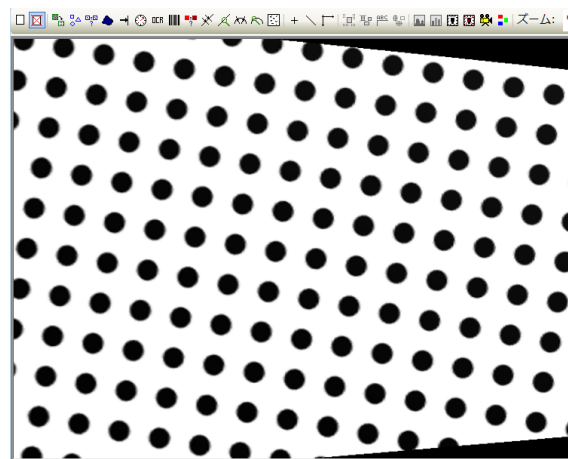
Example of image after correcting lens distortion and camera tilt

7.5.5 Point Teaching

The <Teach Points> button appears by selecting a calibration in the sequence or calibration tree on Vision Guide window.

Clicking the <Teach Points> button changes the Vision Guide window to the [Teach Calibration Points] mode where you can teach points for the currently selected calibration.

The [Teach Calibration Points] window is used to teach calibration points for the currently selected calibration scheme.

The points to teach vary depending on settings of the calibrations.
The point teaching panel appears as necessary according to settings of the calibration. Follow the descriptions on the screen to jog the robot and teach points.
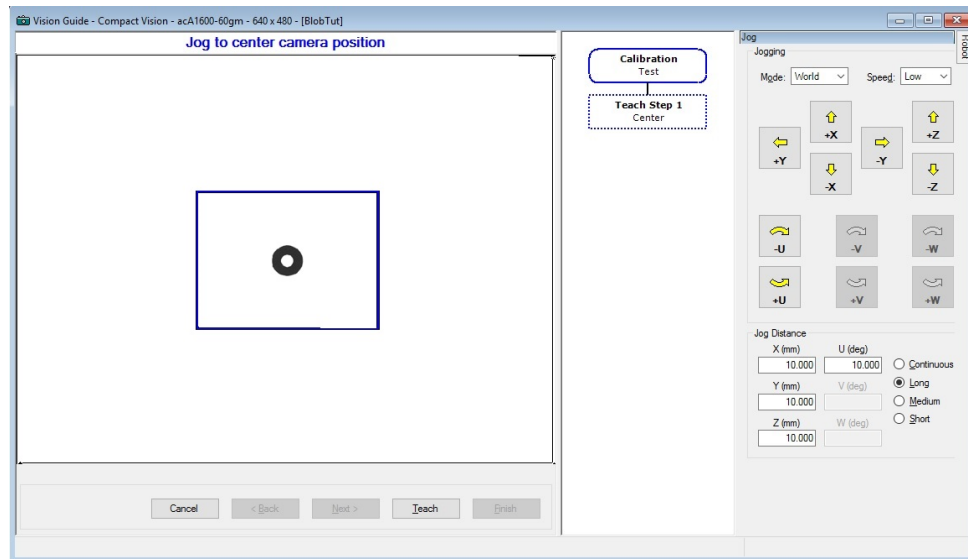
[Teach Calibration Points] dialog

An image is displayed on the left side of the window. Clicking the [Jog] tab at the right end opens the Jog window.

At the bottom of the display is a message box that displays instructions. After performing the required instruction, click on the <Next> button located to the right of the message box to continue to the next step. When all steps have been completed, a message box will appear indicating you are finished.

The instructions displayed vary for the types of calibration you are teaching points for.

Instruction	Description
Jog to fixed reference	This instruction is required for mobile camera calibration that is using a taught point for the reference (as opposed to using an upward looking camera to automatically search for the reference point). Jog the robot until the end effector is lined up with the calibration target.
Jog to fixed reference at 180 degrees	This is an optional step. Set the TwoRefPoints property to true before teaching points to enable this feature. If you are using a calibration tool that has been accurately defined, you can leave TwoRefPoints set to false. If you are not using a calibration tool, then set TwoRefPoints to true. You will be prompted jog the robot 180 degrees from the current position and line up the end effector with the calibration target.
Jog to top left camera position	Jog the robot until the calibration target is in the upper left corner of the vision display area.
Jog to top center camera position	Jog the robot until the calibration target is in the top center of the vision display area.
Jog to top right camera position	Jog the robot until the calibration target is in the upper right corner of the vision display area.
Jog to center right camera position	Jog the robot until the calibration target is in the center of the right side of the vision display area.
Jog to center camera position	Jog the robot until the calibration target is in the center of the vision display area.
Jog to center left camera position	Jog the robot until the calibration target is in the center of the left side of the vision display area.
Jog to bottom left camera position	Jog the robot until the calibration target is in the lower left corner of the vision display area.
Jog to bottom center camera position	Jog the robot until the calibration target is in the bottom center of the vision display area.
Jog to bottom right camera position	Jog the robot until the calibration target is in the lower right corner of the vision display area.

7.5.6 Teach Calibration Points

The <Calibrate> button appears by selecting a calibration in the sequence or calibration tree. Click the <Calibrate> button to start the calibration.
A confirmation message is displayed before the operation starts if the robot will be moved.

7.5.7 Calibration Complete Dialog

The dialog box as shown below appears after a calibration cycle has been completed. This dialog shows a summary of calibration values for the current calibration and the previous calibration. If this is the first calibration, then the previous calibration values will be blank.
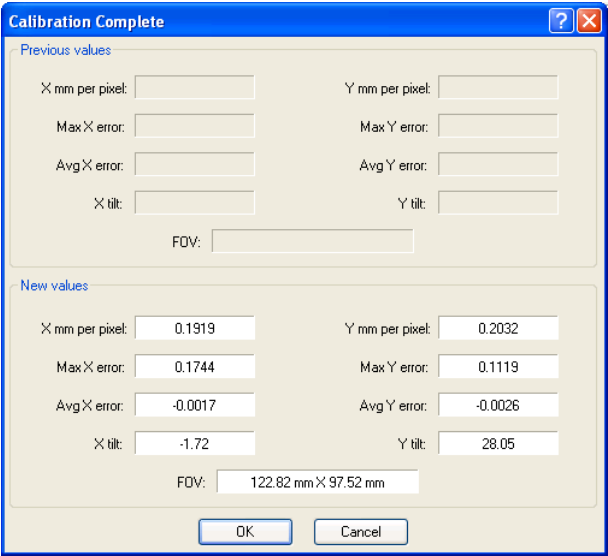
Calibration Status dialog that appears after Calibration is complete

The table below describes the values displayed for the [Calibration Complete] dialog. After examining the results, click OK to use the new calibration values, or Cancel to abort.

Tips: If there is an error of more than 1 mm or a tilt value is larger than 1, it can indicate that the calibration has not been completed properly.
Check that the calibration point is detected correctly during the calibration cycle, and check if the robot calibration points and the reference point are out of alignment at calibration.
To disable this dialog at runtime, set the ShowConfirmation property to false.

Note: When telecentric lenses are used, abnormal tilt values may be displayed.

Value	Description
X mm per pixel, Y mm per pixel	
	This is the resolution of the camera. The average width and height for one pixel.
Max X error, Max Y error	This is the maximum error that occurred during calibration verification. These values should be below the millimeter per pixel values. If they are not, the cause could be that the reference point(s) was not taught properly, or that the vision system is not locating the calibration target consistently, due to improper teaching or lighting.
Avg X error, Avg Y error	This is the average error that occurred during calibration verification.
X tilt, Y tilt	These values are relative indicators of camera tilt. The directions are as viewed from the camera in the image buffer coordinate system (plus x is right and plus y is down).
	For X tilt, a plus value indicates tilt to the right, minus is tilt to the left. For Y tilt, a plus value indicates tilt down, minus indicates tilt up.
	Vision Guide 7.0 compensates for camera tilt. However, it is recommended that you keep the tilt values below 1.0. This will help the vision system to find parts more accurately and repeatedly, depending on the application.
	Sometimes a high tilt value is caused by bad reference points. For example, if you are calibrating a fixed camera, which requires that you type in the coordinates of the calibration targets, inaccurate target location values could cause higher tilt values.
FOV	This is the width and height of the camera's <u>Field of View</u> in millimeters.

7.6 Calibration Procedures

This section contains step by step instructions for calibrating each of the camera installations.

7.6.1 Calibration Procedure: Mobile

Mobile camera calibration will allow you to search for objects from a camera mounted on a robot joint and get their coordinates in the robot world.

Preparation 1: Mount the camera to the robot

- Mount the camera to the robot.
  You can mount the camera at any rotational angle. The camera must be aligned vertically with the Z axis of the local coordinate system you will be using.

Preparation 2: Decide on reference point type

There are the following three choices for the reference point:

> Auto reference
> Using manually taught point
> Using a point that is found with an upward camera.

To use auto reference, set the AutoReference property to True. For best accuracy, set AutoRefMode to Fine.

Caution: By using Auto reference, move a robot and acquire reference point coordinate automatically when performing calibration. Be careful of interference between the robot and peripherals. Be careful especially when using in Fine mode since robot orientation moves largely (switching posture of left end effector and right end effector) during calibration. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during processing of Auto reference function.

For greatest accuracy, you should use an upward camera to find the reference target.
See the procedure for *7.6.3 Calibration procedure: Fixed upward* later in this chapter.

If you are not using auto reference or an upward camera to find the reference point, then use one of the following for training the reference point:

- Use a rod mounted to the U axis that extends through the center of the axis.

- Use the tool that is being used to pick up parts if it can be aligned with the calibration reference point.

If you define a tool (using TLSet command) for the calibration rod of pick up tool, then you will not have to teach zero and 180 degree reference point(s) during calibration.

Preparation 3: Create vision sequence to find dot grid pattern
(When performing distortion correction)

(1) Create a calibration plate.

(2) Create a vision sequence by refereeing the following section.
7.4.3 Vision Sequence for Distortion Correction

Preparation 4: Create vision sequence to find calibration reference target

(1) Create a vision sequence by referring to the following section.
7.4.1 Vision Sequence for Detecting One Target

When preparation is completed, run the calibration wizard to configure necessary settings.

Step 1: Run Calibration wizard

Click the <New Calibration > button on the Vision Guide 7.0 tool bar or select a calibration from the Vision menu to run the calibration wizard.

Enter the name for calibration in [Enter name for new calibration].

Select a camera for calibration in [Select camera for new calibration].

Settings can be copied by selecting a source calibration data in [Copy from existing calibration].

Click the <Next> button to proceed to Step 2.

Step 2: Setting for calibration type and camera direction

Click <Robot Camera>.

Select a robot on which a camera is mounted in [Select Robot].

Select a camera mount position from the following in [Mounting & Orientation].

For horizontal articulated (SCARA) robot:

Mobile J2 (Mobile camera: mounted on Joint #2)	Camera is mounted downward on the Arm #2 of horizontal articulated (SCARA) robot or Joint #2 of Cartesian coordinate robot.
Mobile J4 (Mobile camera: mounted on Joint #4)	Camera is mounted downward on the Arm #4 of horizontal articulated (SCARA) robot or Joint #4 of Cartesian coordinate robot.

For 6-axis robot:

Mobile J5 (Mobile camera: mounted on Joint #5)	Camera is mounted downward on the Arm #5 of vertical 6-axis robot.
Mobile J6 (Mobile camera: mounted on Joint #6)	Camera is mounted downward on the Arm #6 of vertical 6-axis robot.

After selecting the camera mount type, click the <Next> button.

Step 3: Specify a target sequence

Specify a target sequence.

Select a vision sequence created in *Preparation 4: Create vision sequence to find calibration reference target* from the list.
In this step, vision sequences for the camera specified in Step 1 are only displayed in the list.

After selecting the target sequence, click the <Next> button.

Step 4: Setting a local

Select a local you will be using for vision calibration.

In this dialog box, you can run the local wizard to define local coordinate system to the local number.
To run the wizard, select a local number other than "0", then click <Local Wizard…> button.
Details of local setting using a camera are described in *7.7 Local setting using a camera*,

After selecting the local coordinate, click the <Next> button.

Step 5: Setting for reference point type

Set a type of reference point you will be using for calibration.
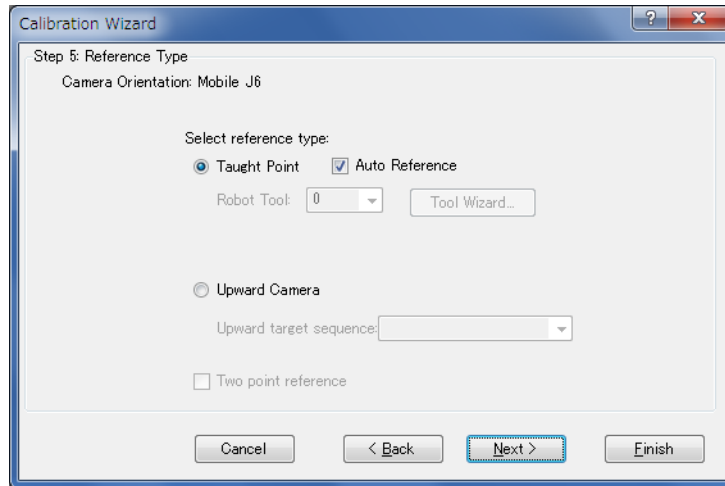
The reference point used for calibration is a point specified by the robot coordinate.

When <Taught Point> is selected:

If Auto Reference is checked, the reference will be found automatically. Otherwise, you need to jog the robot and teach the reference point manually.

It is necessary to specify a tool number and an arm number, which will be a reference point for an end effector attached at the end of the robot arm. (Setting for additional arm is available only for SCARA robots.)

Setting of tool and arm for a camera mounted on either of Joint #2, #4, or #6 can be omitted by selecting the [Auto Reference] checkbox. If setting is skipped, calculation for arm and tool is automatically performed and specifies the reference point automatically.

You can also run the tool wizard and define the tool by selecting a tool number other than "0" and clicking the <Tool Wizard…> button in this step.

In addition, you can run the arm wizard and define the arm for camera by selecting an arm number other than "0" and clicking the <Arm Wizard…> button.

How to define camera tool and arm.
Refer: *7.8 Detecting mobile camera mount position*

TIP: When performing the calibration with omitting the teaching of reference point by Auto Reference function and cannot acquire required calibration accuracy, accuracy may increase when performing the calibration without Auto Reference function.

When <Upward Camera> is selected:
You can detect a reference point accurately by using a fixed upward camera which has been calibrated.

If < Upward Camera > is selected for a type of reference point, also select a vision sequence which will be used for detection in [Upward target sequence].
The vision sequences where the selected camera is specified are only displayed in the list.

When [Two point reference] is selected:

The second reference point, that U is rotated 180 degrees, will be used for calibration.
This setting can be omitted when the tool is used.

After setting the reference point type, click the <Next> button.

Step6: Setting for AutoReference Parameters

Sets the parameter used when automatically finding the reference point. This step will appear if you selected the [AutoReference] check box in Step 5.
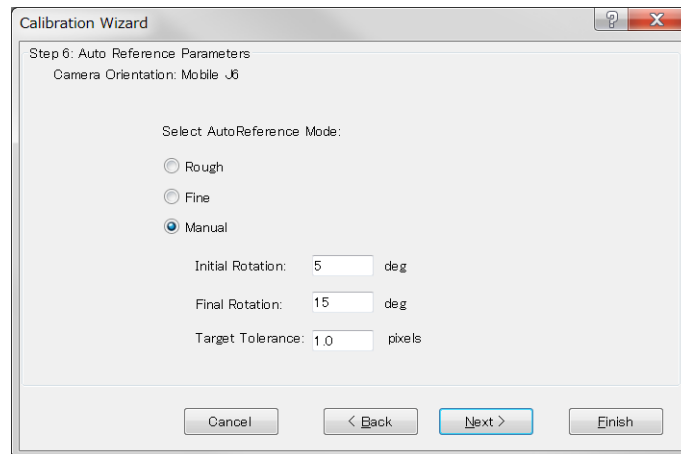
Sets the mode used when automatically finding the reference point. Note that robot movement will vary according to the position in which the camera is installed.

When <Rough> is selected:

Used for small robot movement with rough positioning.
With a MobileJ2 camera, this setting will move the arm in small increments.
With a MobileJ4 or J6 camera, this setting will rotate the tool in small increments.

When <Fine> is selected:

Used for large robot movements with precise positioning.
With a MobileJ2 camera, the arm will move according to changes in the right/ left orientation of the robot.
With a MobileJ4 or J6 camera, this setting will rotate the tool in large increments.

When <Manual> is selected:

This setting allows the user to enter in the angle of robot movement and the target tolerance manually. Select <Manual> to reduce robot movement if you feel the robot moves too great a distance even when <Rough> mode is selected. The arm will not move according to changes in the right/ left orientation.

After setting is completed, click the <Next> button.

Step7: Setting for camera points

Specify whether or not to generate camera points automatically when executing a calibration.
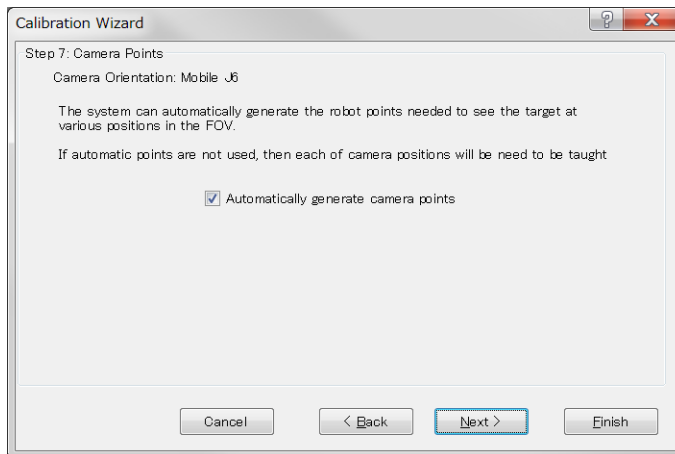
When [Automatically generate camera points] is selected:

Multiple camera points are generated automatically by operating the robot automatically while detecting a target object in the FOV of the camera. The position of the target object in the FOV is detected for each camera point. When using the auto generation, it is necessary to teach a point where the target object is placed near the center of the FOV.

When [Automatically generate camera points] is not selected:

It is necessary to jog the robot manually to teach a necessary number of camera points before executing a calibration.

After setting the auto camera point generation, click the <Next> button.

Step8: Setting for correction of lens distortion and camera installation distortion

Specify whether or not to correct lens distortion and installation distortion of the camera.
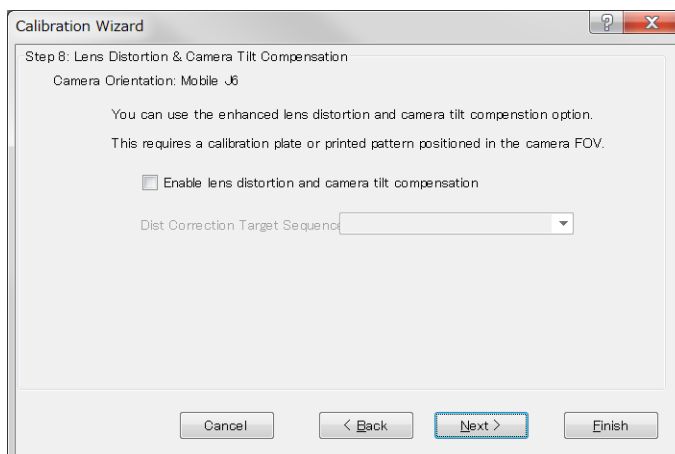
Selecting the checkbox enables correction. To correct distortion, it is necessary to create a target sequence for distortion correction in advance and specify it in this step.

After setting the distortion correction, click the <Next> button.

Step9: Setting for lighting control

Set the control for lighting used for calibration. If the lighting control is not necessary, setting is not necessary to be changed.
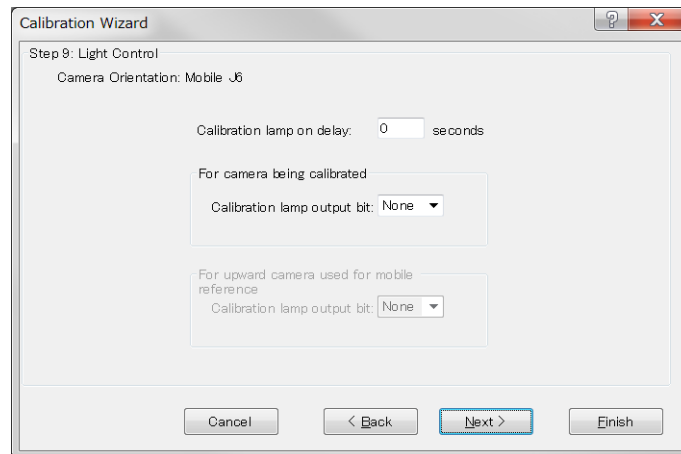
If you use lighting for the camera, specify a wait time before the lighting turns on in milliseconds. Also, specify an output bit for turning on the lighting.

If <Upward Camera> is selected for the reference point type in Step 5, the output bit for turning on the lighting of the upward camera can also be specified.

After setting the lighting control, click the <Next> button.

Step10: Setting for robot motion

Configure the settings for robot motion.
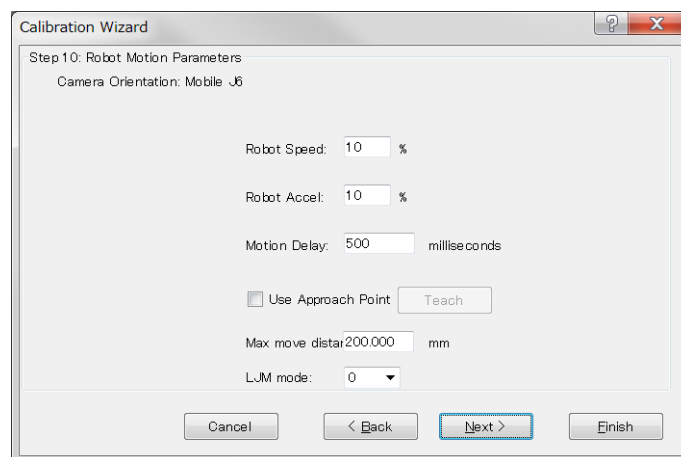
Set a speed and acceleration, and settling time after the robot motion (wait time before image capturing). To perform a fine calibration, set a slow speed and acceleration to ensure enough settling time.

An approach point can also be specified.
If the approach point is specified, the robot always moves to the calibration point from the specified approach point.
This allows the robot to approach to the calibration point in a fixed direction, and the robot position will be stabilized.

How to configure the approach point:

Select the [Use Approach Point] checkbox, then click the <Teach> button.

Teach the approach point in the displayed point teach dialog window.

After setting the approach point, click the <Next> button.

Step11: Confirmation of settings

The configured items are displayed. Check the settings.

```
Calibration Wizard                                          ?   X

  Step 11 : Summary


              Camera:           1, Camera 1

              Camera Orientation:   Mobile J6

              Reference Type:     Taught Points

              Target Sequence:    None




              Click Finish to save the new calibration or click Cancel



         Cancel        < Back        Next >        Finish
```

Click the <Finish> button to finish the wizard.

Detecting distortion correction (When distortion correction is enabled)

(1) Select the created calibration in the sequence or calibration tree.

(2) Place the dot grid pattern on the work plane.

(3) Select the Cal property below the DistCorrect property in the property list to execute detection of the dot grid pattern.

By selecting the created calibration scheme for the Calibration property (a sequence for positioning the calibration target), you can check the image which is corrected lens distortion and camera tilt. Even when this setting is omitted, distortion will be corrected automatically when executing the calibration.

Teach points

(1) Click the <Teach Points> button.
The [Teach Calibration Points] dialog appears.

(2) Follow the instructions displayed in the message box at the bottom of the dialog.
A necessary number of camera points and details of reference point teaching vary
depending on types of camera mount and reference point, and other settings.

(3) Teach camera points.
If the auto camera point generation is enabled, teach one camera position.
If the auto camera point generation is disabled, teach nine camera positions.
The camera positions indicate nine robot positions. Teach the first point so that the
target comes to the upper left corner, and then teach the second point so that the target
comes to the center of the image display area. Teach the remaining points accordingly
at any places in the field of view. For greatest results, teach the points so as to spread
them all over the field of view.

When using the vertical 6-axis robot, the V coordinate of the camera point is normally
zero, and the W coordinate is either zero or 180 depending on the direction of the local.
It is not necessary to change the V and W coordinates when teaching the calibration
points. This is for setting the camera positions to be relatively on the same lines as the
locals of the camera point.

(4) When "Teach Points" is selected for reference point type.
Teach the reference point.
Teach the reference point to the robot so that the tool used for picking up a workpiece
comes right above the workpiece. (This is same as teaching points as destinations of
robot motion.)

Calibration

Click the <Calibrate> button to start the calibration cycle.

Robot moves to each camera position and executes the calibration target vision sequence.

After moving to all nine positions, the system determines the calibration parameters and
repeats the cycle in order to collect the statistical data.

Clicking <Cancel> button stops the calibration.

7.6.2 Calibration Procedure: Fixed Downward

This calibration will allow you to search for objects from a fixed camera looking down into the robot work envelope and get their positions in the robot world. In this section, the procedure to plate the calibration on the work plane will be described. For details of the calibration procedure when using the target on the tool attached to the end effector as a reference point, refer to *"7.6.3   Calibration Procedure: Fixed Downward (when performing correction of lens distortion and camera tilt)"*.

Preparation 1: Mount the camera

(1) Mount the camera so that it will be looking down into the robot envelope.
     Make sure that the robot does not touch the camera.

Preparation 2: Prepare for distortion correction
                    (when performing distortion correction)

(1) Create a pattern for detecting distortion.
     Refer: *7.2 Correction of Lens Distortion and Camera Tilt*
     The optional calibration plate can also be used.

(2) Create a vision sequence.
     Refer: *7.4.3 Vision Sequence for Distortion Correction*

Preparation 3: Make a calibration plate

(1) Make a plate with nine holes or targets that span the field of view of the camera.

Preparation 4: Create vision sequence to find calibration reference targets

(1) Create a vision sequence.
     Refer: *7.4.1 Vision Sequence for Detecting One Target*

When preparation is completed, run the calibration wizard to configure necessary settings.

Step 1: Start the calibration wizard

(1) Click the <New Calibration [icon]> button on the Vision Guide 7.0 tool bar or select a calibration from the Vision menu to run the calibration wizard.

Enter the name for calibration in [Enter name for new calibration].

Select a camera for calibration in [Select camera for new calibration].

Settings can be copied by selecting a source calibration data in [Copy from existing calibration].

Click the <Next> button to proceed to Step 2.

Step 2: Setting for calibration type and camera direction

Click <Robot Camera>.

Select a robot on which a camera is mounted in [Select Robot].

Select a camera mount position from the following in [Mounting & Orientation].

After selecting the camera mount type, click the <Next> button.

Step 3: Specify a target sequence

Specify a target sequence.

Select a vision sequence created in *Preparation 4: Create vision sequence to find calibration reference target* from the list.
In this step, vision sequences for the camera specified in Step 1 are only displayed in the list.

After selecting the target sequence, click the <Next> button.

Step 4: Setting a local

Select a local you will be using for vision calibration.

In this dialog, you can run the local wizard to define local coordinate system to the local number.
To run the wizard, select a local number other than "0", and then click the <Local Wizard…> button.
Details of local setting using a camera are described in *7.7 Local setting using a camera*.
After selecting the local, click the <Next> button.

Step 5: Setting for reference point type

Set a type of reference point you will be using for calibration.
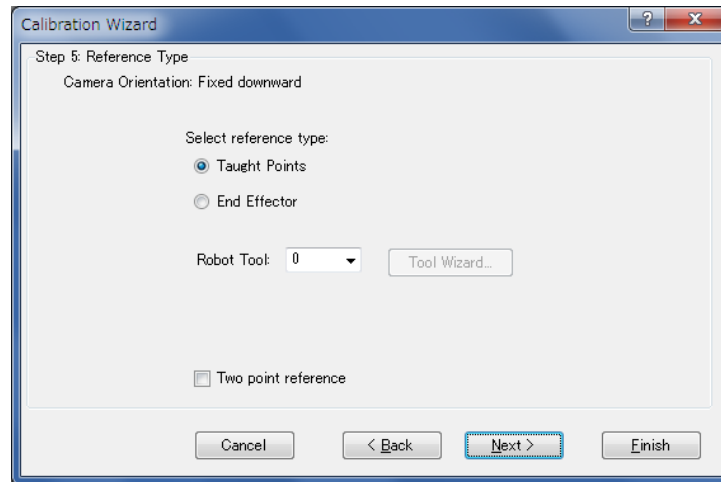
The reference point used for calibration is a point specified by the robot coordinate.

Select <Taught Points>.

You need to jog the robot and teach the reference point manually.

It is necessary to specify a tool number and an arm number, which will be a reference point for an end effector attached at the end of the robot arm. (Setting for additional arm is available only for SCARA robots.)

You can also run the tool wizard and define the tool by selecting a tool number other than "0" and clicking the <Tool Wizard…> button in this step.

In addition, you can run the arm wizard and define the arm for camera by selecting an arm number other than "0" and clicking the <Arm Wizard…> button.

Tool and arm settings for camera:
   Refer: *7.8 Detecting mobile camera mount position*

When < End Effector> is selected:
   Refer: 7.6.3 Calibration procedure: Fixed upward camera

When the [Two point reference] checkbox is selected:
   Two reference points will be used for the calibration.

After setting the reference point type, click the <Next> button.

Step 6: Setting for correction of lens distortion and camera installation distortion

Specify whether or not to correct lens distortion and installation distortion of the camera.
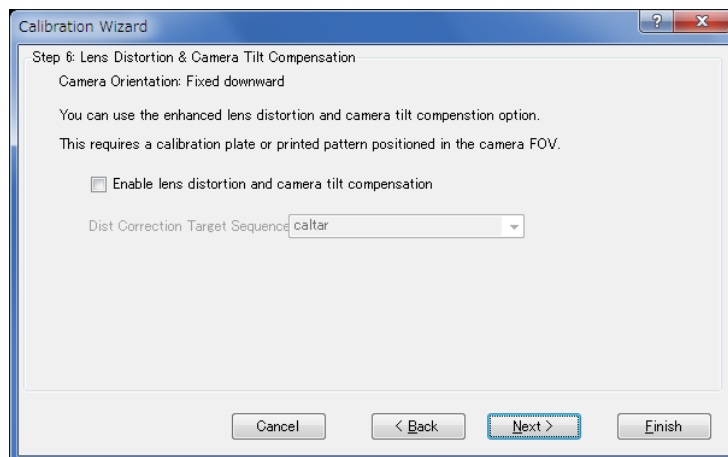
Selecting the checkbox enables correction. To correct distortion, it is necessary to create a target sequence for distortion correction in advance and specify it in this step.

After setting the distortion correction, click the <Next> button.

Step 7: Setting for lighting control

Set the control for lighting used for calibration. If the lighting control is not necessary, setting is not necessary to be changed.
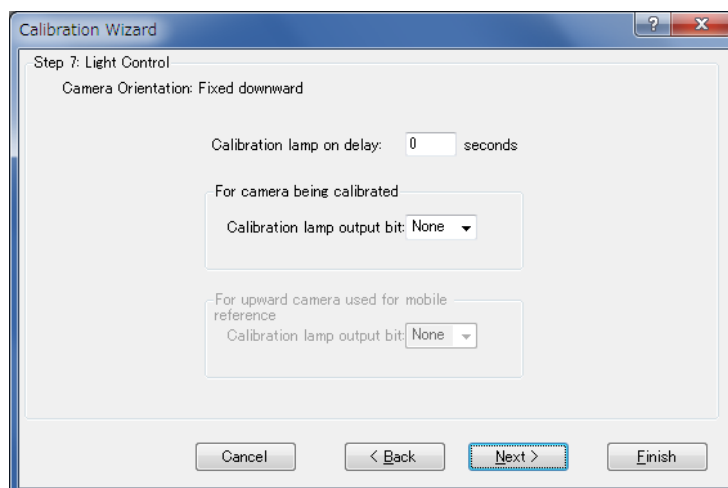
If you use lighting controls, specify a wait time before the lighting turns on in milliseconds. Also, specify an output bit for turning on the lighting.

After setting the lighting control, click the <Next> button.

Step 8: Setting for robot motion
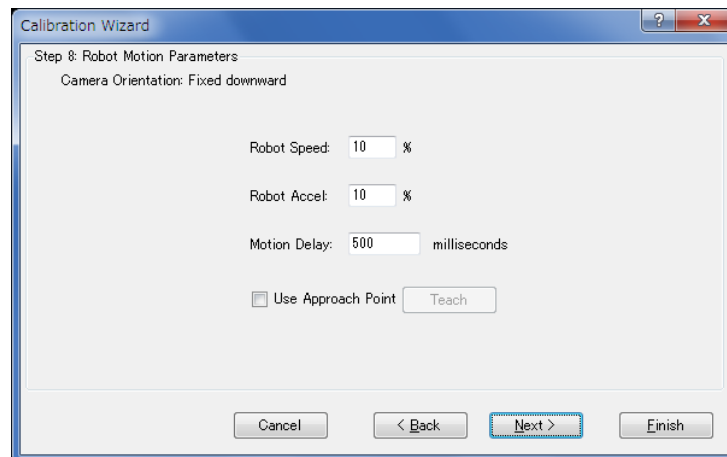
Configure the settings for robot motion.

Set a speed and acceleration, and settling time after the robot motion (Motion Delay). To perform a fine calibration, set a slow speed and acceleration to ensure enough settling time.

An approach point can also be specified.
If the approach point is specified, the robot always moves to the camera point from the specified approach point.
This allows the robot to approach to the camera point in a fixed direction, and the robot position will be stabilized.

How to configure the approach point:
Select the [Use Approach Point] checkbox, then click the <Teach Points> button.
Teach the approach point in the displayed point teach dialog window.

After setting the approach point, click the <Next> button.

Step 9: Confirmation of settings

The configured items are displayed. Check the settings.

Click the <Finish> button to finish the wizard.

Detecting distortion correction (When distortion correction is enabled)

(1) Select the created calibration in the sequence or calibration tree.

(2) Place the dot grid pattern on the work plane.

(3) Select the Cal property below the DistCorrect property in the property list to execute detection of the dot grid pattern.

By selecting the created calibration scheme for the Calibration property (a sequence for positioning the calibration target), you can check the image which is corrected lens distortion and camera tilt. Even when this setting is omitted, distortion will be corrected automatically when executing the calibration.

Teach points

(1) Click the <Teach Points> button.
The [Teach Calibration Points] dialog appears.

(2) Follow the instructions displayed in the message box at the bottom of the dialog to teach reference points.
You will be prompted to teach the point if TwoRefPoints is true, you must move the Joint #4 180 degrees to teach the point again. When the tool is used, this step can be skipped. To skip the step, click the <Next> button to move to the next step.

Calibration

Click the <Calibrate> button to start the calibration cycle.

The calibration software positions the nine targets and then determines the calibration parameters after positioning the targets again to collect the statistic data.

Clicking <Cancel> button stops the calibration.

7.6.3 Calibration Procedure: Fixed Upward

This calibration will allow you to search for objects from a fixed camera looking up and get their positions in the robot world.

The same calibration is possible for a fixed downward camera by setting the reference point type to "EndEffector".

Preparation 1: Mount the camera

(1) Mount the camera so that it will be looking up into the robot envelope.

Preparation 2: Create vision sequence to find calibration end effector target

(1) Create a sequence to locate the target on the end effector. Create one or more objects in the sequence to locate the target. The calibration software will use the last step in the sequence to get the location of the target. The last step results for X and Y should be the center of the target.

(2) During calibration, the end effector will be move to nine different points in the field of view of the camera and search for the target. Also, at each position, the calibration software will turn the U axis 180 degrees and search for the target again. This allows the software to determine the center of the U axis for each point.   For best results, use a round target.

When preparation is completed, run the calibration wizard to configure necessary settings.

Step 1: Run the Calibration wizard

Click the <New Calibration 🖼 > button on the Vision Guide 7.0 tool bar or select a calibration from the Vision menu to run the calibration wizard.

Enter the name for calibration in [Enter name for new calibration].
Select a camera for calibration in [Select camera for new calibration].
Settings can be copied by selecting a source calibration data in [Copy from existing calibration].

Click the <Next> button to proceed to Step 2.

Step 2: Setting for calibration type and camera direction

Click <Robot Camera>.

Select a robot on which a camera is mounted in [Select Camera].

Select a camera mount position from the following in [Mounting & Orientation].
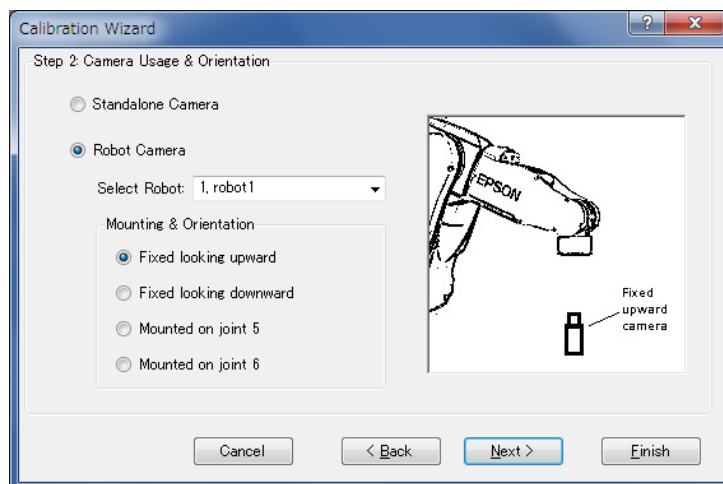
After selecting the camera mount type, click the <Next> button.

Step 3: Specify a target sequence

Specify a target sequence.

Select a vision sequence created in *Preparation 2: Create vision sequence to find calibration end effector target* from the list.
In this step, vision sequences for the camera specified in Step 1 are only displayed in the list.

After selecting the target sequence, click the <Next> button.

Step 4: Setting a local

Select a local you will be using for vision calibration.

In this dialog box, you can run the local wizard to define local coordinate system to the local number.

To run the wizard, select a local number other than "0", then click <Local Wizard…> button.

Details of local setting using a camera are described in *7.7 Local setting using a camera*,

After selecting the local, click the <Next> button.

Step 5: Setting for reference point type

Set a type of reference point you will be using for calibration.

The reference point type for the fixed upward camera is <End Effector>.

The reference point is a target on the tool mounted to the robot end effector.

The target needs to be seen from the camera.

(Although <Reference type> can be selected in this step, select <End Effector> for the fixed downward camera)

You can also run the tool wizard and define the tool by selecting a tool number other than "0" and clicking the <Tool Wizard…> button in this step.

In addition, you can run the arm wizard and define the arm for camera by selecting an arm number other than "0" and clicking the <Arm Wizard…> button.

(Setting for additional arm is available only for SCARA robots.)

Tool and arm settings for camera:
    Refer: *7.8 Detecting mobile camera mount position*

When the [Two point reference] checkbox is selected:
    Two reference points will be used for the calibration.

After setting the reference point type, click the <Next> button.

Step 6: Setting for camera point

Specify whether or not to generate camera points automatically when executing a calibration.
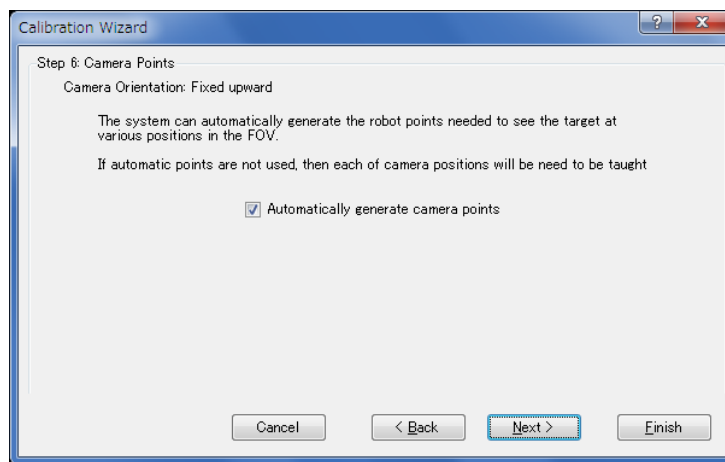
When the checkbox is selected:
    Multiple camera points are generated automatically by operating the robot automatically while detecting a target object in the FOV of the camera. The position of the target object in the FOV is detected for each camera point. It is not necessary to teach other camera points by jogging the robot.

When the checkbox is not selected:
    It is necessary to jog the robot manually to teach a necessary number of camera points before executing a calibration.

After setting the auto camera point generation, click the <Next> button.

Step 7: Setting for correction of lens distortion and camera installation distortion

Specify whether or not to correct lens distortion and installation distortion of the camera.
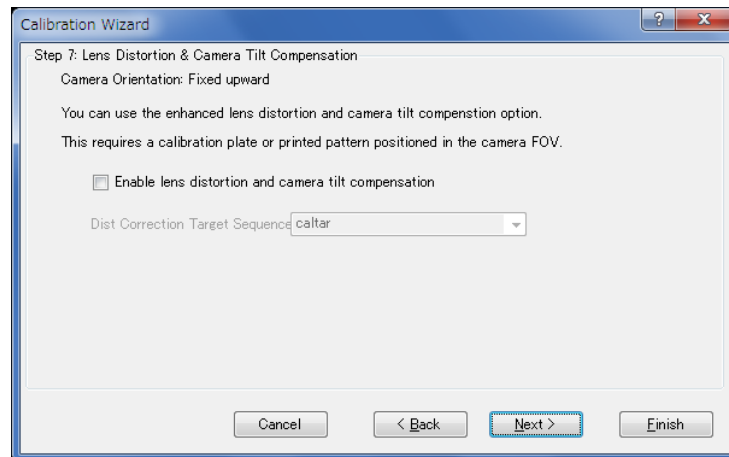
Selecting the checkbox enables correction. To correct distortion, it is necessary to create a target sequence for distortion correction in advance and specify it in this step.

After setting the distortion correction, click the <Next> button.

Step 8: Setting for lighting control

Set the control for lighting used for calibration. If the lighting control is not necessary, setting is not necessary to be changed.

If you use lighting controls, specify a wait time before the lighting turns on in seconds. Also, specify an output bit for turning on the lighting.

After setting the lighting control, click the <Next> button.

Step 9: Setting for robot motion

Configure the settings for robot motion.

Set a speed and acceleration, and settling time after the robot motion (Motion Delay). To perform a fine calibration, set a slow speed and acceleration to ensure enough settling time.

An approach point can also be specified.
If the approach point is specified, the robot always moves to the calibration point from the specified approach point. This allows the robot to approach to the calibration point in a fixed direction, and the robot position will be stabilized.

How to configure the approach point:
Select the [Use Approach Point] checkbox, then click the <Teach> button.
Teach the approach point in the displayed point teach dialog window.

After setting the approach point, click the <Next> button.

Step 10: Confirmation of settings

The configured items are displayed. Check the settings.
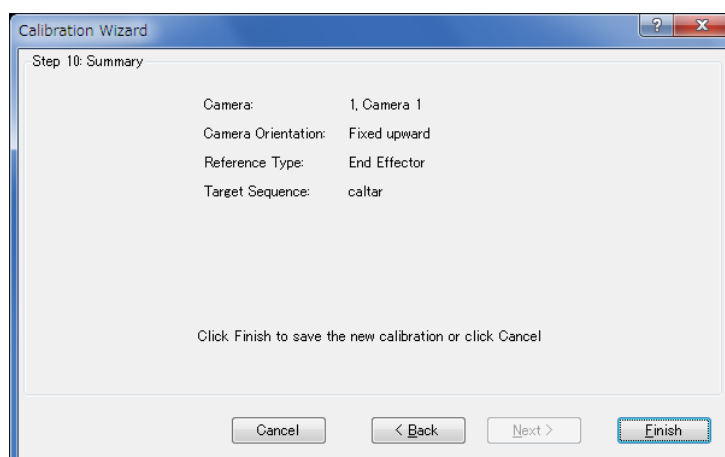
Click the <Finish> button to finish the wizard.

Teach points

(1) Click the <Teach Points> button.
The [Teach Calibration Points] dialog appears.

(2) Follow the instructions displayed in the message box at the bottom of the dialog to teach reference points.
You will be prompted to teach the point if TwoRefPoints is true, you must move the Joint #4 180 degrees to teach the point again. When the tool is used, this step can be skipped. To skip the step, click the <Next> button to move to the next step.

Calibration

Click the <Calibrate> button to start the calibration cycle.

The calibration software moves the robot to each camera position to search the target. When TwoRefPoints is set to "True", the robot rotates the Joint #4 180 degrees and searches the target again. It repeats the calibration to collect the statistic data.

Clicking <Cancel> button stops the calibration.

7.6.4 Calibration Procedure: Standalone

This calibration will allow you make physical measurements. Any camera calibrated as a "Standalone" camera cannot be used to calculate Robot world coordinates. The standalone calibration returns CameraX and CameraY values in millimeters.

Step 1: Mount the camera

(1) Mount the standalone camera at an angle of 40 to 90 degrees against the work plane.

Step 2: Make a dot grid pattern (When performing distortion correction)

(1) Create a dot grid pattern which has more than 100 points. The grid pattern must cover the entire field of view with minimum distortion. Accuracy of the grid pattern affects accuracy of image processing.

Step 3: Make a calibration plate

(1) Make a plate with nine holes or targets that span the field of view of the camera.

Step 4: Create vision sequence to find calibration reference targets
(When performing distortion correction)

(1) Create a sequence.
Refer: *7.4.3 Vision Sequence for Distortion Correction*

Step 5: Create vision sequence to find calibration reference targets

(1) Create a vision sequence.
Refer: *7.4.2 Vision Sequences for Detecting Nine Targets*

Step 6: Create a calibration scheme

(1) Click the <New Calibration ▣> button on the Vision Guide 7.0 tool bar or select a calibration from the Vision menu.

(2) The calibration wizard is run.
Select the name of calibration and a camera. Select a source calibration for copying settings as necessary.

(3) Select a standalone camera.

(4) Configure the vision sequence for detecting the calibration reference targets.

(5) Specify whether or not to enable distortion correction.
If distortion correction is enabled, select a vision sequence for detecting the dot grid pattern.

(6) Configure the settings by following the steps to complete the wizard.

Step 7: Setting for correction of lens distortion and camera installation distortion

(1) Select the created calibration in the sequence or calibration tree.

(2) Place the dot grid pattern on the work plane.

(3) Select the Cal property below the DistCorrect property in the property list to execute detection of the dot grid pattern.

(4) By selecting the calibration scheme created in Step 6 for the Calibration property (a sequence for positioning the calibration target), you can check the image which is corrected lens distortion and camera tilt. Even when this setting is omitted, distortion will be corrected automatically when executing the calibration.

Step 8: Calibration

(1) Remove the dot grid pattern and place the calibration plate creates in Step 3.

(2) Click the <Teach Points> button to set the coordinates for the nine targets in the calibration plate.

(3) Click the <Calibrate> button to start the calibration cycle. The calibration software positions the nine targets and then determines the calibration parameters after positioning the targets again to collect the statistic data.

7.7 Local Detection Using a Camera

Local detection can be performed by using a camera with a 6-axis robot.
Select the [Locals] tab in the Robot Manager and start the Local Wizard. These setting wizards can be displayed from the calibration wizard.

7.7.1 Defining a Local on Work Plane

- This section describes how to define the local coordinate parallel to the work plane by detecting a calibration plate on the work place by the mobile camera. This function is enabled when the mobile camera which is mounted on Arm #6 (J6) on the 6-axis robot is used.

Note: To configure the local on the work plane by using a camera, use an optional calibration plate. Local setting using a camera may not be available depending on hardware configuration of the camera.
Refer: *7. Vision Calibration*

Defining of the local coordinate using a camera can be executed in the local wizard.

Run the local wizard by either of the following ways.

(1)   Select the [Locals] tab in the Robot Manager.
The <Local Wizard…> button appears. Click the button.

(2)   The <Local Wizard…> button appears when a robot local number other than "0" is selected in the step for selecting a robot local number in the calibration wizard.
Click the button.

The following describes the steps after starting the local wizard. (Contents of the wizard vary depending on choice of the items)

Step 1: Select type

When the local setting wizard is run, the following dialog box appears.
Click the <Detect the local of a calibration plate using a mobile camera> button.
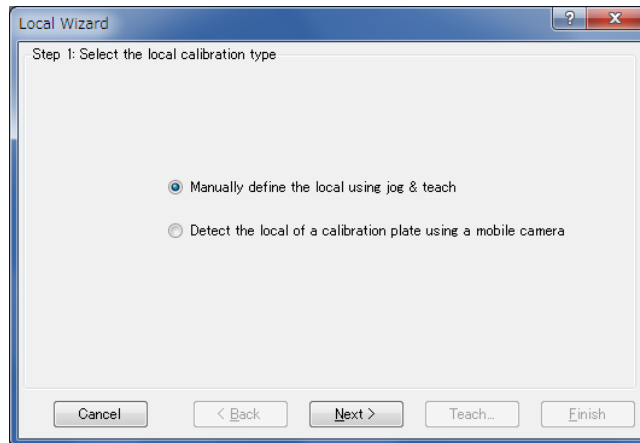
Click the <Next> button.

Step 2: Select local number

Select a local number to configure.

Click the <Next> button.

Step 3: Select vision component
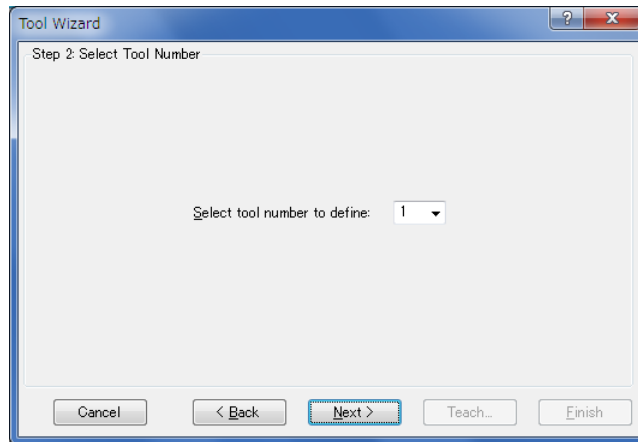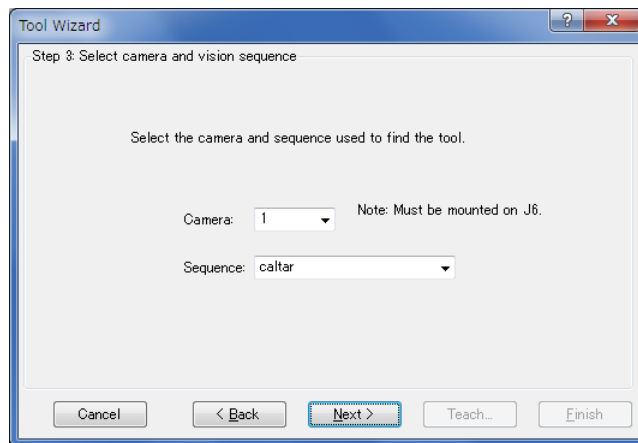

Set the following items.

Item	Description
Camera	Select a camera used in the calibration. Also, the camera needs to be mounted on Arm #6 (J6).
Sequence	Select a vision sequence for detecting the calibration plate.
Camera Tool	Specify the tool number for the calibrated mobile camera. When "Auto" is selected, the camera tool will be detected automatically.
Calibration Plate	Select type of the calibration plate.
Teach Local Reference Point	When the local reference points are taught, the detected local plane will be set to pass the taught points. When the reference points are not taught, the local plane will be set to pass Tool 0. If this function is enabled, it is necessary to jog the robot so that the end effector comes to the local plane and teach points.

Click the <Next> button.

Step 4: Teach camera point

Click the <Teach…> button. The jog dialog appears.
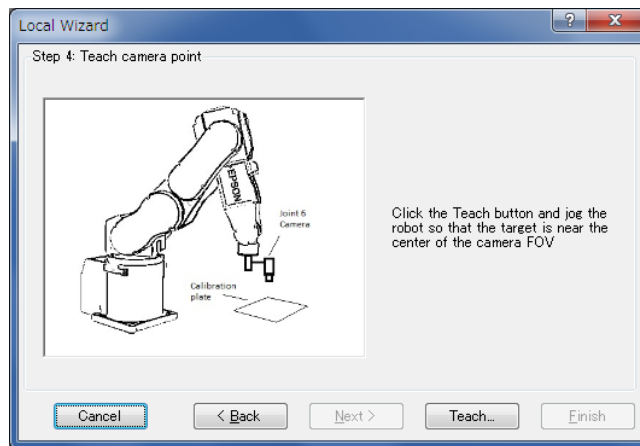
Jog the robot and move the center of the calibration plate to the center of the field of view.
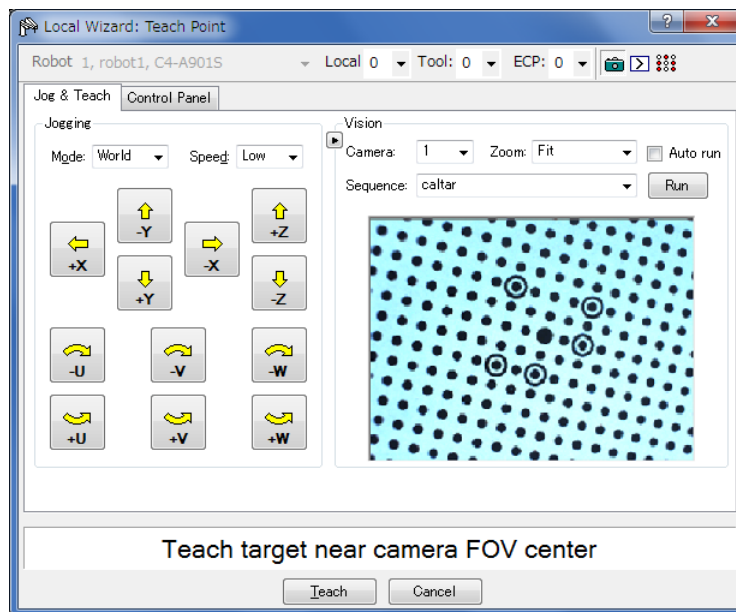
Click the <Teach> button when the camera point is determined.

Go to the next step.

Step 5: Teach local reference point

This step appears only when the local reference point needs to be taught.

Clicking the <Teach…> button displays the jog dialog.
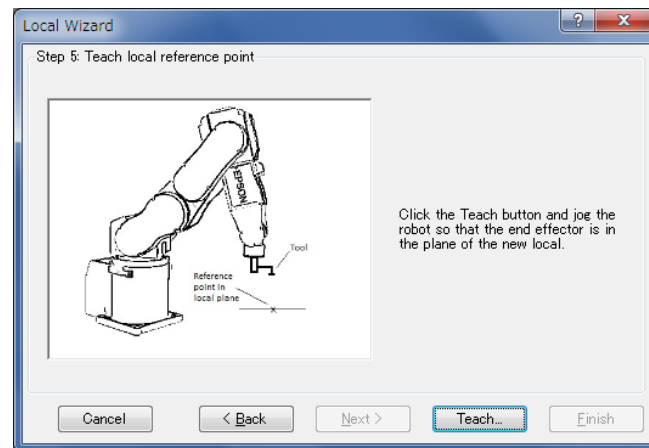
Jog the robot so that the end effector comes to the local plane.

Clicking the <Teach> button teaches the point and the wizard returns to this step.

Step 6: Define Vision local

Change the following settings as necessary.
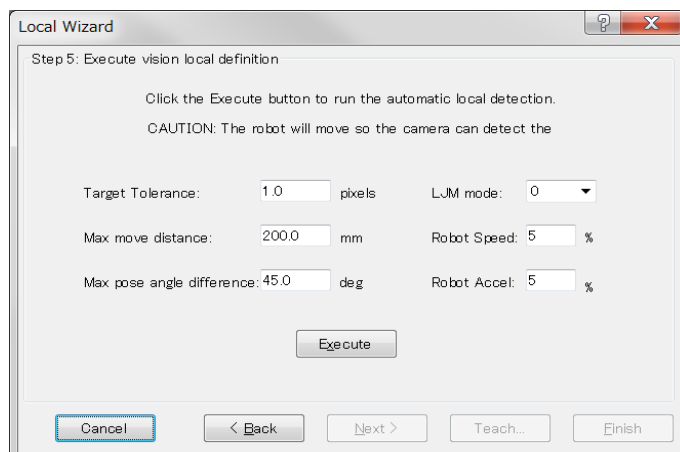
Change the following settings as necessary.

Item	Description
Target Tolerance	Detection is performed so that the error on the image can be settled within the allowance specified in this box.
Max move distance	Specify a limit of move distance for the arm end. If "0" is specified, distance will not be restricted.
Max pose angle difference	Maximum displacement angle of Tool orientation (UVW). (Unit: degree) If "0" is specified, angle will not be restricted.
LJM mode	Specify the value used for LJM function of SPEL+. The LJM mode controls the posture flag for point data to prevent unintentional rotation of the wrist. If "0" is specified, do not use LJM
Robot Speed	Sets the robot speed. Set a low speed value to configure precise local settings.
Robot Acceleration	Sets the robot acceleration. Set a low acceleration value to configure precise local settings.

Click the <Finish> button.
After the detection is completed, results will be displayed. Check the results.

Clicking the <Finish> button sets the results to the specified local number.

Note: The robot moves automatically according to the results of calibration plate recognition and detection of the target.
Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during local detection.

7.8 Detecting Mobile Camera Mount Position

The mount position of a mobile camera can be detected.

A camera installed on Arm #2 (J2) of SCARA robot is configured as a parameter for additional arm settings.

A camera installed on Joint #4 (J4) of SCARA robot or Joint #6 (J6) of vertical 6-axis robot can be configured as a tool.

7.8.1 Tool Setting of Camera Installation Position

The following describes the steps to configure tool settings for a camera installed on Joint #4 (J4) of SCARA robot or Joint #6 (J6) of vertical 6-axis robot.

Create a vision sequence necessary for detecting a target object in advance.

Run the tool wizard by either of the following ways.

(1)   Select the [Tools] tab in the Robot Manager.
The <Tool Wizard…> button appears.
Click the button.

(2)   The <Tool Wizard…> button appears when a tool number other than "0" is selected in the step for selecting a tool number in the calibration wizard.
Click the button.

Step 1: Select type

When the tool wizard is run, the following dialog box appears.
Click the <Define a tool for a mobile camera> button.

Click the <Next> button.

Step 2: Select tool number

Select a tool number to configure.

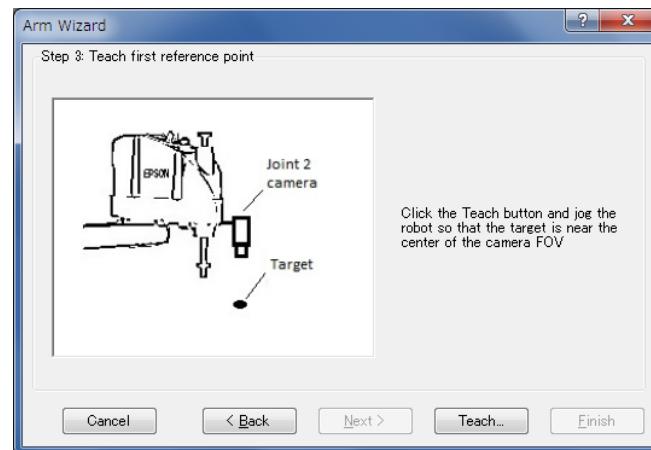Click the <Next> button.

Step 3: Configure Vision

Select a sequence to detect a camera and target object.

Click the <Next> button.

Step 4: Teach camera point

Click the <Teach…> button. The point teaching dialog box appears.

Jog the robot so that the target object can be detected near the center of the field of view.

Click the <Teach> button.
Go to the next step.

Step 5: Execute

Change the following settings as necessary.

Item	Description
Initial Rotation	Angle to rotate the tool initially in a rough positioning. (deg.)
Final Rotation	Final rotation angle of the tool (more than 90 deg. is recommended). The greater the angle of final rotation is, the higher accuracy is provided to tool setting.
Target Tolerance	Detection is performed so that the error on the image can be settled within the allowance specified in this box.
Max move distance	Specify a limit of move distance for the arm end. If "0" is specified, distance will not be restricted.
LJM mode	Specify the value used for LJM function of SPEL+. The LJM mode controls the posture flag for point data to prevent unintentional rotation of the wrist. If "0" is specified, do not use LJM
Robot Speed	Sets the robot speed. Set a low speed value to configure precise local settings.
Robot Acceleration	Sets the robot acceleration. Set a low acceleration value to configure precise local settings.

Click the <Execute> button.

After the detection is completed, results will be displayed. Check the results.

Clicking the <Finish> button sets the results to the specified tool number.

Note: The robot moves automatically according to detection results of the target object.
Be careful of interference between the robot and peripherals.

7.8.2 Arm Setting of Camera Installation Position

The following describes the steps to configure installation position of a camera installed on Joint #2 (J2) of SCARA robots additional arm parameters.

Create a vision sequence necessary for detecting a target object in advance.

Run the arm wizard by either of the following ways.

(1)   Select the [Arms] tab in the Robot Manager.
       The <Arm Wizard…> button appears.
       Click the button.

(2)   The <Arm Wizard…> button appears when an arm number other than "0" is selected
       in the step for selecting an arm number in the calibration wizard.
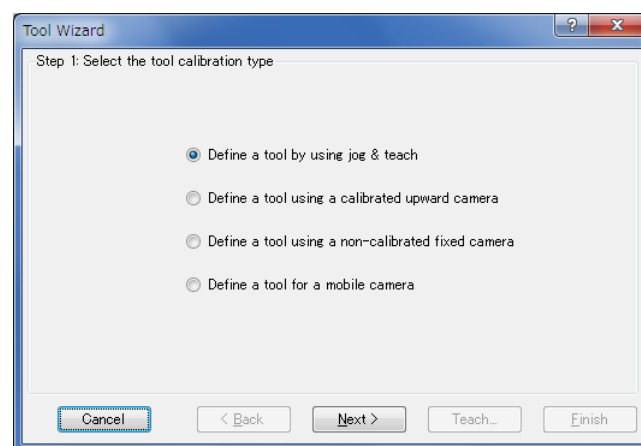   Click the button.

Step 1: Select arm number

Select an additional arm number to configure.
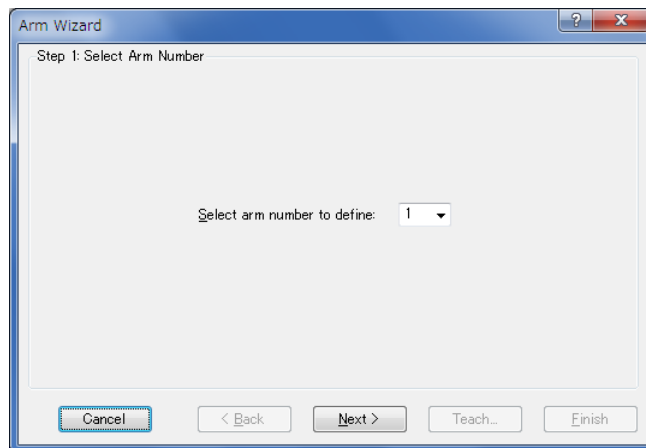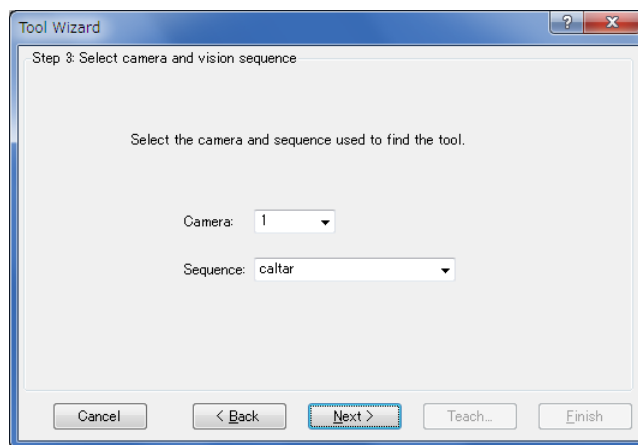

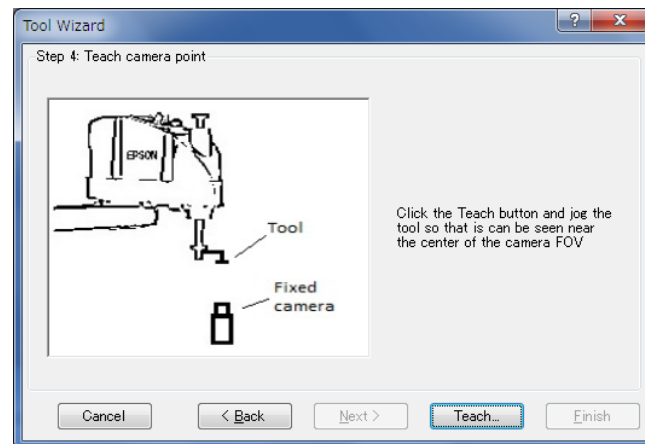Click the <Next> button.

Step 2: Configure Vision

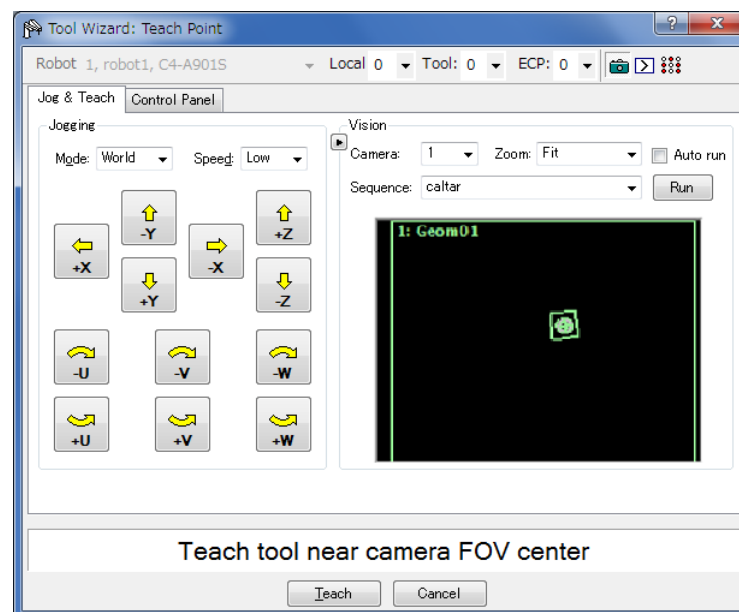Select a sequence to detect a camera and target object.
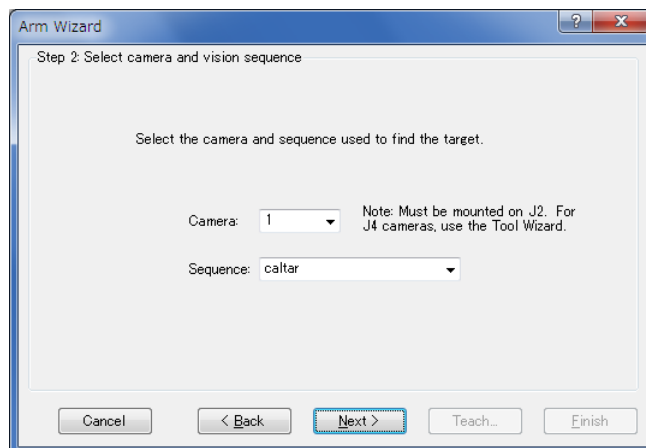
Click the <Next> button.

Step 3: Teach camera point

Click the <Teach…> button. The point teaching dialog box appears.

Jog the robot so that the target object can be detected near the center of camera's FOV.
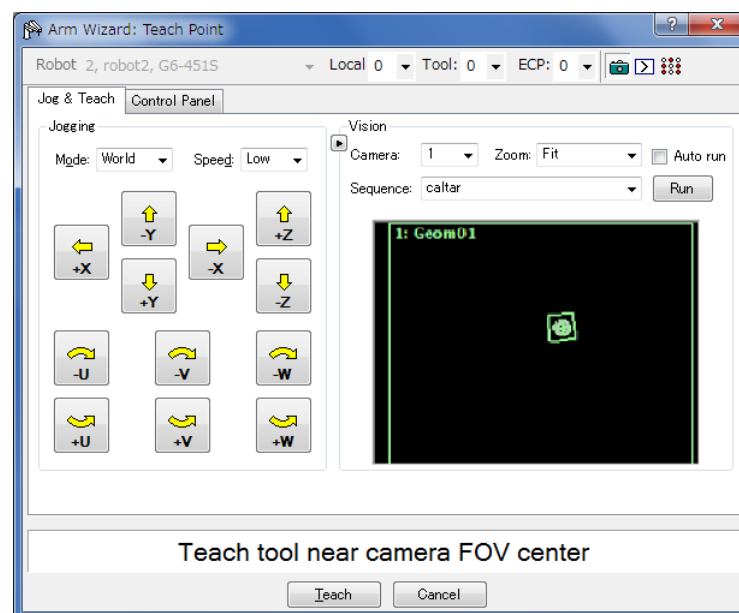
Click the <Teach> button.
Go to the next step.

Step 4: Execute

Set the following items as necessary.

Item	Description
Rotation	Specify a rotation angle (deg) used for performing a rough arm setting.
Target Tolerance	Specify a pixel distance to consider that the vision detection result matches the target position.

Mode	Rough: A mode to run a rough arm setting.
	           Robot motion is small.
	           Robot aims at accuracy of approximately 1 mm..
	 Fine: A mode to run a fine arm setting.
	           Robot moves largely with arm orientation change.
	           Robot aims at high accuracy rather than rough accuracy.

Robot Speed	Sets the robot speed. Set a low speed value to configure precise local settings.
Robot Acceleration	Sets the robot acceleration. Set a low acceleration value to configure precise local settings.

Click the <Execute> button.

After the detection is completed, results will be displayed. Check the results.

Clicking the <Finish> button sets the results to the specified additional arm number.

Note: The robot moves automatically according to detection results of the target object.
Be careful of interference between the robot and peripherals.

7.9 Tool Setting Using Camera

Tool coordinates of the tool installed at the end of the robot can be detected by using a fixed camera.

The tool setting wizard has almost the same steps as the wizard in *7.8.1 Tool setting of camera installation position*.

Create a vision sequence necessary for detecting a target in advance.

Run the tool wizard by either of the following ways.

(1)　Select the [Tools] tab in the Robot Manager.
　　　The <Tool Wizard…> button appears.
　　　Click the button.

(2)　The <Tool Wizard…> button appears when a tool number other than "0" is selected in the step for selecting a tool number in the calibration wizard.
　　　Click the button.

Step 1: Select type

When the tool wizard is run, the following dialog box appears.

Select either of the following buttons.

<Define a tool using a calibrated upward camera>

This function is enabled when the calibrated upward camera is specified to the calibration property of the sequence.

<Define a tool using a non-calibrated fixed camera>

Click the <Next> button.

Step 2: Select tool number

Select a tool number to configure.

Click the <Next> button.

Step 3: Configure Vision

Select a sequence to detect a camera and target object.

Click the <Next> button.

Step 4: Teach camera point

Click the <Teach…> button. The point teaching dialog box appears.

Jog the robot so that the target object can be detected near the center of camera's FOV.

Click the <Teach> button.
Go to the next step.

Step 5: Execute

When using a fixed camera which is not calibrated, configure the following items. When calibrated fixed camera is used, setting is not necessary.

Item	Description
Initial Rotation	Angle to rotate the tool initially in a rough positioning (deg).
Final Rotation	Final rotation angle of the tool (more than 90 deg. is recommended). The greater the angle of final rotation is, the higher accuracy is provided to tool setting.
Target Tolerance	Detection is performed so that the error on the image can be settled within the allowance specified in this box.
Max move distance	Specify a limit of move distance for the arm end. If "0" is specified, distance will not be restricted.
LJM mode	Specify the value used for LJM function of SPEL+. The LJM mode controls the posture flag for point data to prevent unintentional rotation of the wrist. If "0" is specified, do not use LJM
Robot Speed	Sets the robot speed. Set a low speed value to configure precise local settings.
Robot Acceleration	Sets the robot acceleration. Set a low acceleration value to configure precise local settings.

Click the <Execute> button.

After the detection is completed, results will be displayed. Check the results.

Clicking the <Finish> button sets the results to the specified tool number.

Note: The robot moves automatically according to detection results of the target object. Be careful of interference between the robot and peripherals. Also, use with avoiding singularity nearby posture that each axis extends to prevent an error during tool setting.

8. Histogram Tool

Histograms are a very powerful feature when working with machine vision systems. They allow you to view data in a chart format that makes it much easier to see things like why a part isn't found properly. They also give hints for things like how to find a medium gray blob when a dark and light gray blob can also be found within the search window.

The [Vision Guide Histogram] dialog displays gray levels along the horizontal axis and pixel counts along the vertical axis. This lets us easily see how many pixels for each gray level are shown in the current image. The left side of the [Histogram] graph (or lower gray level numbers) shows pixels that are approaching darker gray levels (black). Darker pixels are represented with a low value (approaching 0). The right side of the [Histogram] graph (or higher gray level numbers) shows pixels that are approaching lighter gray levels (white). Lighter pixels are represented with a high value (approaching 255).

8.1 Using Histograms

To bring up the [Histogram] dialog, first select the vision object that you would like to see a Histogram for. (This can be done from the drop down list at the top of the [Object] tab.) Histograms are supported for the Correlation and Blob objects. Next run the vision object or vision sequence to make sure that the vision object you selected has valid results. Then click on the <Histogram 🔳 > toolbar button or select [Histogram] from the [Vision] menu. The [Histogram] dialog will appear as shown in *8.3 Histograms with Correlation Objects* if the vision object is a Correlation object and as shown in *8.4 Histograms with Blob Objects* if the vision object is a Blob object.
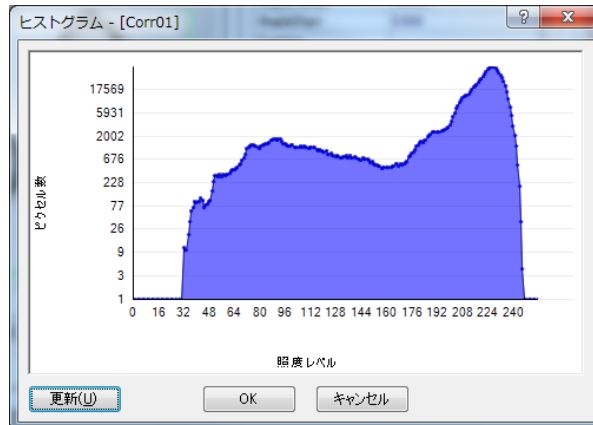
8.2 Example of Histograms

The figure below shows 10 dark rings on a light gray background. The best vision tool for finding these parts will probably be a Blob object since each of these parts will rotate up to 37.0 degrees. However, we are going to use this same image for showing the [Histogram] dialog when used with a Correlation object and also when used with a Blob object.
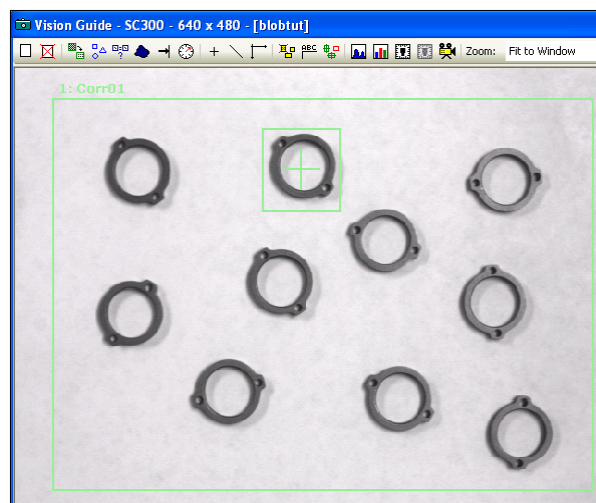
Example Image with 10 Rings using Corr01 object

8.3 Histograms with Correlation Objects

There are some important differences in the [Histogram] dialog when used with a Correlation object vs. using it with a Blob object. Figure below shows the [Vision Guide Histogram] dialog that was generated for the Corr01 object. Notice that it shows the spread of pixels over the various gray levels very nicely so that we can see how the image is dispersed. You can easily see that there are a lot more light pixels than dark pixels which is true since most of the image in *8.2 Example of Histograms* is the gray background and not the dark rings.

Example Histogram for Correlation object "Corr01"

8.4 Histograms with Blob Objects

Figure below shows the Histogram for the same image when used with a Blob object. Notice that there are 2 vertical bars with values attached at the base called ThresholdLow and ThresholdHigh. These are Blob object properties that are used to specify which gray levels to include as part of the found blob and which to include as part of the background.
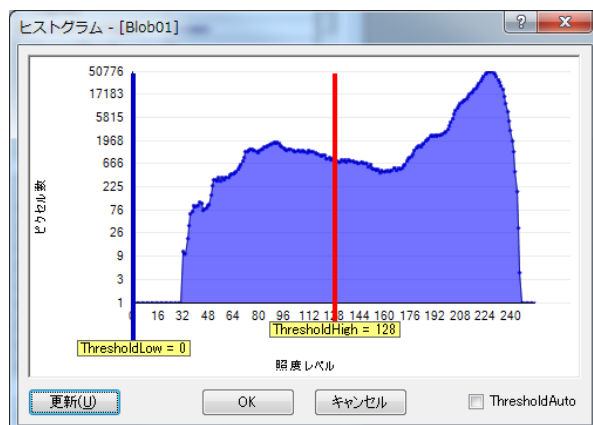
Example Histogram for Blob object "Blob01" (Dark on Light)

The area between the ThresholdLow and ThresholdHigh slider bars is the grouping of pixels that represent the levels of gray we want to define as dark or white pixels, which is set with the ThresholdColor property. The Polarity property defines whether we are looking for dark objects or light objects.

Adjusting the ThresholdLow and ThresholdHigh Properties

Look at the above figure again. Notice that the ThresholdLow property is set to 0 and the ThresholdHigh property is set to 128. These are the default values for these Blob object properties. When we first run the Blob object on the Rings image (with NumToFind set to 10) we get a result as shown in the figure below. Notice that the Extrema for many of the blobs found does not go around the outer parts of the rings and in some cases only a portion of the part is detected as a blob or 1 part is found as 2 blobs (see the arrows in the figure below that indicate the problem areas). This is because the ThresholdLow and ThresholdHigh properties must be adjusted based on the Histogram results.
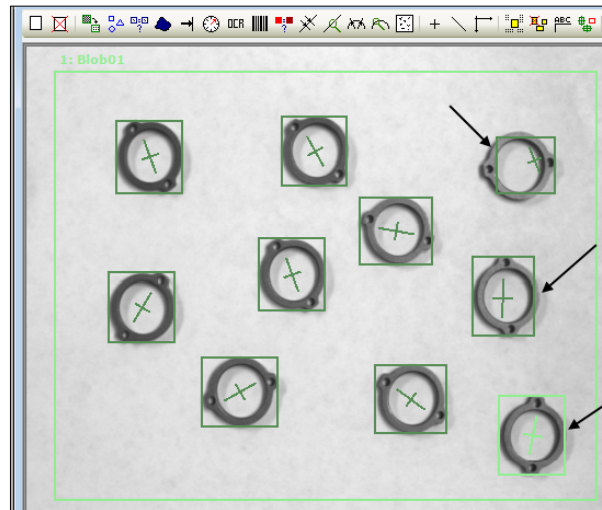

Image of 10 Rings with poor threshold settings

If we use the [Histogram] dialog to examine the histogram for the rings image we see a large distribution of gray starting at a level around 32. Then another large peak starts at around 170. Since the largest distribution of gray in the rings image is the light background it is easy to see that the distribution of pixels above 170 is or the background. Also, since the other peak in the histogram runs from about 32 till up to 170 this shows the distribution of the darker parts of our image which are the rings (the part we are interested in). The ThresholdLow and ThresholdHigh values can be adjusted so that our found boxes for each blob fall right around the outer bounds of the rings. This is done by clicking on the ThresholdHigh and ThresholdLow bars in the [Histogram] dialog and dragging them to the positions as shown in the figure below.
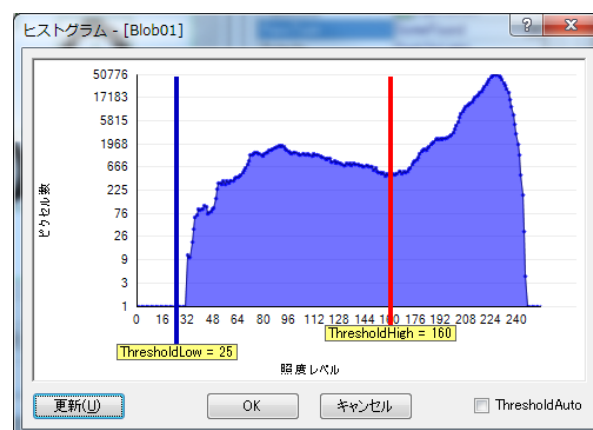
Histogram of Rings Image with better Threshold Settings

If we look at the rings image after running the Blob object with our new ThresholdLow and ThresholdHigh settings, we can see that the returned results are much more in keeping with what we wanted. Each ring is now found with the correct extrema for each blob.
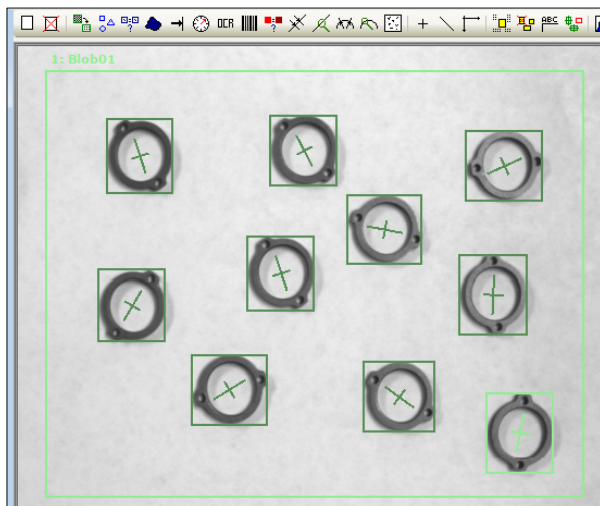

Image of 10 Rings with Improved Threshold Settings

Also, the ThresholdAuto checkbox is located at the lower right corner of the dialog window. By checking the checkbox, appropriate ThresholdLow and ThreshouldHigh values considering from the search window can be set.

In the case of the above figure, the ThreshouldLow property value is set to 163, and the ThreshouldHigh property value is set to 255. The status of the ThreshouldAuto checkbox is linked to the ThreshouldAuto property value. If the ThreshouldAuto property value is "True", the threshold values are calculated each time the vision sequence is executed. This enables the blob detection to function properly even when the lightening system is changed.

Note:
If ThreshouldAuto is set to "True", the setting value declines to the threshold value which is capable to detect blobs (at least one blob can be detected) even when the image is homogeneous (all black or white), such as when the target work piece is unable to be captured.

Vision Guide 7.0 (Ver.7.4) Software Rev.1
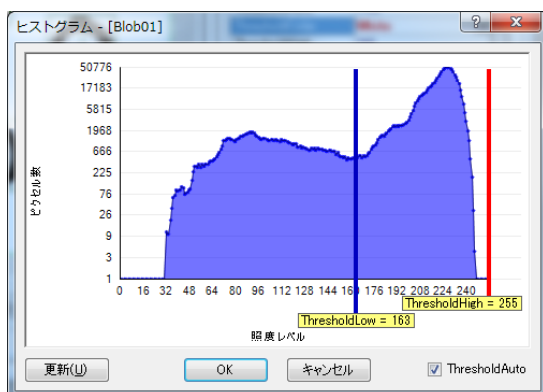
9. Using Vision Guide Statistics

Selecting <Statistics> from the [Vision] menu or clicking on the <Statistics> toolbar button opens the [Statistics] dialog.

Statistics calculations are a built-in feature of Vision Guide 7.0 and can prove to be quite useful.

Vision Guide 7.0 automatically keeps track of result data for future statistical reference for each vision object every time it is run from the Vision Guide window, Run window, or Operator window. This means that each time you run a vision object (regardless of the method used to initiate the vision object execution) the result data for that instance of execution is stored for use in calculating statistical information about that object.

Since most vision applications can be solved in a variety of ways, many times you may want to try multiple methods and compare the results. This is where the statistics feature is invaluable. It allows you to run reliability tests on vision objects thereby pinpointing which vision solution is best for your application. Best of all, these tests can be initiated from the SPEL$^+$ language or even from the Vision Guide point and click interface where no programming is required.

9.1 Dialog Box Options/ Info

Option	Description
Object	Dropdown list for selecting which object to review statistics for.
Found	Displays the number of times the object was found vs. the number of times it was run. For example, 5/6 means the object was run 6 times and found 5.
Reset Current	Reset the statistics calculations for the current vision object. This clears all data for the current object only.
Reset All	Reset the statistics calculations for all vision objects in the sequence.
Close	Closes the [Statistics] dialog. All results remain in place.
Help	Opens the on-line help for Statistics.
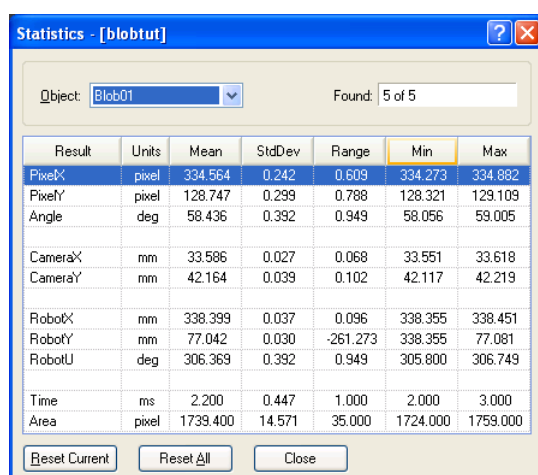
9.2 Vision Objects and Statistics Supported

The following list of vision objects are supported for use with statistics:
Blob, Correlation, Geometric, Edge, Line, Point, Polar, CodeReader, ColorMatch, BoxFinder, CornerFinder, LineFinder, LineInspector, ArcFinder, ArcInspector, DefectFinder

The following statistics are available from the [Statistics] dialog:

Statistic	Description
Mean	The computed average for all found objects of the specified object.
Standard Deviation	The "Sample Standard Deviation" which is basically the square root of (($\sum$ (xi - mean)2) / n - 1). Only found object results are used for this computation.
Range	The range of the results which is (Min - Max). This is computed for found objects only.
Min	The minimum result of all found objects of the specified object.
Max	The maximum result of all found objects of the specified object.

9.3 Vision Object Results Supported

The following list of vision object results are displayed in the [Statistics] dialog. Not all results are returned for every object type. See the individual result descriptions in the Vision Guide 7.0 Properties and Results Reference Manual for more information.

Result	Description
PixelX	The X coordinate position of the found part's position in pixel coordinates.
PixelY	The Y coordinate position of the found part's position in pixel coordinates.
Angle	The amount of rotation associated with a part. This shows the amount of rotation a part may have vs. the originally taught position.
Area	The number of connected pixels found for a blob.
CameraX	The X coordinate position of the found part's position in the camera coordinate frame.
CameraY	The Y coordinate position of the found part's position in the camera coordinate frame.
Length	The length of a Line object in millimeters. (Requires that a calibration be done)
PixelLength	The length of a Line object in pixels. (Does not require calibration.)
RobotX	The X coordinate positions of the found part's position with respect to the Robot World Coordinate System.
RobotY	The Y coordinate positions of the found part's position with respect to the Robot World Coordinate System.
RobotU	The U coordinate positions of the found part's position with respect to the Robot World Coordinate System. (Rotational orientation of the part in robot space.)
Time	The amount of time which it took to execute the vision object. (i.e. to find the part.)
Score	The measure of how well a particular part was found. (i.e. how well a correlation object matches it's model, or the strength of transition for an Edge object.)

9.4 Vision Object Statistics Available From SPEL+

The following list of Vision Object Statistics results are available from the SPEL+ Language.

Vision Object	Statistics Results Supported from SPEL+
Blob	AngleMax, AngleMean, AngleMin, AngleStdDev AreaMax, AreaMean, AreaMin, AreaStdDev CameraXMax, CameraXMean, CameraXMin, CameraXStdDev CameraYMax, CameraYMean, CameraYMin, CameraYStdDev PixelXMax, PixelXMean, PixelXMin, PixelXStdDev PixelYMax, PixelYMean, PixelYMin, PixelYStdDev RobotXMax, RobotXMean, RobotXMin, RobotXStdDev RobotYMax, RobotYMean, RobotYMin, RobotYStdDev RobotUMax, RobotUMean, RobotUMin, RobotUStdDev TimeMax, TimeMean, TimeMin, TimeStdDev
Correlation	AngleMax, AngleMean, AngleMin, AngleStdDev CameraXMax, CameraXMean, CameraXMin, CameraXStdDev CameraYMax, CameraYMean, CameraYMin, CameraYStdDev PixelXMax, PixelXMean, PixelXMin, PixelXStdDev PixelYMax, PixelYMean, PixelYMin, PixelYStdDev RobotXMax, RobotXMean, RobotXMin, RobotXStdDev RobotYMax, RobotYMean, RobotYMin, RobotYStdDev RobotUMax, RobotUMean, RobotUMin, RobotUStdDev ScoreMax, ScoreMean, ScoreMin, ScoreStdDev TimeMax, TimeMean, TimeMin, TimeStdDev
Geometric	AngleMax, AngleMean, AngleMin, AngleStdDev CameraXMax, CameraXMean, CameraXMin, CameraXStdDev CameraYMax, CameraYMean, CameraYMin, CameraYStdDev PixelXMax, PixelXMean, PixelXMin, PixelXStdDev PixelYMax, PixelYMean, PixelYMin, PixelYStdDev RobotXMax, RobotXMean, RobotXMin, RobotXStdDev RobotYMax, RobotYMean, RobotYMin, RobotYStdDev RobotUMax, RobotUMean, RobotUMin, RobotUStdDev ScoreMax, ScoreMean, ScoreMin, ScoreStdDev TimeMax, TimeMean, TimeMin, TimeStdDev
Edge	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev CameraYMax, CameraYMean, CameraYMin, CameraYStdDev PixelXMax, PixelXMean, PixelXMin, PixelXStdDev PixelYMax, PixelYMean, PixelYMin, PixelYStdDev RobotXMax, RobotXMean, RobotXMin, RobotXStdDev RobotYMax, RobotYMean, RobotYMin, RobotYStdDev ScoreMax, ScoreMean, ScoreMin, ScoreStdDev TimeMax, TimeMean, TimeMin, TimeStdDev

Vision Object	Statistics Results Supported from SPEL+
Polar	AngleMax, AngleMean, AngleMin, AngleStdDev
	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	RobotUMax, RobotUMean, RobotUMin, RobotUStdDev
	ScoreMax, ScoreMean, ScoreMin, ScoreStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
Line	AngleMax, AngleMean, AngleMin, AngleStdDev
	LengthMax, LengthMean, LengthMin, LengthStdDev
	PixelLengthMax, PixelLengthMean, PixelMeanMin, PixelLengthStdDev
Point	AngleMax, AngleMean, AngleMin, AngleStdDev
	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
CodeReader	AngleMax, AngleMean, AngleMin, AngleStdDev
	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	RobotUMax, RobotUMean, RobotUMin, RobotUStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
ColorMatch	CameraXMax, CameraXMean , CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
BoxFinder	AngleMax, AngleMean, AngleMin, AngleStdDev
	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	RobotUMax, RobotUMean, RobotUMin, RobotUStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev

Vision Object	Statistics Results Supported from SPEL+
CornerFinder	AngleMax, AngleMean, AngleMin, AngleStdDev
	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	RobotUMax, RobotUMean, RobotUMin, RobotUStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
LineFinder	AngleMax, AngleMean, AngleMin, AngleStdDev
	LengthMax, LengthMean, LengthMin, LengthStdDev
	PixelLengthMax, PixelLengthMean, PixelMeanMin, PixelLengthStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
LineInspector	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	LengthMax, LengthMean, LengthMin, LengthStdDev
	PixelLengthMax, PixelLengthMean, PixelMeanMin, PixelLengthStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
ArcFinder	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	RobotUMax, RobotUMean, RobotUMin, RobotUStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev
ArcInspector	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	LengthMax, LengthMean, LengthMin, LengthStdDev
	PixelLengthMax, PixelLengthMean, PixelMeanMin, PixelLengthStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev

Vision Object	Statistics Results Supported from SPEL+
DefectFinder	AngleMax, AngleMean, AngleMin, AngleStdDev
	AreaMax, AreaMean, AreaMin, AreaStdDev
	CameraXMax, CameraXMean, CameraXMin, CameraXStdDev
	CameraYMax, CameraYMean, CameraYMin, CameraYStdDev
	PixelXMax, PixelXMean, PixelXMin, PixelXStdDev
	PixelYMax, PixelYMean, PixelYMin, PixelYStdDev
	RobotXMax, RobotXMean, RobotXMin, RobotXStdDev
	RobotYMax, RobotYMean, RobotYMin, RobotYStdDev
	RobotUMax, RobotUMean, RobotUMin, RobotUStdDev
	TimeMax, TimeMean, TimeMin, TimeStdDev

10. Tutorial

10.1 Quick Start: A Vision Guide 7.0 Tutorial

10.1.1 Tutorial Overview

The purpose of this chapter is to walk you through a simple vision application to help introduce you to some of the Vision Guide 7.0 basic usage concepts and to show you how easy it is to use. In many cases we will simply explain the steps to follow but will not explain the details behind what was done. Then in the later chapters, you can read about the details.

We will not be using actual parts for this tutorial but instead we will use simple drawings of objects that you can then make copies of and put under a camera to follow the tutorial. This will guarantee that everyone has the same results when trying this tutorial.

This tutorial will show you how to create a simple application to use vision to find a part and then move the robot to it. It is assumed for the purposes of this tutorial that your robot is a SCARA type with a camera mounted at the end of the 2nd arm link.

This chapter will include the following sections:

- Items required for this tutorial.

- Camera lens configuration.

- Creating a new EPSON RC+ 7.0 Project.

- Creating a new vision sequence.

- Using a Blob object to find a part.

- Writing a SPEL+ program to interact with the vision sequence.

- Calibrating the robot with the camera.

- Using vision to instruct the robot to move to the part.

- Finding and moving to multiple like parts.

In *Items Required for this Tutorial* later in this chapter, you will find two pages with printed targets for our vision tutorial. These pages will be used throughout the tutorial.

Following the target pages is a section entitled *Start EPSON RC+ 7.0 and Create a New Project*. This is where we will start the actual tutorial.

10.1.2 Items Required for this Tutorial

This tutorial will guide you through using an EPSON robot with Vision Guide 7.0. It is assumed that you are comfortable using EPSON RC+ 7.0 and EPSON Robots. If you are a little uncertain as to how to use EPSON RC+ 7.0 you probably will want to spend a little time reviewing it prior to starting this tutorial. The following items will be required to work through this tutorial:

- EPSON RC+ 7.0 must be installed on your PC.

- A camera for Vision Guide 7.0 must be installed and working properly.

- An EPSON robot should be mounted to a base with a work surface attached in front of the robot so the robot can access the work surface.

- If the robot is a SCARA robot, the camera should be attached to the robot at the end of the 2nd arm link. If the robot is a 6 axis robot, the camera should be attached to the robot's J6 flange. The camera mounting bracket for the particular robot you are using is available from EPSON. The camera should be mounted so that it is facing downward.

NOTE ☞ Make sure that the PC's main power, controller power, and camera power are turned off when attaching the Ethernet cables between the PC, camera, controller, and Ethernet switch (if used). Failure to turn power off could result in damage.

- Copies of the pages that contain drawings of the target parts should be made available to put under the camera. (The target part drawings can be found in the next section following this list.)

- A camera lens must be selected that will allow a field of view of about 40 mm × 30 mm. In our tests here we have been using a Vision Guide 7.0 mobile camera (CV1 with NET 1044BU camera) with 16 mm lens at a distance of about 210 mm from the work surface. However, to get proper focus will require some adjustment. We'll help guide you through this.

- We won't be picking up any parts in this tutorial. However, we will be moving the robot to the part positions drawn on our target sheets. You will need to attach a gripper or rod to the Z axis shaft. A rod that comes straight down the center of the Z axis shaft will work fine. This gripper or rod will be used for calibration and also moving to the target parts on the target pages.

A single target feature for tutorial (a drawing of a washer)
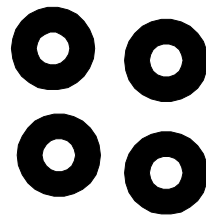
Multiple target features for tutorial (multiple washers)

10.1.3 Start EPSON RC+ 7.0 and Create a New Project

(1) Turn on your controller.

(2) Double click the <EPSON RC+ 7.0> icon on the Windows desktop to start EPSON RC+ 7.0.

(3) Click on the Project menu from the EPSON RC+ 7.0 menu bar.

(4) Click on the New Project menu entry of the Project menu. The [New Project] dialog will appear.

(5) Type a name for the new project. We suggest that you use your name with the letters "tut" attached to the end. That way each individual who tries this tutorial will have a unique name for his or her project. For our project here we will use the name "vgtut". After you type a name in the name field, click the <OK> button. You have just created a new project called xxxxtut.

10.1.4 Create a New Vision Sequence

(1) Once you have created a new EPSON RC+ 7.0 project, you will notice that many of the toolbar icons are enabled (no longer grayed out). Look for the <Vision 📷> toolbar button. Click on the vision toolbar button to open the Vision Guide window.

(2) Before we can do anything in the Vision Guide 7.0 window we must first create a vision sequence. This is accomplished by clicking on the <New Sequence ☐> toolbar button. Remember that this toolbar button is located on the Vision Guide window toolbar and not on the EPSON RC+ 7.0 main toolbar. The [New Sequence] dialog will now appear.

(3) Enter the name "blobtut" for the new vision sequence and then click the OK button. We will be using the name "blobtut" later in our EPSON RC+ 7.0 code so be sure to type this name in exactly as spelled above without the quote marks. We have now created a new vision sequence called "blobtut" and everything we do from now on will be for this vision sequence.

10.1.5 Camera Lens Configuration for Tutorial

We mentioned earlier that our target field of view for this tutorial should be about 40 mm × 30 mm. We have found that a 16mm lens and 1 mm extension tube work fine for this field of view at distances ranging from 167 mm up to 240 mm. This means that a 16 mm lens and 1 mm extension tube will probably also work for your tutorial.

If you have not attached a lens to your camera and tried to focus your lens, it would probably be a good idea to do this at this time. In the chapter Installation there is an explanation on how to determine which lens to use and focusing. You will probably want to review this information before proceeding if you are not familiar with how to select a camera lens and focus the camera to your part.

In order to check your focus, obviously you will need to view an image from Vision Guide 7.0. Since we already have the Vision Guide window open, we now only need to make sure we can properly see the target feature on the target sheet. The steps to accomplish this are described in the next section.

Position the mobile camera over the target sheet and focus the lens

(1) Get the copies of the target sheets that you made earlier. Select the target sheet with the single washer feature printed in the center and put it in a location on the work surface where the robot can then position the camera over it easily. The best position is normally directly in front of the robot.

(2) Now move the robot so that the camera is located over the picture of a washer. You should be able to move the robot manually without having to turn servo power on.

(3) You should be able to see the target feature (the washer) in the image display of the Vision Guide window. Bring the target feature into focus by adjusting the camera lens focus. If you cannot see the target feature or cannot bring it into proper focus go to the section titled *Checking and Adjusting for Proper Focal Distance* in the chapter *Installation*. This section will explain the details on selecting and focusing a camera lens.

The camera should now be in position directly over the target feature drawn on the target sheet. You should be able to see the target clearly on the image display of the Vision Guide window. The Vision Guide window should now appear as shown in the figure below. Notice the target feature (the washer) is shown in the center of the image display of the Vision Guide window.
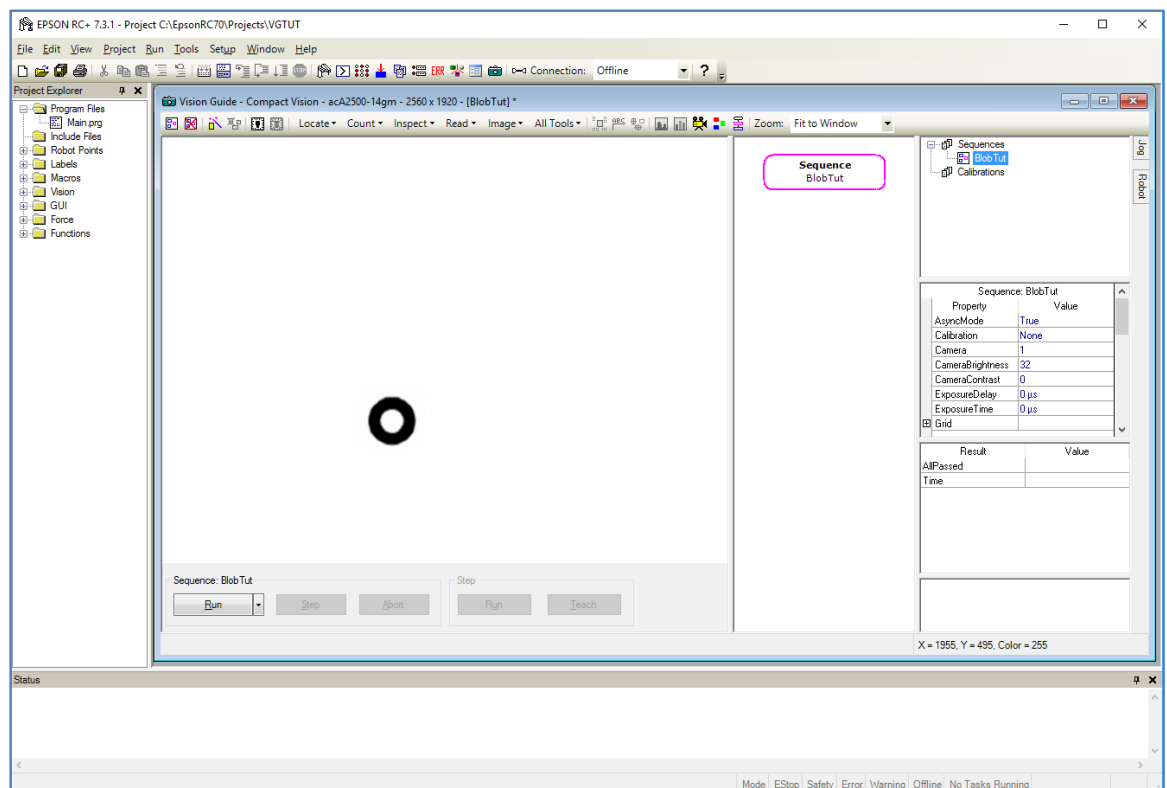
The Vision Guide window with target feature in center of image display

10.1.6 Using a Blob Object to Find a Part

Now that the target feature (washer) is located in the center of the image display of the Vision Guide window, we can create a Blob object to find the washer. The steps to create a Blob object, configure it, and then find the washer are shown next.

Step 1: Create a New Blob Object

(1) Click the <New Blob> toolbar button ![blob] on the Vision Guide 7.0 toolbar with the left mouse button and then release the button. (Do not continue holding the left mouse button once you have clicked on the New Blob toolbar button.)

(2) Now move the mouse downward towards the image of the washer in the center of the image display. As you move off the Vision Guide 7.0 toolbar the mouse pointer will change to the Blob object icon.

(3) Continue moving the mouse to the center of the image display and click the left mouse button. This will position the new Blob object onto the image display.

Your screen should look similar to the one shown in the figure below.
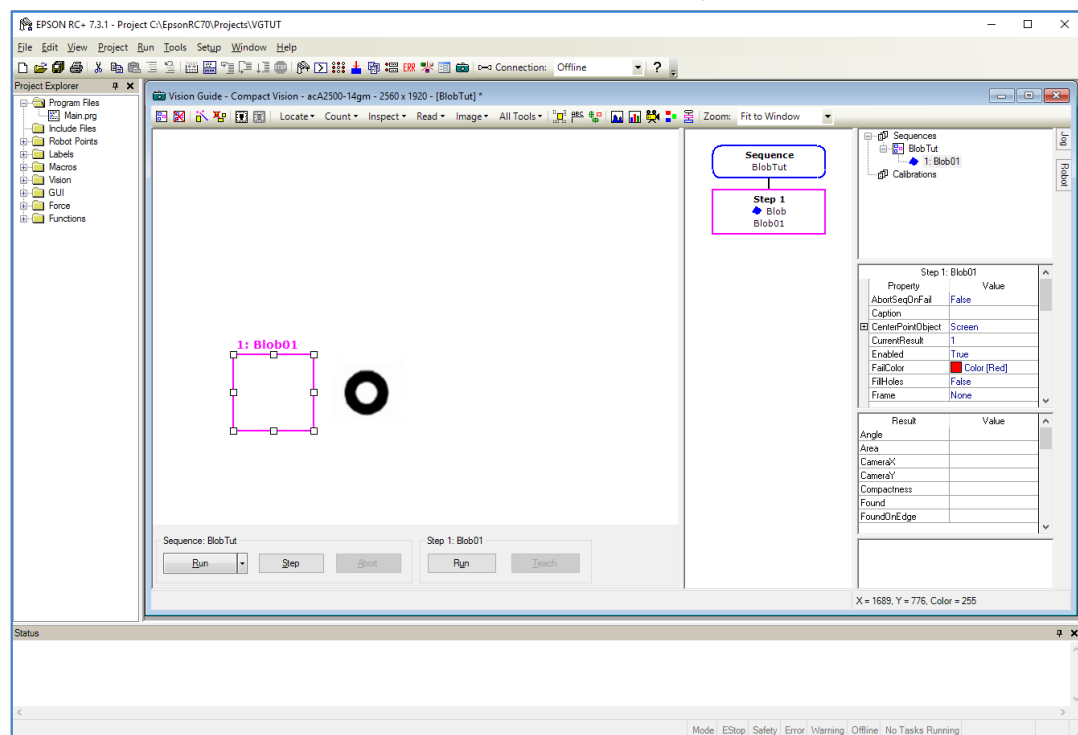
Vision Guide window with new Blob object "Blob01"

Vision Guide 7.0 (Ver.7.4) Software Rev.1

Step 2: Position and Size the Blob Object

We now need to set the position and size of the search window for the Blob object. The search window for the "Blob01" Blob object is the box you see to the left of the washer in the figure below. Let's make this search window large so that we can search nearly the entire field of view for the washer.

(1) Move the mouse pointer over the name label of the Blob object and hold the left mouse button down. While continuing to hold the left mouse button drag the Blob object to the upper left corner of the image display so that the upper left corner of the search window is almost touching the upper left corner of the image display.

(2) Move the mouse pointer over the lower right sizing handle of the Blob01 search window and hold the left mouse down. While continuing to hold the left mouse down, drag the lower right sizing handle to the bottom right corner of the image display. The Blob object's search window should now cover the entire image display. This will allow us to find any blob that is located within the camera's field of view.

The Screen shot below shows the search window for the "Blob01" Blob object repositioned and sized so that it covers the entire image display.
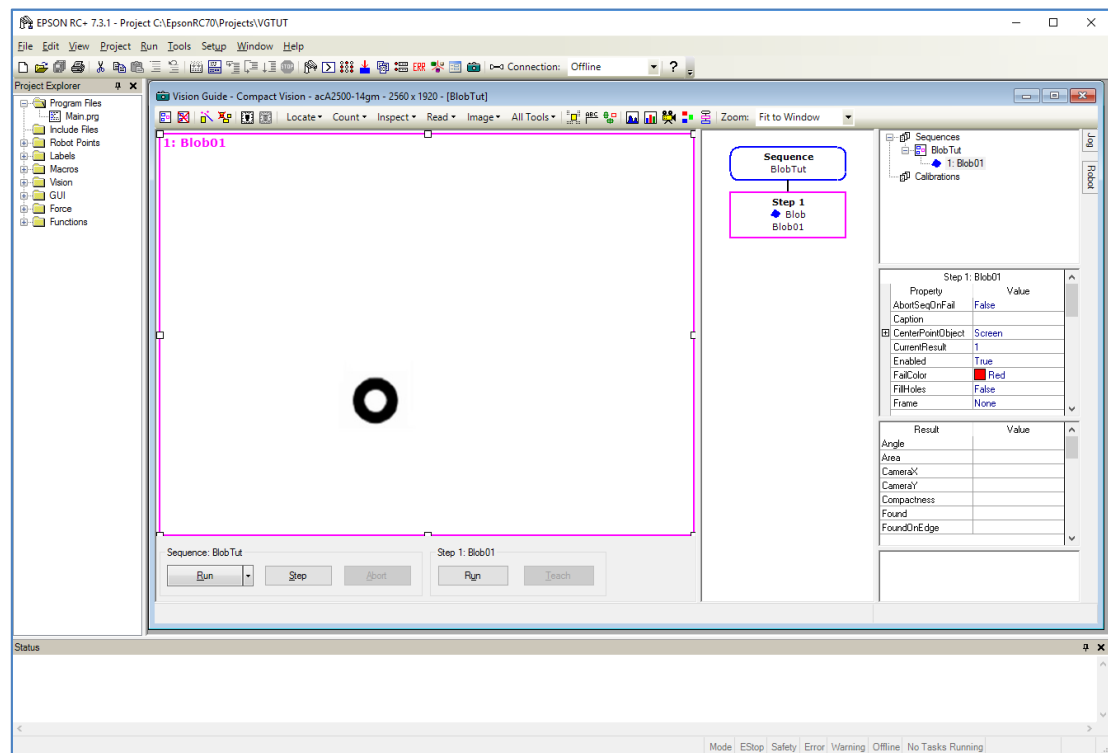
"Blob01" Blob object with large search window

Step 3: Set Properties and Run the Blob Object

The search window is now large enough for the washer to be seen inside of the search window. We are now ready to test our Blob object to make sure it can find the washer.

(1) Click on the Blob01 object located in the tree in the upper right side of the Vision Guide window. This will display the properties and results for the Blob01 object.

Step 1: Blob01	
Property	Value
AbortSeqOnFail	False
Caption	
⊞ CenterPointObject	Screen
CurrentResult	1
FailColor	■ Color [Red]
Frame	None
FrameResult	1
Graphics	All
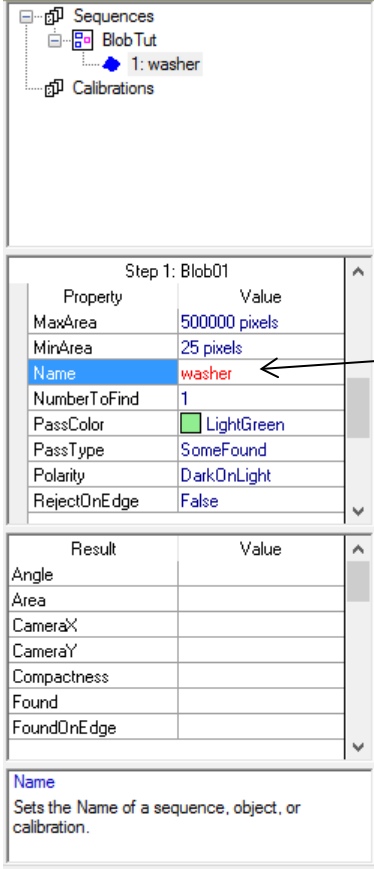
Result 1 of 1	Value
Angle	
Area	
CameraX	
CameraY	
Compactness	
Found	
FoundOnEdge	

Caption
Optional caption for the object that replaces the default caption

(2) Look at the Properties list and find the Name property. Double click on the Value field for the Name property to cause the current name to be highlighted. Now enter the name "washer". We have just changed the name of our Blob object to "washer". If you look at the top of the [Object] tab in the Name drop down list, and the name label on the search window, you will see that the name has been changed in both places.

Enter the name "washer" here

Properties for Blob object

(3) While you are looking at the Properties list you might also want to examine the Polarity property. Since we just created a new blob, the default value for Polarity is DarkOnLight, which means find a dark blob on a light background. We could also change this property to LightOnDark. But since we want to find a dark blob (the washer) on a light background, leave the Polarity property as is.

(4) We are now ready to run the Blob object "washer". To run the object click on the Run button located at the bottom right of the run panel. This will cause the Blob object "washer" to execute and in this case the Blob object will find a blob that looks like a washer. You can tell that the Blob was found by examining the color of the search window after running the Blob object. The search window will be green if the Blob object finds a blob and red if it does not. (You can also examine the Results list to see if the Blob was found or not. More on that next.)

(5) Now try moving the paper with your target washer just a little and click the Run button again. Be sure to keep the washer inside the search window that you created. You can see the new position of the blob is found and highlighted in green on the image display window. (If you move the paper so that the washer is outside of the search window boundary, then the Blob object will not find the washer and you can see this because the search window will turn red when the object is not found.)

Step 4: Examining the Results

Now that you have run the Blob object called "washer" you can examine the results that are returned for this object. The results are displayed in the Results list located just below the property list.

(1) Look at the Results list for the result called Found. The value of the Found result should be True at this point because we just found the blob. If no blobs were found then the Found result would display False.

(2) You can also see the Area result at the top of the Results list. This shows the area of the blob that was found.

(3) Use the scroll bar to move the results list down to the bottom. At the bottom of the Blob results list you will see the Time result. This result tells us how much time it took to find this blob.

Click the scroll bar button here (or use the slider) to scroll down the Results list

Results list scrolled down to show Time result

⚠️ CAUTION	■ Ambient lighting and external equipment noise may affect vision sequence image and results. A corrupt image may be acquired and the detected position could be any position in an object's search area. Be sure to create image processing sequences with objects that use search areas that are no larger than necessary.

Step 5: Saving Your Vision Sequence

We should probably save our work at this time. As with any application, it's a good idea to periodically save your work often. The Project Management feature of EPSON RC+ 7.0 causes everything related to this project to be saved at once. This is done in one easy step as shown below:

Click on the <Save 🖫> toolbar button. It is located on the left side of the EPSON RC+ 7.0 main toolbar. When changes have been made anywhere in your project, the <Save> toolbar button will be shown in blue.

We have now successfully created a new Blob object; properly positioned and sized the search window; and we ran the Blob object to find a blob. Let's take the next step and write a simple program to run our vision sequence from the SPEL+ language and retrieve some of the results for use in our application.

10.1.7 Writing a SPEL⁺ Program to Work with the Vision Sequence

One of the most powerful features of Vision Guide 7.0 is that any vision sequences that are created from the point and click environment can be used from the SPEL+ language. This means that vision sequences are a core part of EPSON RC+ vision applications and are not just a prototyping tool for which you later have to rewrite everything in SPEL+. Vision sequences and the SPEL+ language are integrated together to give you the best of both worlds: the ease of use of a point and click vision development environment and the power and flexibility that a language provides. Let's write a quick program to find whether or not the blob was found, check the area of the blob, and then print a message with this information.

Open the Program File Called MAIN.PRG

You should already know how to open the main.prg program file since you are familiar with the EPSON RC+ 7.0 environment. However, for those of you who may not know how to open a file we have included the basic steps below:

(1) Click on the <Open File 📂> toolbar button on the EPSON RC+ 7.0 toolbar. This will open the [Open File] dialog that contains the program called main.prg that is automatically created when you first create a new project. You will see the following [Open File] dialog box:
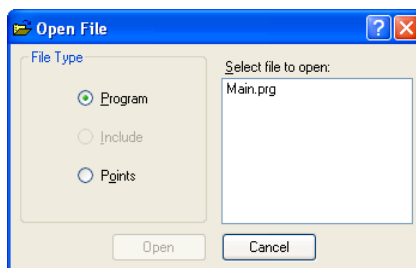
The EPSON RC+ Open File Dialog

(2) As you can see in the [Open File] dialog, the main.prg program file should already be highlighted since you haven't yet created any other programs. Go ahead and click on the <Open> button at the bottom left of the dialog to open the main.prg program file.

Create a SPEL⁺ Program to Work with our Vision Sequence

Shown below is an example program that will run the vision sequence "blobtut" and then check some of the properties associated with the Blob object called "washer". For example, we will check if the blob is found or not. If it is we will display a "The washer was found!" message and the area of the blob. If no blob is found we will display a "The washer was not found!" message.

You should now see an editor window with MAIN.PRG displayed on the Title bar. The cursor will be located at the beginning of the 1st line in the edit window. (The perfect position to begin typing in our program.) Go ahead and enter the following program into the editor window. Don't worry about the capital vs. small letters. The editor automatically capitalizes all keywords.

```
Function main
  Real area
  Boolean found

  VRun blobtut
  VGet blobtut.washer.Found, found

  If found = TRUE Then
    VGet blobtut.washer.Area, area
    Print "The washer was found!"
    Print "The washer area is: ", area, " pixels"
  Else
    Print "The washer was not found!"
  EndIf
Fend
```
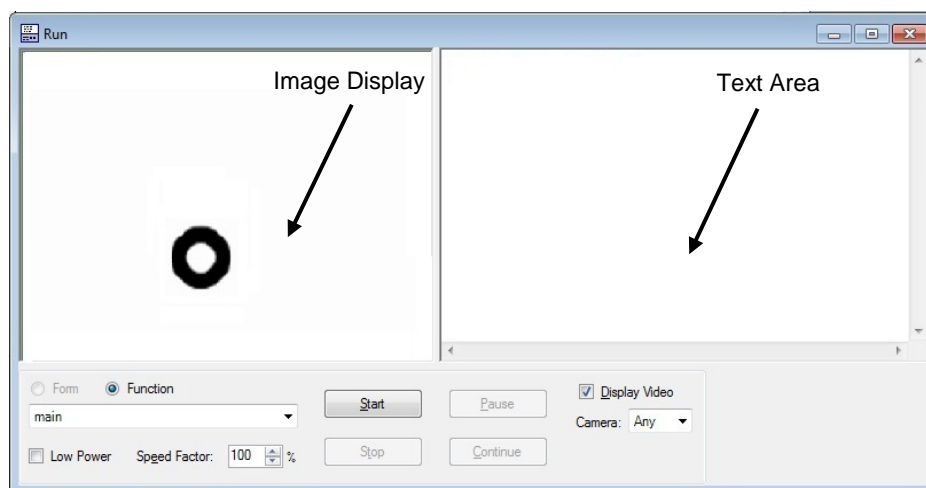
Running the main Function

You should be familiar with the Run window for EPSON RC+ 7.0. We will use the Run window to run our sample program that was created in the previous section. The steps to accomplish this are shown below:

(1) Click on the <Run 🖩 > toolbar button on the EPSON RC+ 7.0 toolbar. This will cause the Run window to appear. Notice that in the figure below, the Run window is split into 2 parts. The left half of the Run window is the image display (note the washer shown in the center) and the right half of the Run window is the text area which is used for displaying text messages.

NOTE
☞
If the Run window only shows the Text Area and doesn't appear with both an image display and a text area, click on the Display Video check box on the Run window.

Run window with Image Display and Text Area

(2) Click on the Start button located at the bottom left corner of the Run window. This will cause the function called "main" to run.

(3) You can see sample results in the figure in Step (4) from running the function "main". Notice that when the blob is found, it is highlighted in green in the image display on the left side of the Run window. The right side of the Run window displays text that states that the blob was found and the area of the blob.

(4) Double click the Control-menu box of the Run window that is located in the upper left corner of the Run window. This will close the Run window.
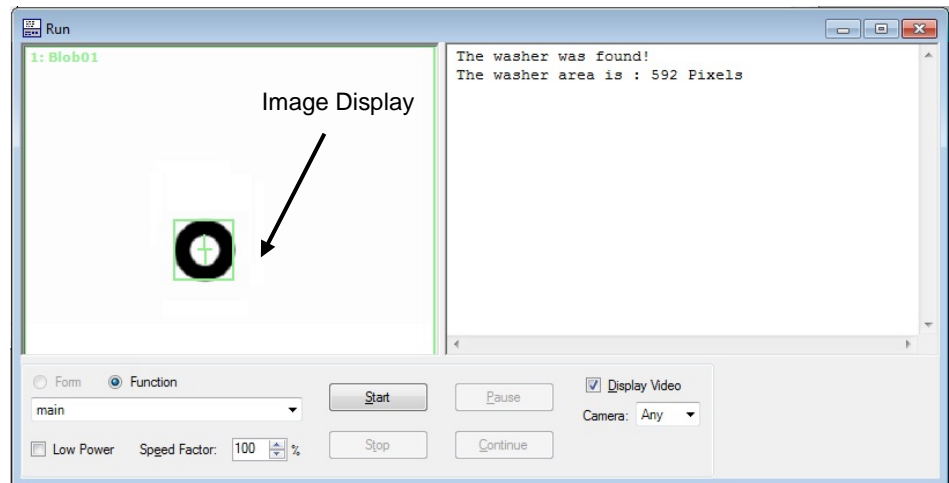
A sample Run window display after running "main"

10.1.8 Calibrating the Robot Camera

This part of the tutorial will guide you through how to calibrate the robot with the mobile camera that is mounted to the end of joint 2 on a SCARA. Calibration provides the mechanism to allow the vision system to understand the robot's world coordinate system. Once calibrated, the vision system can be used to find the robot coordinate position of parts for the robot to pick up. However, before we start calibration we need to prepare the robot for use by turning the motors on and calibrating (homing) the robot.

Step 1: Turn On Motors

(1) Click on the <Robot Manager 🗔> toolbar button on the EPSON RC+ 7.0 main toolbar. You will see the following dialog appear.
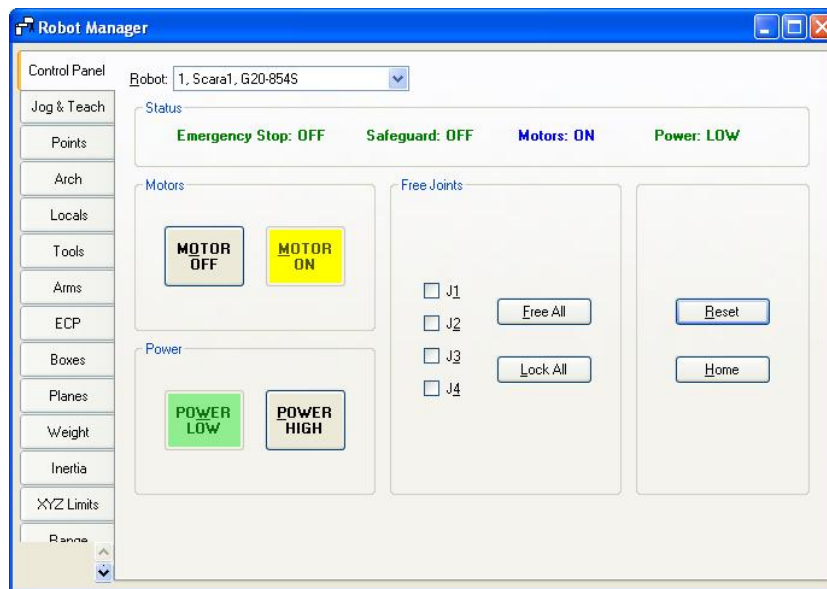
The Robot Control Panel

(2) Click the <MOTOR ON> button in the Motors group of the Control Panel.

(3) A message box will appear asking if you are "Ready to turn robot motors ON". Click the <Yes> button. This will turn the robot's motors on.

Step 2: Use the Tool Setting to create a new tool

In order to use the robot with a vision system for a guidance application requires that precise measurement of the position of the tool or gripper mounted at the end of the arm. Some tools are offset from the center of the Z-axis flange and other tools are mounted in line with the center of the Z-axis. Regardless of how careful we are when mounting robot tooling, you can almost always guarantee that there will be some offset from the center of the Z axis. For this reason we must use the Tools feature of SPEL+ to compensate for this adjustment.

A good understanding of tools is required to use Vision Guide for robot guidance. We will use the Tool Wizard feature to create a tool for our calibration rod (gripper).

(1) Click on the [Tools] tab.

(2) Click the <Tool Wizard> button.

(3) For the first point in the Tool Wizard, jog the robot to the target position (position of washer on your paper target) so that the rod or gripper is located exactly in the center of the washer. You will need to jog the robot downward so that you are close to touching the paper but don't quite touch the paper. (Anywhere from 5-10 mm space above the paper will be fine. You just need to jog as close to the paper as is required to be able to see the rod (or gripper) well enough to center it on the washer.

(4) For the second point in the Tool Wizard, jog the U axis approximately 180 degrees, then jog the rod (or gripper) until it is centered over the washer.

(5) Click the <Finish> button to define the new tool and exit the Tool Wizard. We have just defined the parameters for Tool 1.

The new tool values for Tool 1 for our Tutorial will appear as shown in the figure below.
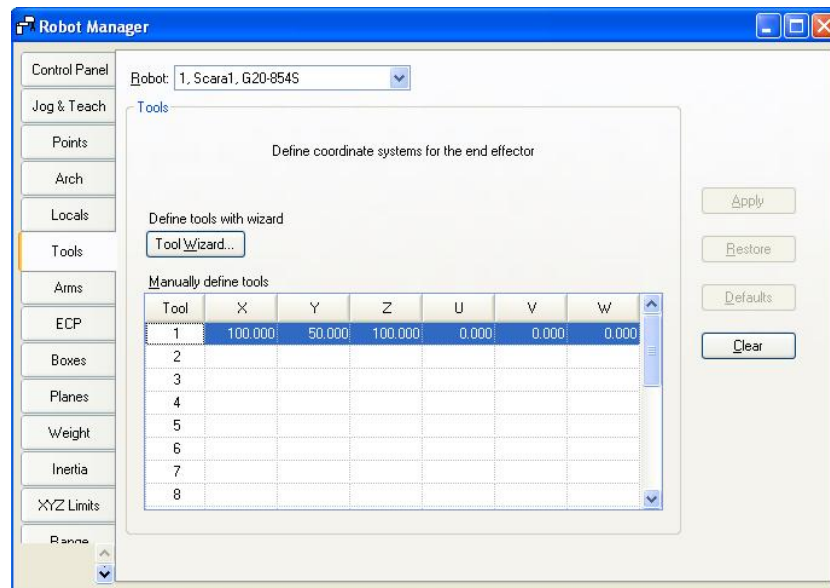
The [Tools] Tab (for defining tools)

Step 3: Test the New Tool

You should now be back in the Jog and Teach Window. And since we haven't jogged the robot since we positioned it above the target position (the washer), the robot's gripper (or rod) should still be located just above the target position.

(1) Jog the robot about 10-15 mm up in the Z direction (+Z). This will get our gripper (or rod) away from the work surface just in case there is a problem.

(2) Click on the down arrow for the Tool drop down list which is located in the upper right corner of the Jog and Teach Window.

(3) Click on "1" since we want to select Tool 1. (Remember this was the tool we just taught in Step 4.)

(4) Now try jogging the U axis in the positive or negative direction. You should see the tip of the gripper (or rod) stay in its' current XY position while the robot rotates around this point. (This is easily seen for tools that are offset from the center of the Z-axis. If you are using a rod that comes straight down from the Z-axis, this may be difficult to see.

(5) You should expect some movement of the gripper (rod) from the XY position when the U axis is moving. However, the gripper should return to the XY position of the target (the washer) after each U axis jog step is complete.

(6) If your tool doesn't appear to be working properly, go back to the beginning of step 2 and try again.

(7) Close the Robot Manager. We are now finished defining and testing the tool.

Step 4: Starting the Camera Calibration Process

We are now ready to calibrate the robot with the mobile camera.

(1) Click on the <Vision 📷 > toolbar button on the EPSON RC+ 7.0 toolbar to bring the Vision Development Window back to the front position on the screen.

(2) Click on the <New calibration 🔲 > toolbar button on the Vision Guide window. This will open the [New Calibration] dialog.

(3) Type in the name "downcal" without the quotes and click the OK button.

(4) The orientation will be for a mobile camera mounted on robot joint 2 so change the CameraOrientation property to Mobile J2.

(5) Change the RobotTool property to 1. This will select Tool 1 to be used when teaching the reference point for the calibration.

(6) Set the TargetSequence property to blobtut. We will use this sequence for finding the parts during calibration.

Step 5: Teaching the Calibration Points

(1) Click the <Teach Points> button located below the video display.

(2) The Vision Guide window will change the Teach Points mode, as shown below. Notice the message at the top of the Vision Guide window that requests that you "Jog to top left camera position", as shown in the figure below. This means to jog the robot (and camera) to a position where the washer can be seen in the upper left corner of the image display of the Vision Guide window. The figure below shows the approximate position where you should jog the robot. As you can see in the figure, this is the 1st of 9 camera positions required for calibration.
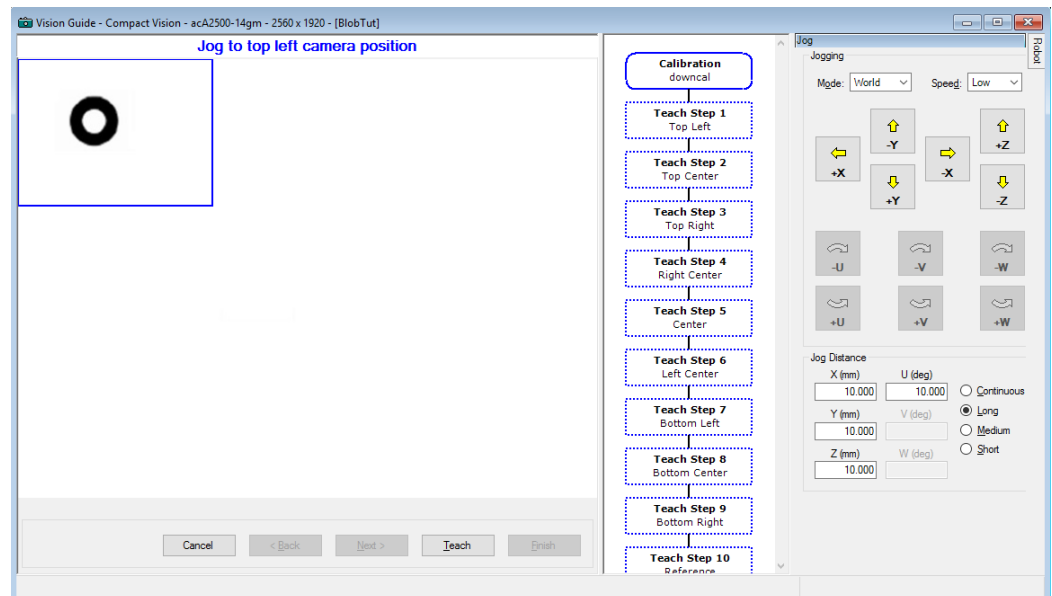
Teaching camera positions for calibration

(3) **Teaching the 1st Camera Position:** Jog the robot so that the camera is positioned where you can see the washer in the top left corner position of the image display of the Vision Guide window.

(4) Click the Teach button on the Vision Guide window.

(5) **Teaching the 2nd Camera Position:** Jog the robot to the right of the 1st camera position so that the camera is positioned where you can see the washer in the top center position of the image display of the Vision Guide window.

(6) Click the Teach button on the Vision Guide window.

(7) **Teaching the 3rd Camera Position:** Jog the robot to the right of the 2nd camera position so that the camera is positioned where you can see the washer in the top right position of the image display of the Vision Guide window.

(8) Click the Teach button on the Vision Guide window

(9) **Teaching the 4th Camera Position:** Jog the robot down from the 3rd camera position so that the camera is positioned where you can see the washer in the center right position of the image display of the Vision Guide window. This position will be below the 3rd camera position starting a zigzag pattern when you refer to all the camera positions in order.

(10) Click the Teach button on the Vision Guide window.

(11) **Teaching the 5th - 9th Camera Positions:** Continue jogging the robot and teaching points as instructed at the bottom of the Vision Guide window for each of the 5th through 9th camera positions. The Vision Guide window will display which position to move to for each camera position. Camera positions 5-9 are described below:

```
5 – center
6 – left center
7 – bottom left
8 – bottom center
9 – bottom right
```

(12) Next, the message shown at the top of the window that says "Jog to fixed reference". This means to jog the robot so that the gripper (rod) is centered on the calibration target part (in this case the washer). At this point you can ignore the video display, since we are aligning the robot gripper with the calibration target. Go ahead and jog the robot so that the gripper is positioned at the center of the washer. The positioning of this is very important so you will need to jog the robot down close to the washer to get a good alignment.

(13) Once you have the robot gripper positioned at the center of the washer, click the <Teach> button located at the bottom of the Vision Guide window.

(14) Jog the Z-axis up high enough to keep the gripper away from the work surface and any possibility to bumping into other objects during calibration because the robot gripper was too low. When we teach each of the 9 calibration points this Z height will be used for each.

(15) After the last camera position is taught a dialog will appear which says "All calibration points are now taught". Click the Finish button to continue.

Teaching the points required for Calibration is now complete.

Step 6: Running the Camera Calibration

Shown below are the final steps required to complete the camera calibration process:

(1) Click the <Calibrate> button on bottom of the Vision Guide window [Calibration] tab.

(2) A message box will appear with the message "CAUTION Robot will move during calibration at Speed 10, Accel 10 Continue?" Click on the <Yes> button to continue calibration. The robot will move through each of the 9 points 2 times during calibration. If you need to abort the calibration while the robot is moving around, click the <Abort> button at the bottom of the [Calibration Cycle] dialog.

(3) After the calibration cycle is complete, the **[**Calibration Complete] dialog appears and displays the calibration result data. Examine the data and then click the <OK> button.

(4) Notice that the calibration results are displayed in the Results list on the [Calibration] tab.

Step 7: Assign the "downcal" Calibration to the blobtut Sequence

Now that the "downcal" calibration has been created, we will need to assign this calibration to our vision sequence (blobtut). This will give the blobtut sequence the ability to calculate results in Robot and Camera world coordinate values.

(1) Click on the [Sequence] tab on the Vision Guide 7.0 window to bring the [Sequence] tab to the front.

(2) Set the Calibration property to "downcal" by clicking once on the value field of the Calibration property, then clicking on the down arrow, and finally clicking on the calibration displayed which is called "downcal". This was the calibration we just finished creating.

(3) Click the [Jog & Teach] tab on the Vision Guide window. To see this tab, you may have to click the right arrow in the upper right corner of the tab control.

(4) Use the jog buttons to position the camera over the washer so that you can see the washer within the image display.

(5) Click on the [Object] tab.

(6) Click on the <Run Object> button to run the "washer"" object. When the part is found look at the Result list. You will see that the CameraX, CameraY, RobotX, RobotY and RobotU results no longer show a "nocal" result. You can now see coordinate position data with respect to the Robot and Camera coordinate systems.

10.1.9 Teaching Points for Vision Guidance

Now we must teach a few points to define the Z height of the washer pickup position, the position where the camera will take a picture of the washer for processing, and a safe position that for this tutorial will act as a starting position.

Step 1: Define the "camshot" Position

The "camshot" position is the position where the robot must be located so that the camera is positioned over the part in a way that allows the part to be seen by the vision system. The robot must be moved towards the part until the washer can be seen in the image display on the screen. It is a good idea to position the robot so that the washer is located in the middle of the image display and not close to one of the edges of the search window. Since we just finished running the washer object, the camera should be in a good position for acquiring an image that contains the washer.

(1) Click on the <Robot Manager> button on the main tool bar, then click on the [Jog & Teach] tab.

(2) Since we want the camera shot position to be taught using Tool 1, check the drop down list box labeled Tool to make sure that it is set to 1. If it is not set to 1, then go ahead and click the arrow on the drop down list box and set the Tool to 1.

(3) Verify that the current point in the Point # field is P0. We want to teach the camshot position as point P0. If it is not point P0, then select P0 in the point # field.

(4) Click on the <Teach> button on the [Jog & Teach] window. You will be prompted for a label. Enter the name "camshot". This will teach the "camshot" position.

Step 2: Define a Safe Position Away from the Washer

We will need a taught point position away from the washer that will be used as a safe position that we will move to at the start of the program.

(1) Select P1 in the point field.

(2) Jog the Z-Axis of the robot up and then jog in X and Y to position the robot in a safe position. This will be like your start position for your program. The robot will always move to this position first before moving to the washer.

(3) Click on the <Teach> button on the [Jog & Teach] page. Enter "safept" for the label. This will teach the "safept" position.

Step 3: Calculate a Z Height for the "washer" Pickup Position

If we were doing an actual application to pick up a real washer, rather than moving to a drawing of a washer, we would need to set a Z height for the washer pickup position. Let's find a good Z height for our washer position.

⚠ CAUTION	■ Be sure to set the Z height for robot motion carefully. If the calculated Z height is incorrect, it may cause a malfunction of the system and/or safety problems.

(1) Click on the <Robot Manager> button on the main EPSON RC+ toolbar. Then click on the [Jog & Teach] tab.

(2) Use the Jog buttons to position the gripper of the robot about 5-10 mm above the washer.

(3) Write down the current Z coordinate value when the robot gripper is just above the washer. This z coordinate will be used in our program later to move the robot to this height.

(4) Select P2 in the point field.

(5) Click the <Teach> button on the [Jog & Teach] window. Enter the label name "washpos". This will teach our initial "washpos" position. (However, the Vision System will be used to calculate a new X and Y position and then move to this point. We will also set a fixed Z height in our program based on the current Z coordinate position.)

10.1.10 Using Vision and Robot to Move to the Part

Now all that's left is to modify our program to cause the vision system and robot to work together to find the position of the washer and then move to it.

Step 1: Modify the SPEL⁺ Program

(1) Click on the <Open File 📂> toolbar button on the EPSON RC+ 7.0 toolbar.

(2) The MAIN.PRG program file should already be highlighted since you still haven't created any other programs. Click on the <Open> button at the bottom left of the dialog to open the MAIN.PRG program file. You should see the following program that we ran earlier in this tutorial.

```
Function main
  Real area
  Boolean found

  VRun blobtut
  VGet blobtut.washer.found, found

  If found = TRUE Then
    VGet blobtut.washer.area, area
    Print "The washer was found!"
    Print "The washer area is:", area, "Pixels"
  Else
    Print "The washer was not found!"
  EndIf
Fend
```

(3) Now modify the program so that it appears as shown on the next page.

NOTE
👉
Remember that the SPEL⁺ language uses apostrophes ' to denote comments. Any characters that follow an apostrophe are considered comments and are not required for the program to execute properly. (This means that you can omit any characters that follow an apostrophe out to the end of the line where the apostrophe exists.)

```
Function main

    '*********************************************
    ' Very important statement below: Use the    *
    ' Z height which you wrote down earlier in   *
    ' Step 3 of "Teaching Robot Points for use   *
    ' with Vision Guidance. Substitute the       *
    ' Z height value (a negative number) which   *
    ' you wrote down in the place of the xx      *
    ' shown below.                               *
    '*********************************************
    #define ZHeight -xx

    Real area, x, y, u
    Boolean found
    Integer answer
    String msg$, answer$

    Power Low              'Run robot at a slow speed and accel
    Tool 1                 'Use Tool 1 for positioning
    Jump safept            'Move robot to safe start position

    Do                     'Continue looping until user stops
       Jump camshot        'Move robot to take picture
       VRun blobtut        'Run the vision sequence blobtut
       VGet blobtut.washer.RobotXYU, found, x, y, u

       If found = True Then
         VGet blobtut.washer.area, area
         Print "The washer was found!"
         Print "The washer area is: ", area, "Pixels"
         washpos = XY(x, y, ZHeight, u) 'Set pos to move to
         Jump washpos
         msg$ = "The washer was found!"
       Else
         msg$ = "The washer was not found!"
       EndIf
       msg$ = msg$ + CRLF + "Run another cycle(Y/N)?"

       Print msg$

       Input answer$

       If Ucase$(answer$) <> "Y" Then
        Exit Do
       EndIf
    Loop
Fend
```

Step 2: Run the Program to Find the Washer and Move to it

(1) Click on the <Run> toolbar button on the EPSON RC+ 7.0 main toolbar. This will cause the program to be compiled and then open the Run window.

(2) Click on the <Start> button on the Run window.

(3) Your program will now find the washer and move the robot to it. After you successfully find the washer, try moving the washer a little and then click the <Yes> button in the dialog that asks if you would like to run another cycle. If the washer is not found, a different dialog will appear asking whether or not you would like to try again. Clicking <No> in either dialog will cause the program to stop running.

11. Using Vision Guide 7.0 in SPEL+

11.1 Overview

Vision Guide 7.0 was designed to keep vision programming at a minimum, yet allow the application programmer to manipulate vision sequences, objects, and calibrations from SPEL+ programs. In most applications, the vision sequences created with the point and click interface do most of the work. It is a simple matter to run the sequences from SPEL+ and use the results to guide the robot, inspect parts, etc. With this type of design, an applications engineer can create sequences without doing any programming before actually designing the application. This can aid in verifying field of view, accuracy, etc. before quoting a job. Then, at design time, he/she can use the sequences that were created in the actual project.

Vision Guide 7.0 is supported by commands in the SPEL+ language. After creating vision sequences with the point and click interface, you can run these sequences and get the results from them in your SPEL+ programs. You can also read and set properties for vision sequences and objects.

This chapter contains instructions on how to use Vision Guide 7.0 commands in SPEL+ programs.

11.2 Vision Guide 7.0 SPEL⁺ Commands

Here is a summary of the vision commands available for use in SPEL+ programs.

Basic Vision Commands

VRun	Runs a vision sequence in the current project.
VGet	Gets a result or property for a sequence or object and stores it in one or more variables.
VSet	Sets a property value for a sequence or object.

Before programming with vision commands

Before you begin programming with vision commands, you need to create vision sequences in an EPSON RC+ 7.0 project. The vision commands you use in your SPEL+ programs will refer to these sequences and the objects created within them.

11.3 Running Vision Sequences from SPEL⁺: VRun

You can run any vision sequence in the current project from a SPEL+ program using the VRun command. For example:

```
Function VisionTest


   VRun seq1
Fend
```

The simple program above will run the sequence called "seq1" and display the graphical results on the Run window or Operator window, depending on where the program is started from.

The actions taken by VRun depend on the RuntimeAcquire sequence property. This property determines if a picture is taken or not before processing the sequence or if a strobe trigger is being used. The table below shows what happens when VRun executes for the different settings of RuntimeAcquire.

RuntimeAcquire	VRun Actions	Usage
0 – None	Do not take picture. Use previous picture. Run each sequence step.	Allows more than one sequence to run on the same picture. Normally, the first sequence takes the picture with RuntimeAcquire set to Stationary. Then the remaining sequences do not take a picture.
1 – Stationary	Acquire new picture. Run each sequence step.	Default. This is the most common way to use VRun. Take a picture and run the sequence.
2 – Strobed	Wait for strobe input. Acquire new picture. Run each sequence step.	Used with strobe input. VRun will return immediately, then the system will wait for strobe trigger before taking a picture and running the sequence.

When the sequence property AsyncMode is True, VRun returns after the camera exposure, and continues in the background to finish the image grab, and then process the sequence. The next vision command for the same sequence, such as VGet, will automatically wait for processing to finish.

When RuntimeAcquire is set to Strobe, VRun arms the trigger, then returns. The system waits in the background for the trigger to fire and grab the image, then the sequence is processed. You may want to wait for the grab to be completed before you retrieve results from the sequence. You can do this by checking the AcquireState sequence result.

```
Function VisionTest
  Integer state
  Boolean passed
  VRun strobedSequence
  ' Wait for image to be grabbed
  Do
    VGet strobedSequence.AcquireState, state
  Loop Until state = 3
  ' Now retrieve the results
  VGet strobedSequence.AllPassed, passed
Fend
```

If you execute VRun with RuntimeAcquire = Strobe, and then you execute a second vision command, such as VGet, without waiting for AcquireState =3, then the second command will wait until the trigger is received and the sequence runs. If the trigger is never received, the SPEL+ task will have to be aborted.

When you run multiple SPEL+ tasks that run vision sequences, the images are grabbed and the sequences are processed in parallel if the cameras are not the same.

If multiple SPEL+ tasks use the same camera and RuntimeAcquire is set to Strobe, then you must only allow one sequence to process at a time using SyncLock and SyncUnlock. In the example below, sequences seq1 and seq2 both use camera 1.

```
Function visTask1
  Integer state
  Do
    SyncLock 1 ' lock access to camera 1
    VRun seq1
    Do
      VGet seq1.AcquireState, state
    Loop Until state = 3
    VGet <some results here>
    SyncUnlock 1 ' unlock camera 1 access
Fend


Function visTask2
  Integer state
  Do
    SyncLock 1 ' lock access to camera 1
    VRun seq2
    Do
      VGet seq2.AcquireState, state
    Loop Until state = 3
    VGet <some results here>
    SyncUnlock 1 ' unlock camera 1 access
Fend
```

For details on SyncLock and SyncUnlock, refer to the *SPEL+ Language Reference Manual*.

Note:

To use an ac2500-14gm/gc GigE camera in strobe mode (external trigger mode), an external strobe light is necessary. If the strobe light is not used, the camera works in the rolling shutter mode and cannot recognize moving objects correctly.

11.4 Accessing Properties and Results in SPEL⁺: VGet, VSet

The VGet and VSet commands can be used to read and set sequence and object properties in SPEL+ programs. For example, you can change a search window size and location, an accept parameter, camera gain, and maximum area to name a few. Nearly all of the properties and results that can be accessed from the Vision Guide 7.0 point and click interface can also be accessed from a SPEL+ program. There are also some special properties that can only be accessed from using VSet or VGet since they set or return multiple results. (e.g. SearchWin, RobotXYU, and ModelWin to name a few.)

A common syntax is used for both VGet and VSet. Each command must first refer to the name of a sequence. In addition, the name of a vision object must follow the sequence name in order to access properties and results for vision objects. A period is used to separate the sequence, object, and property or result name. If multiple results are used, specific results are selected by appending the result number in parenthesis after the result Name.

For sequence properties and results, the following syntax is used:

      VGet *seqName.propName*, *var*   *'put property value in variable*

      VSet seqName.*propName*, *value*   *'set property to value*

For object properties and results, the following syntax is used:

      VGet *seqName.*objName.resultName, var

      VGet seqName.objName.propertyName, var

      VSet seqName.ob*jName.propertyName*, *value*

For object multiple results, the following syntax is used:

      VGet *seqName.objName.resultName(resultnum)*, *var*

The sequence and object names can also be string variables. See *11.5 Using Variables for Sequence and Object Names*.

11.4.1 Using VGet

VGet retrieves a property or result and puts it in a SPEL+ variable. You must supply a variable in your program of the proper data type to receive the value from VGet.

Here is an example of using VGet in a SPEL+ program.

```
Function Inspect
  ' Run the vision sequence
  VRun InspectPart
  Integer i, numberFound
  Real area
  VGet inspPart.Part1.NumberFound, numberFound
  For i = 1 to numberFound
    ' Loop through each item that was found
    ' Get the area of the blob result
    VGet inspPart.Part1.Area(i), area
    Print "Area of result ", i, " is ", area
  Next i
Fend
```

11.4.2 Using VSet

VSet sets a property value at runtime. This allows developers to dynamically adjust property settings from a SPEL+ program. In most cases you can set property values from the Vision Guide window and then run vision sequences from SPEL+ programs with no modifications of the properties. However, for those cases that require dynamic adjustments, the VSet SPEL+ command can be used.

Here is an example of using VSet in a SPEL+ program. Notice that the first call to VSet sets a sequence property. The second call to VSet sets an object property called SearchWin that is used to re-define the position and size of a search window before running the sequence.

```
Function findPart

  ' Set camera gain for sequence "findPart"
  VSet findPart.CameraGain, 100

  ' Set search window for object "part"
  VSet findPart.part.SearchWin, 100, 100, 50, 50

  ' Run the sequenced
  VRun findPart
Fend
```

11.5 Using Variables for Sequence and Object Names

String variables can be used for the calibration, sequence, and object name arguments in the VRun, VGet and VSet commands. The example below uses the string variables seq$ and obj$ to specify which vision sequence and which vision object to work with.

```
Function visTest

  #define PICKZ -100.0
  String  seq$, obj$
  Boolean found
  Real    x, y, u

  seq$ = "test"
  obj$ = "Blob01"
  VSet seq$.Camera, 1
  VSet seq$.Calibration, "CAMCAL1"
  VRun seq$
  VGet seq$.obj$.RobotXYU, found, x, y, u
  If found Then
    pick = XY(x, y, PICKZ, u)
    Jump pick
    On vacuum
    Wait .1
    Jump place
    Off vacuum
    Wait .1
  EndIf
  Jump park
Fend
```

Arrays can also be used for sequence and object name arguments in the VRun, VGet, and VSet commands. See the example below.

```
Function test
  String obj$(10)
  Integer count

  obj$(0) = "corr01"
  obj$(1) = "corr02"
  obj$(2) = "blob01"
  obj$(3) = "blob02"

  For count = 0 to 3
    VRun seqname
    VGet seqname.obj$(count).Found, found
  Next count
Fend
```

11.6 Using Sequence Results in SPEL⁺

After running a sequence with the VRun command, there are several results that can be used in your SPEL+ program. The VGet command is used to read results and properties. One of the most common results to use for robot guidance is the RobotXYU result. For example:

```
Function getPart
  #define PICKZ -100.0
  Boolean found
  Real x, y, u

  VRun findPart
  VGet findPart.corr01.RobotXYU, found, x, y, u
  If found Then
    pick = XY(x, y, PICKZ, 0)
    Jump pick
  EndIf
Fend
```

Notice in the example above that the found status is checked before moving the robot to the position found by the vision system. It is important to verify that a vision object was found before using its results to move the robot.

11.7 Accessing Multiple Results from the SPEL⁺ Language

Some Vision Guide objects such as the Correlation and Blob objects have the ability to find multiple features with one object by using the NumberToFind property. The CurrentResult property is used to set which result will be displayed in the Results list if we examine the results in the [Object] tab. It is also used to determine which result record to return results for. For example, if we want to get the Area result from the 3rd result returned from a Blob object, CurrentResult must be set to 3. You have already seen how this can be done from the [Object] tab Properties list. Now let's take a look at how to access multiple results from SPEL⁺.

Multiple result access from SPEL⁺ treats the result as an array where the result number is the subscript number next to the result to be obtained. The first example below shows how to get the 3rd Area result and put it in a variable called area from the SPEL+ Language.

```
VGet seqname.objname.Area(3), area
```

The 2nd example below shows how to get that same 3rd Area result but this time assign it as the value of the 3rd element in an array called *area*.

```
VGet seqname.objname.Area(3), area(3)
```

Variable names can also be used to represent the element in an array rather than fixed elements as in example #2 above. Notice that the variable called var is used as the subscript for the Area result.

```
VGet seqname.objname.Area(var), area(var)
```

The 4th example assumes that you have used a single vision object to find multiple like parts (let's say up to 10). You would now like to pick these parts (let's say they are pens)

with a robot so you will need to store the X, Y, and U coordinates in array variables for the coordinate values of each of the parts found. The code below retrieves these coordinates using the RobotXYU result and puts them into X, Y and U arrays that could later be used for the robot to move to.

```
Function test
  Boolean found(10)
  Integer numFound, i
  Real x(10), y(10), u(10)

  Jump camshot     'move camera into position snap shot

  VRun seq01       'run the vision sequence to find the pens

  VGet seq01.blob01.NumberFound, numFound  'how many found

  For i = 1 to numFound 'get robot coords
    VGet seq01.blob01.RobotXYU(i), found(i), x(i), y(i), z(i)
  Next i

  ' Add code for robot motion here……
Fend
```

11.8 Using Vision Commands with Multitasking

You can execute vision sequences and retrieve results in more than one task in SPEL[+]. SPEL[+] automatically handles executing one vision command at a time internally. When multiple tasks require the vision system, the commands are executed on a first come first serve basis. While one task is using the vision system, the other tasks will block (wait) for the first task to finish the current vision command.

In some cases, you may be running the same sequence from different tasks using different cameras. You will want to run the sequence and get the results in one task while other tasks wait. Use the SyncLock and SyncUnlock commands to lock the vision system for one task, then release it for others.

```
Function VisionTask(camera As Integer)
  Boolean found
  Do
    SyncLock 1 ' Lock vision just for this task
    VSet FindParts.Camera, camera
    VRun FindParts
    VGet FindParts.AllFound, found
    SyncUnlock 1 ' Unlock vision for other tasks
  Loop
Fend
```

11.9 Using Vision with the Robot

Before you can use results from a vision sequence to guide the robot, you need to calibrate the camera that is used by the sequence. Refer to *7.Vision Calibration* for details.

If you attempt to get a result from a sequence object that requires calibration and there is no calibration, then a run time error will occur.

11.9.1 Position Results

All position results reported by objects in a vision sequence that uses a robot-calibrated camera are in the specified local coordinate system. Unlike other robot/vision systems, there is no extra step to do a pixel coordinate to robot coordinate transformation function call. This is completely handled by Vision Guide 7.0. The X, Y, and U coordinates of the part in the robot local coordinate system can easily be retrieved.

To accurately position an end effector at a position determined by the vision system, you need to define a tool for the end effector. See *Defining a Tool* in the next section.

The following results can be used for guiding the robot:

RobotXYU	Returns x, y, and u along with found status.
RobotToolXYU	Upward camera only
	Returns X, Y, and U coordinate values to define a robot tool with found status
RobotX	Returns the X coordinate.
RobotY	Returns the Y coordinate.
RobotU	Returns the U coordinate.
RobotPlacePos	Upward camera only
	Returns a robot position in Point in order to place the detected part to the work place position registered to CalRobotPlacePos.
	Note:
	CalRobotPlacePos needs to be set for each object.

The U coordinate zero degree position is along the Y-axis pointing straight out from the robot base.

11.9.2 Defining a Tool

It is important to define tools for the robot end effectors. This tells the robot where the end effector is, so that all position information is with respect to the end effector, not the TOOL 0 position. Use the TLSET command to define a tool in SPEL$^+$. The following are three methods for defining tool offsets.

Define a tool with the Robot Manager Tool Wizard

You can use the Tool Wizard in the Robot Manager to define a tool.

To use the Tool Wizard, perform the following steps:

(1) Open the Robot Manager.

(2) Click the [Tools] tab.

(3) Click the <Tool Wizard> button.

(4) Follow the steps in the wizard to create the tool.

Using an upward looking camera to calculate tool offsets

Here is an example of how to use an upward looking camera to calculate tool offsets. It uses the RobotToolXYU result to determine the tool offsets. The function first runs a sequence to locate the tip of the tool. Then the tool offsets are retrieved using VGet RobotToolXYU, and the tool is defined using TLSet.

```
Function DefineTool
  Boolean found
  Real xTool, yTool, uTool

  VRun findTip
  VGet findTip.tip.RobotToolXYU, found, xTool, yTool, uTool
  If found Then
    TlSet 1, XY(xTool, yTool, 0, 0)
  EndIf
Fend
```

Manually calculating tool offsets

In the following steps, you must Jog the robot with the Jog buttons from the Jog & Teach window. You cannot free the axes (using SFREE), and move the robot manually when calculating the tool offsets as shown below.

Perform the following steps to calculate tool offsets.

NOTE

☞
- When using the vertical 6-axis robot, execute the following Local command and then follow the steps on the next page.
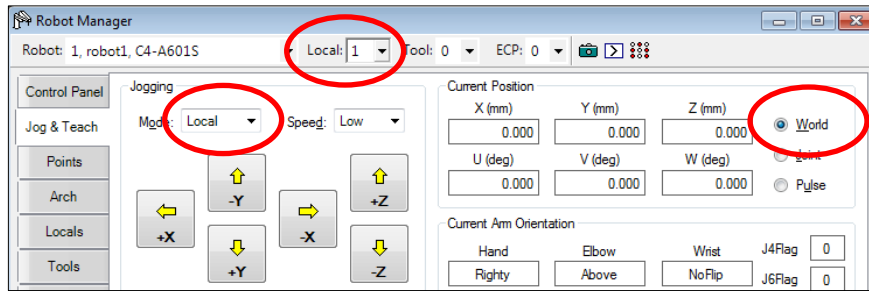
    ```
    > Local 1, Here
    ```

- When acquiring the current position in the calculation steps below, the position in Local 1 should be used.
  If using the Jog & Teach window, set the jog mode to "Local", display mode for the current position to "World", and the Local number to "1".
  If using the command window, the position of Local 1 can be checked by executing the command as follows.

    ```
    > Print Here@1
    ```

(1) Position the U axis at zero degrees.

(2) Set tool to zero (TOOL 0).

(3) Jog the end effector over a reference point and align it carefully. Do not change the U axis setting.

(4) Write down the current X and Y coordinates as X1 and Y1.

(5) Jog the U axis to the 180-degree position.

(6) Jog the end effector over the reference point and align it carefully. Do not change the U axis setting.

(7) Write down the current X and Y coordinates as X2 and Y2.

(8) Use the following formula to calculate tool offsets:

```
xTool = (X2 - X1) / 2
yTool = (Y2 - Y1) / 2
```

(9) Define the tool from the Tools page on the Robot Manager or from the Command window.

```
TLSET 1, XY(xTool, yTool, 0, 0)
```

(10) Test the tool settings:

Set the current tool to the tool you defined in the previous step. For example, TOOL 1.

Jog the end effector over the reference point.

Now jog the U axis.

The end effector should remain over the reference point.

11.9.3 Tool Calculation for Circuit Board Held by the Robot

In this example, Vision Guide 7.0 calculates a tool for a circuit board that is being held by the robot. This requires one upward looking camera. You will need to teach the place position for the board one time after calibrating the camera.

To teach the place position:

(1) Pick up the circuit board with the robot.

(2) Call the example function CalcBoardTool to calculate tool 1.

(3) Switch to Tool 1.

(4) Jog the robot to the place position of the board.

(5) Teach the place position.

Holding position

Circuit board

TOOL offset to be calculated
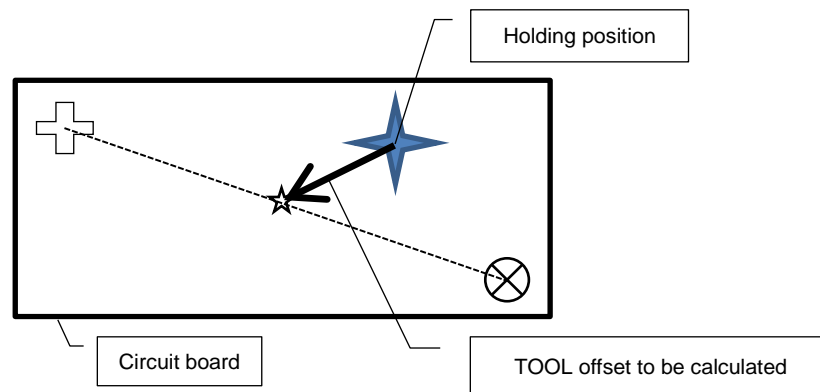
```
Function CalcBoardTool As Boolean
  Boolean found
  Real x, y, theta
  Real toolX1, toolY1, toolU
  Real toolX2, toolY2

  CalcBoardTool = False
  Jump Fid1CamPos  ' Locate fiducial 1 over camera
  VRun SearchFid1
  VGet SearchFid1.Corr01.RobotToolXYU, found, toolX1, toolY1, toolU
  If Not found Then
    Exit Function
  EndIf

  Jump Fid2CamPos  ' Locate fiducial 2 over camera
  VRun SearchFid2
  VGet SearchFid2.Corr01.RobotToolXYU, found, toolX2, toolY2, toolU
  If Not found Then
    Exit Function
  EndIf
  x = (toolX1 + toolX2) / 2
  y = (toolY1 + toolY2) / 2
  theta = Atan2(toolX1 - toolX2, toolY1 - toolY2)
  toolU = RadToDeg(theta)
  TlSet 1, XY(x, y, 0, toolU)
  CalcBoardTool = True
Fend
```

11.9.4 Positioning a Camera for a Pallet Search

When using a J2 mount camera for pallet search, the camera imaging position is hard to control. However, the camera moves easily to each imaging position on the pallet by defining an Arm for the camera using the Arm command. For details on the Arm command, refer to *EPSON RC+ 7.0 SPEL⁺ Language Reference*.

Tips:
The Arm command is only available for the SCARA robots.