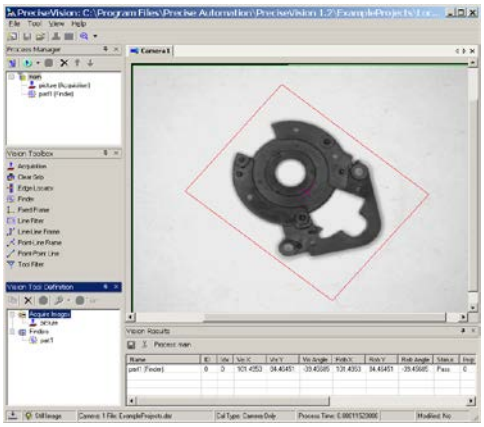


PreciseVision Machine Vision System

Introduction and Reference Manual

P/N: PVS0-DI-S0010, Rev 5.0.0, April 9, 2022

Brooks Automation

Information provided within this document is subject to change without notice, and although believed to be accurate, Brooks Automation assumes no responsibility for any errors, omissions, or inaccuracies.

This technology is subject to United States export Administration Regulations and authorized to the destination only; diversion contrary to U.S. law is prohibited.

Brooks Automation
Precise Collaborative Robotics
201 Lindbergh Avenue
Livermore, CA 94551
Tel: +1-510-498-1130

Brooks Automation
15 Elizabeth Drive
Chelmsford, MA
01824-2400
Tel: +1 978-262-2400
Fax: +1 978-262-2500

Brooks Automation (Germany) GmbH
Daimler-Straβe 7
78256 Steiβlingen,
Germany
Tel: +49-7732-9409-0
Fax: +49-7732-9409-200

Brooks Automation
46702 Bayside Pkwy
Fremont, CA 94538
Tel: +1-510-661-5000
Fax: +1-510-661-5166

Corporate Headquarters
15 Elizabeth Drive
Chelmsford, MA 01824 U.S.A.

Brooks Automation
Precise Collaborative Robotics
201 Lindbergh Avenue
Livermore, CA 94551 U.S.A

For Technical Support:

Location	Contact Number	Website
North America	+1-510-498-1130 (Precise) +1-800-447-5007 (Toll Free) +1-978-262-2900 (Local)	http://www.preciseautomation.com
Europe	+49 800 000 9347 (Toll Free Germany) +49 364 176 9999 6 (Has Toll)	
Japan	+81 120-255-390 (Toll Free) +81 45-330-9005 (Local)	
China	+86 21-5131-7066	
Taiwan	+886 080-003-5556 (Toll Free) +886 3-5525258 (Local)	
Korea	1800-5116 (Toll Free)	
Singapore	+65 1-800-4-276657 (Toll Free) +65 6309 0701 (Local)	

Revision History

Revision	ECO Number	Date	Explanation of Changes
Rev 5.0.0	TBD	April 9, 2022	First version as Brooks. All references to Dalsa cameras have been deleted since support for these cameras was eliminated some time ago.

Warning Labels

The following warning and caution labels are utilized throughout this manual to convey critical information required for the safe and proper operation of the hardware and software. It is extremely important that all such labels are carefully read and complied with in full to prevent personal injury and damage to the equipment.

There are four levels of special alert notation used in this manual. In descending order of importance, they are:

DANGER: This indicates an imminently hazardous situation, which, if not avoided, will result in death or serious injury.

WARNING: This indicates a potentially hazardous situation, which, if not avoided, could result in serious injury or major damage to the equipment.

CAUTION: This indicates a situation, which, if not avoided, could result in minor injury or damage to the equipment.

NOTE: This provides supplementary information, emphasizes a point or procedure, or gives a tip for easier operation

Table Of Contents

PreciseVision Machine Vision System

PreciseVision Introduction

PreciseVision is a powerful, full featured 2D machine vision software package. This software application includes a complete toolkit with everything that is needed to acquire vision images, identify and locate randomly positioned and oriented parts, measure key features and inspect parts for specific defects. Each of the basic vision functions is provided as a type of "Tool" in the vision toolbox and is internally designed as a modern software "Object".

PreciseVision executes on a Windows PC system and includes a built-in interface to communicate with a Precise Controller via Ethernet. This software package has been designed to operate with the Guidance Programming Language, which executes on a Precise Guidance Controller, to produce flexible vision-guided motion applications. The Guidance Programming Language, GPL, includes a complementary built-in communications interface to PreciseVision and has classes and objects for easily controlling the operation of PreciseVision and retrieving vision results. A complete vision guided robot application consists of a list of vision tools executing in PreciseVision and a GPL program executing on a Guidance Controller that takes the vision results and utilizes this information to alter the actions of a robot.

PreciseVision includes a very easy-to-use visual programming interface that allows both simple and complex vision programs to be constructed by dragging-and-dropping tools into a list. Training of the tools is performed by visually positioning and sizing tools on top of a captured camera image. The parameters for each tool can also be access via a "Properties Table". Since each tool is a software object, the output of one tool can be easily linked to the input of other tools. This is a very powerful capability that allows the placement and the actions of tools to be a function of the results of previously executed tools and permits the results of several tools to be combined.

To capture an image, PreciseVision supports third party Ethernet and USB cameras. Vendor and part number information for purchasing and installing recommended hardware and drivers are provided in this document.

While the most common configuration is to have PreciseVision interfaced to a single robot, information is also provided on executing PreciseVision without a robot or when connected to multiple robots.

In addition to the normal mode of executing tools on a real camera image, this package permits a sequence of images to be easily saved to disk and then subsequently played back. This allows vision tools to be tested off-line and provides a convenient means for customers to obtain assistance from Precise and its partner companies remotely.

In this document, instructions for installing PreciseVision and the supported 3rd party hardware is presented. This is followed by an overview and quick start section that introduces machine vision newcomers to PreciseVision's principal concepts and permits experienced individuals to quickly use the

system. Finally, the last sections of this manual provide detailed information on the system's user interface, toolkit and camera calibration procedures.

For further information on programming a vision guided robot application in GPL, please refer to the *Guidance Programming Language, Introduction to GPL*.

Supported Hardware and Software Installation

Supported Vision Hardware

PreciseVision is a software package and does not include hardware. This software is designed to capture vision images that are generated by either Ethernet or USB cameras. PreciseVision supports two camera options:

1. IDS Imaging uEye™ cameras that communicate via a USB 2.0/3.0 connection or Ethernet connection
2. Other 3rd party USB cameras (***No customer support provided***)

Each of these options offers a range of camera resolutions, imaging element sizes and a choice of monochrome or color image capturing. Most of the IDS industrial cameras are fully supported by PreciseVision. In addition, there are a host of other IDS USB cameras available that range from very inexpensive web cams to astronomy cameras, all with different prices, characteristics and installation procedures. Please contact Precise to determine if a specific camera has been qualified with PreciseVision. ***Precise is not able to provide support for USB and Ethernet cameras that have not been tested and qualified.***

When using the standard version of PreciseVision, the system can connect to multiple USB or Ethernet cameras. However, you cannot connect to both Ethernet and USB cameras simultaneously. In the Multi-Instance version of PreciseVision, the same rule applies to each instance, i.e. a single instance cannot be connected to a mix of Ethernet and USB cameras. However, different instances can be connected to either Ethernet or USB cameras.

The following table provides some general information concerning IDS USB cameras and other 3rd party cameras.

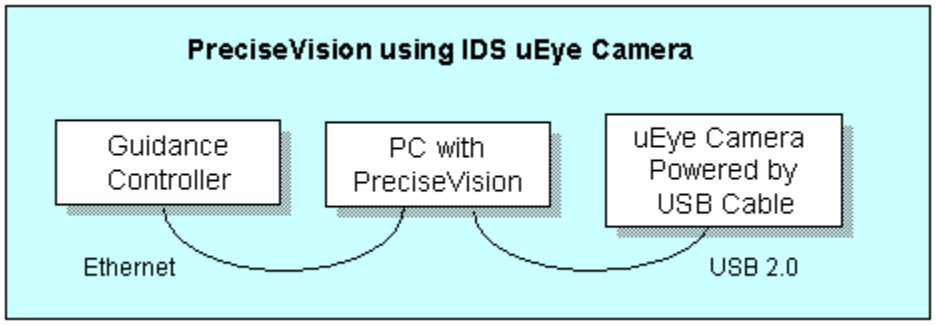
Characteristic	General Comments
Cost	IDS industrial cameras are very reasonably priced for industrial units. However, non-industrial 3rd party USB cameras can be significantly less expensive.
Resolutions	640x480 up to 2560x1920 with even higher resolutions to be announced soon.
Acquisition time	USB 2.0 operate at 480 Mbits/sec and USB 3.0 can operate at 4.8Gbits/sec for reduced requisition times.
Hardware synchronization	Supports trigger input and acquisition output

PC CPU Loading	USB cameras generate a higher load as compared to Ethernet cameras. Multiple cameras will require a higher performance CPU. 5 or more cameras may degrade acquisition time.
Cable length	5M for USB. Powered USB hubs might be able to extend length
Max. number of cameras per PC	Limited by how many cameras can be connected to and serviced by a PC using hubs, maximum of 6 cameras per PV instance. Adding cameras to a port will reduce the pixel clock rate.
Installation and support	USB is typically easier than Ethernet for single cameras if wired point to point since no firewalls, less issues with USB hardware interface, simpler driver installation. Multiple IDS USB cameras are a bit more difficult to setup since each must be assigned an ID and registered with the DirectShow interface.
Arm-mounted cameras	Requires extra external cable for Precise robots
Power	5VDC taken from single USB cable
Size / weight	34x32x34.4 mm, 75g

In the following sections, part numbers are provided to allow hardware to be directly purchased from the hardware vendors together with simplified wiring drawings that illustrate how the components should be connected.

Option 1: IDS uEye USB 2.0 Cameras

- Each instance of PreciseVision supports up to 6 IDS uEye™ USB 2.0 cameras. See the table below for the recommended/supported camera models.
- IDS cameras are typically connected directly to the PC running PreciseVision although a USB hub can be used.
- The PC must support USB 2.0 (not 1.0) and must have DirectX 9 or higher installed. PreciseVision uses DirectShow (DirectX) to interface to the camera driver.
- It is highly recommended that only uEye cameras that support "Global" shutters be utilized and not "Rolling" shutters. A Global shutter exposes all of the pixels of the imaging element at one time. A Rolling shutter exposes lines at different times. This can result in smearing if there is any movement in the image.
- For flexibility in selecting the appropriate lens for your application, a camera with a C-Mount or CS-Mount is highly desirable.



Hardware Specification	
USB CAMERAS	PreciseVision fully supports the IDS Imaging Development Systems GmbH uEye USB 2.0 Cameras listed below. All of these models have Global Shutters, metal cases, C-mount lens adapters, are available in monochrome (M) or color (C), DB9 signal/power connector, and support an external trigger (DIN) and a digital output (DOUT). Effective frame-to-frame time depends on the exposure time and PC overhead. For total time, add vision processing time, which is typically less than 100ms. PreciseVision supports up to 6 USB cameras. UI-2410-M/C, 640x480 resolution, 75 frames/second raw frame rate, 1/3" CCD sensor. UI-2230SE-M/C, 1024x768 resolution, 30 frames/second raw frame rate, 1/3" CCD sensor. UI-2240SE-M/C, 1280x1024 resolution, 15 frames/second raw frame rate, 1/2" CCD sensor. UI-2250-M/C, 1600x1200 resolution, 12 frames/second raw frame rate, 1/1.8" CCD sensor.
CAMERA POWER	Power is supplied directly by the USB cable and is generated by the PC's power supply.
USB INTERFACE	PC must be equipped with USB 2.0 interface (not 1.0).
USB CABLE, TRIGGER SUPPORT, NO DOUT	IDS uEye industrial camera cable: High speed 2.0 Extension cable - Screw type 4-Pin USB type A (F) TO 5-Pin micro-USB type B (M). 3 meter length. Cable breaks out trigger input as separate input. CB-IDS-9UD-A-T-3M

USB CABLE, TRIGGER AND DOUT SUPPORT	Right angled trigger cable for uEYE-Cameras, 3 meter uEye Micro Sub-D connector, 90° angled with screws. USB signal via shielded USB 2.0 cable to USB-A. Breakout cable for trigger input and digital output, cable with open ends. CB-IDS-9UD-A-ST3M-RT

Option 2: Other 3rd Party USB 2.0 Cameras

There are a wide variety of 3rd party USB 2.0 cameras that are available for the home, hobbyist, commercial and industrial markets. Due to the large number of vendors, their different installation procedures and their individual idiosyncrasies, Precise cannot guarantee that any camera brand that you select will work properly and ***we cannot provide assistance for any camera that has not been tested and qualified by Precise.***

Nonetheless, if you wish to interface an unsupported camera without Precise's assistance, please consider the following when selecting a USB camera.

- PreciseVision utilizes DirectShow to interface to the camera. Therefore, the USB camera must be compatible with DirectShow, which is quite common.
- PreciseVison's image acquisition tool has been specially optimized for the IDS uEye family of cameras. Other USB cameras will take somewhat more time to acquire an image.
- For similar features, a camera in a plastic case will be less expensive that a camera in a metal case. You must determine if the additional ruggedness of a metal case is important for your environment.
- Many inexpensive cameras utilize the standard USB connectors that are retained by friction. It is relatively easy for these to become accidentally disconnected as compared to screw in DB connectors such as those used in the IDS uEye cameras.
- Many inexpensive cameras either have integrated fixed lens or lens mounts that do not support the full range of lens that are available. Also, many lens do not have screws for locking in the focus or F-stop adjustments.
- Most cameras do not have trigger inputs and digital outputs. These are important features for conveyor belt tracking and other real-time image capture requirements. Also, since communication standards have not be established for these features, even if the camera has these hardware features, PreciseVision will probably not be able to access this capability.

Installing PreciseVision on a PC

PreciseVision is distributed as a standalone application that executes on a Windows PC system. For the best results, it is suggested that this software be executed on the following equipment:

- 2.3Ghz or faster Pentium 4 PC running a Windows XP or later system
- Depending upon the type of cameras to be utilized:
  - Gigabit Ethernet (recommended and required for certain cameras), otherwise 100Mb Ethernet interface
  - USB 2.0 interface

- 512 MB of RAM
- At least 1 GB of space on the PC's disk
- A 2x CD-ROM drive interfaced to the PC or access to the Precise Support website

While PreciseVision can be executed by itself to develop Vision Processes, the PC must be able to communicate with a Precise Guidance Controller via Ethernet in order to run a complete vision-guided motion application. **Please see the *"Guidance System Setup and Operation, Quick Start Guide"* for instructions for configuring the PC and the Guidance Controller to communicate with each other.**

If you previously installed PreciseVision on your computer, you should un-install the old version by performing the steps below. If you are installing PreciseVision for the first time, you can skip the first set of instructions.

» Shut down all programs that are running including virus protection programs.

» Bring up the Window's Control Panel by clicking "***Start > Settings > Control Panel***".

» Double click on the "***Add or Remove Programs***" selection.

» In the "Add or Remove Programs" popup window, scroll down and click on "***PreciseVision***".

» Click the "***Remove***" button and click on "***Yes***" to confirm the action.

» Close all of the windows that you opened.

To install PreciseVision on your computer, perform the following steps:

» Shut down all programs that are running including virus protection programs.

» Insert the PreciseVision CD-ROM into your computer's CD-ROM drive. A panel should pop up that welcomes you to the Setup Wizard. If the installer does not automatically start, click "***Start > Run***", type in "***D:\setup.msi***" (where D is the CD-ROM drive), and click "***Ok***".

» Follow the instructions in the Setup Wizard to install PreciseVision. Note, PreciseVision relies upon the Microsoft .NET Framework in order to operate. This is a standard module that Microsoft provides free of charge. If the Setup Wizard detects that this module is not available, you will be asked if you wish to install the Framework now. You should respond "***Yes***". This will launch a browser to take you to the download site for the required software.

At the conclusion of this process, PreciseVision will be installed on your PC. This software by itself is sufficient to execute PreciseVision with camera images that have already been captured and saved to a

disk file. But, to operate this software with a camera, **one of the camera interface installation procedures in the following sections must be performed**. In addition, in order to continue to use PreciseVision beyond the trial period, **you must register this product with Precise.** The registration process is described in detailed in a following section.

Installing and Configuring the IDS uEye™ USB Camera Interface

If your system is equipped with an IDS uEye™ USB camera, the IDS uEye device driver must be installed and the camera must be configured. PreciseVision utilizes DirectShow and the IDS driver to communicate with the camera. If your system does not use this type of camera, please skip to the next section of the documentation.

Prior to executing this procedure, ensure that the following requirements are satisfied:

- Initially, it is preferable if your IDS camera is NOT connected to the PC. After the IDS driver is installed, the camera can be attached and the PC will automatically discover the camera.

The first step in the process is to install the IDS USB device driver and its associated configuration and diagnostic software.

» Insert the PreciseVision CD-ROM into your computer's CD-ROM drive.

» In the ***"uEye"*** folder, double click ***"uEye32_xxxxx.exe"*** to execute the installation procedure. If your computer contains a 64bit version of Windows, then visit the IDS web site to download the 64bit version of the driver.

» Follow the instructions in the Setup Wizard to ***"Install driver"***. When asked what type of installation should be performed, select ***"USB"***. You should **NOT** install any Ethernet or custom drivers.

At this point, the required system driver and the support software have been loaded and installed.

Assigning ID's to Multiple uEye Cameras

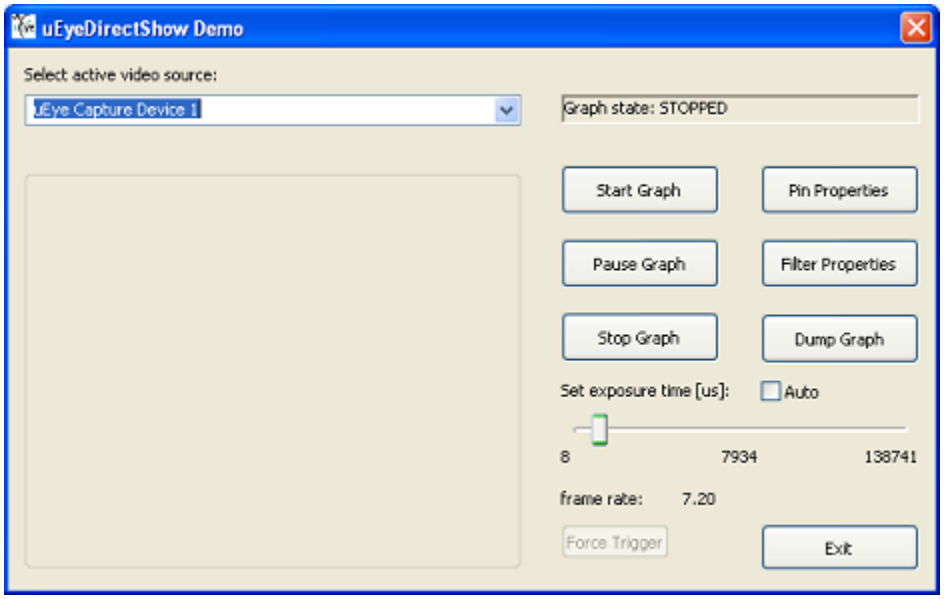
For Ethernet cameras, each camera's Ethernet MAC ID uniquely identifies the camera and is used to associate each camera with the same vision operations whenever PreciseVision is restarted. Unfortunately, USB connections do not have a similar identification mechanism. So, to ensure that the correct cameras are associated with vision operations, a unique camera ID must be stored into the NVRAM of each IDS camera.

The camera ID ranges from 1 to n and is automatically added to the name of each camera. For example, if the ID is set to 3, the camera will be identified as "uEye Capture Device 3".

» Plug the uEye cameras into the PC. This will permit the PC to discover this new hardware and configure the cameras for use.

» Execute the uEye demo software to verify that the cameras are functioning property. Select the demo software by ***RIGHT*** clicking: ***"Start > Programs > IDS > uEye > Samples > uEye DirectShow Demo"***.

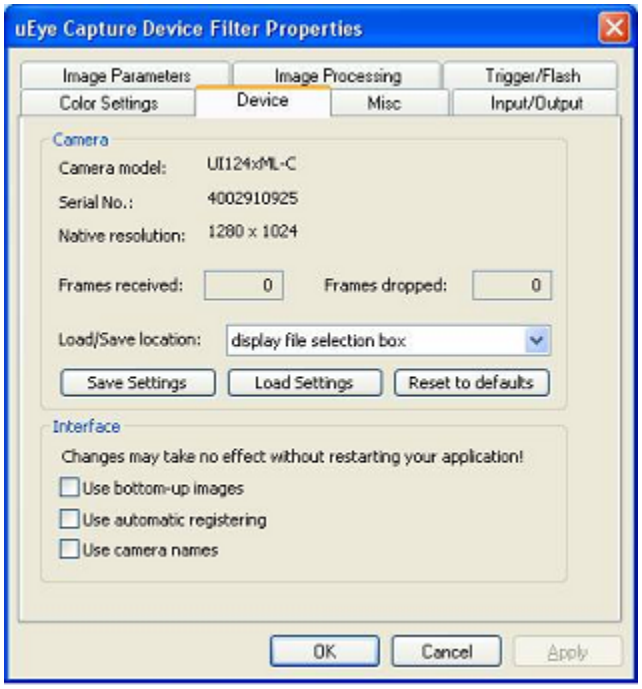
» In order to gain access to key properties when executing the demo software, click "***Run as***", then select "***The following user:***" and "***Administrator***".



In order to receive a clear indication that the camera is properly interfaced and to permit changing the camera ID's, the automatic registration mode of the camera must be disabled.

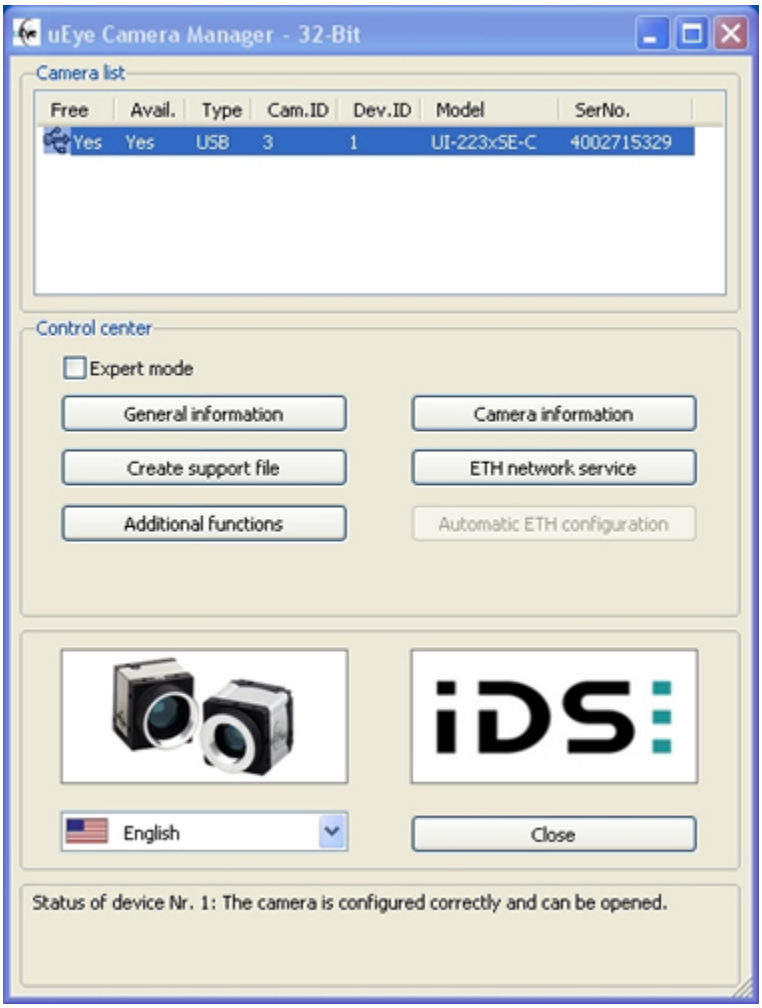
» In the main panel for the demo program, set the "***active video source***" to the camera of interest and press the "***Filter Properties***" button to display the panel below.

» Uncheck "***Use automatic registering***", press "***Ok***", and exit the demo program.

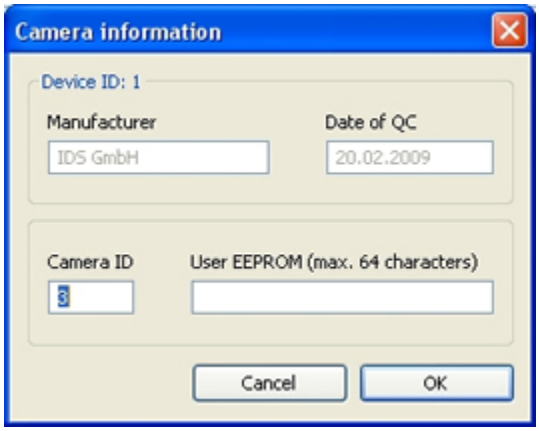


If you only intend to connect a single USB camera to your PC, you can skip to the next section since the cameras default ID of 1 can be used.

» To assign camera ID's, execute the uEye Camera Manager by clicking on ***"Start > Programs > IDS > uEye > uEye Camera Manager"***.



» Select a desired camera by clicking on a line in the Camera List and click ***"Camera Information"***.

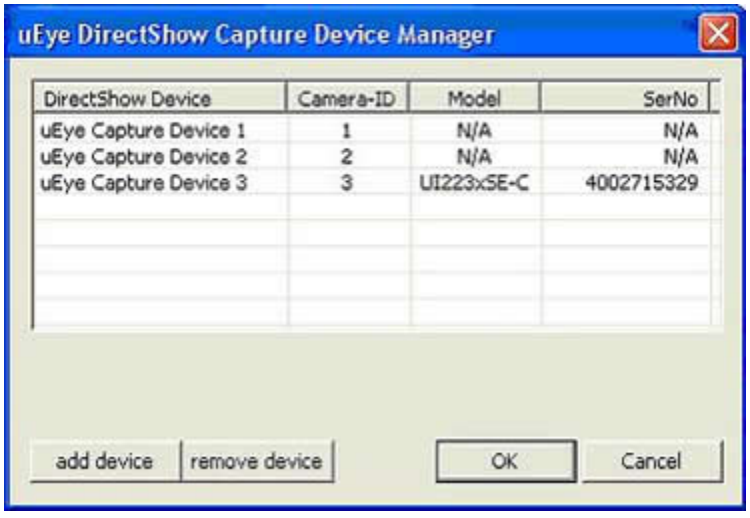


» Change the "Camera ID" to a unique number. These should be assigned so there are no gaps in the numbering (1,2,3,..).

» When completed, click ***"OK"*** and close the Camera Manager when done.

Before PreciseVision can discover multiple USB cameras, they must be registered using the uEye DirectShow Capture Device Manager.

» To register each of the IDS cameras, click ***"Start > Programs > IDS > uEye > DirectShow device manager"***.



» Clear the list by clicking on ***"remove device"*** multiple times.

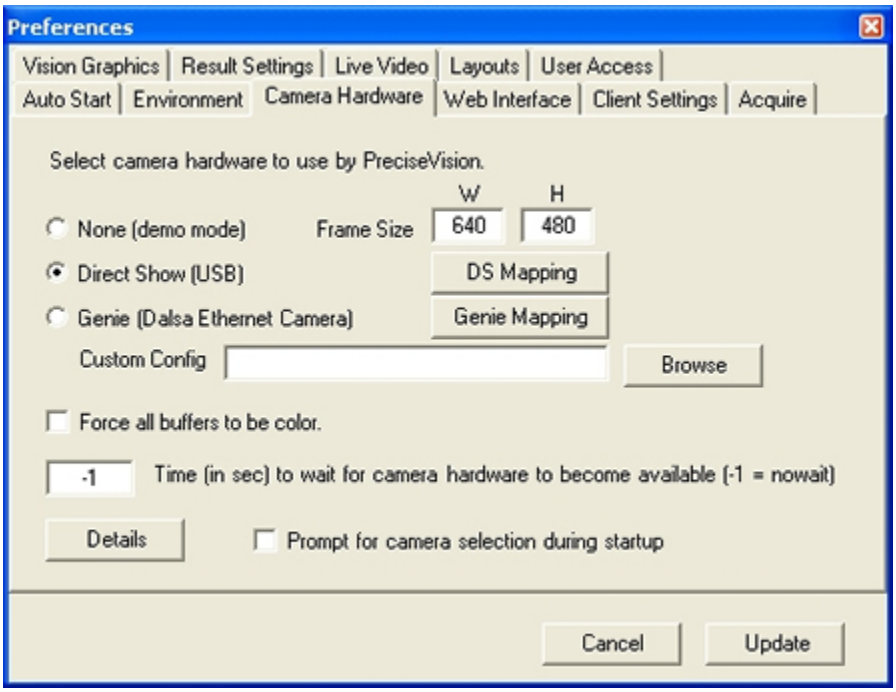
» Populate the list by clicking on ***"add device"*** a number of times until the highest camera ID that you have defined is listed. If the camera ID's were not sequentially assigned, there can be N/A placeholders as shown above. This will not create a problem.

Selecting USB Cameras Instead of Ethernet Cameras

PreciseVision can operate using either Ethernet cameras or USB cameras or in a demonstration mode without a camera. However, at any given time, only one camera type is permitted for each instance of PreciseVision. The first time PreciseVision is executed after being installed, it prompts for the type of camera to be used. Thereafter, the camera type is remembered until it is explicitly changed. The following procedure can be used to change the camera type or to verify that PreciseVision is properly configured to capture images from the correct type of camera.

» Launch PreciseVision by selecting ***"Start > Programs > Precise Automation > PV_ > PreciseVision"***.

» Open the Preferences pop-up by selecting the following within PV: ***"File > Preferences"*** and open the ***"Camera Hardware"*** tab.



» Ensure that the ***"DirectShow (USB)"*** option is selected.

» Restart PreciseVision if the selected camera type is changed.

Assigning Cameras, Selecting the Camera Resolution and Display Mode

When the PC boots or whenever a USB camera is connected or disconnected, Windows automatically assigns each camera an index in a list of cameras. The ordering of the cameras is dependent upon the USB interface to which they are connected. For a given configuration of cameras, the index of each camera will always be the same. However, if a new camera is connected, it will not necessarily be added to the start or the end of the list.

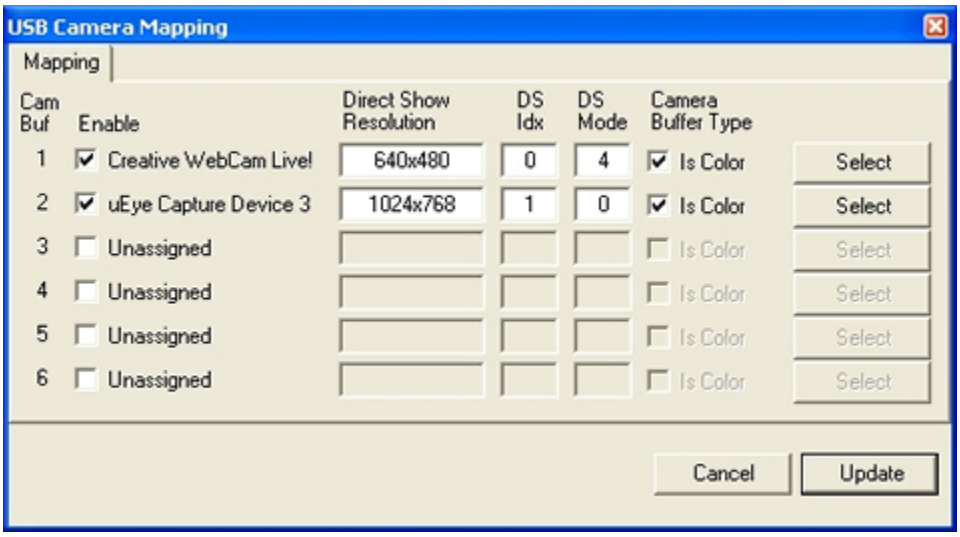
Unlike Ethernet cameras, USB cameras typically have several different formats that can be used to retrieve the image data at different resolutions. The formats that PreciseVision and its DirectShow interface support include the following:

RGB8 (8-Bit Grayscale)
RGB24 (24-bit color with 8-bit R,G,B components)
YUY2 (16-bit color)

When PreciseVision is started, it retrieves information on all of the connected USB cameras and automatically assigns them to a logical camera buffer and selects a format with the highest resolution. PreciseVision permits these automatic assignments to be manually over-written.

To review or change the current camera assignments and settings, perform the following.

» While still displaying the PreciseVision ***"Preferences"*** pop-up, and viewing the ***"Camera Hardware"*** tab (as shown above), click on the ***"DS Mapping"*** button.

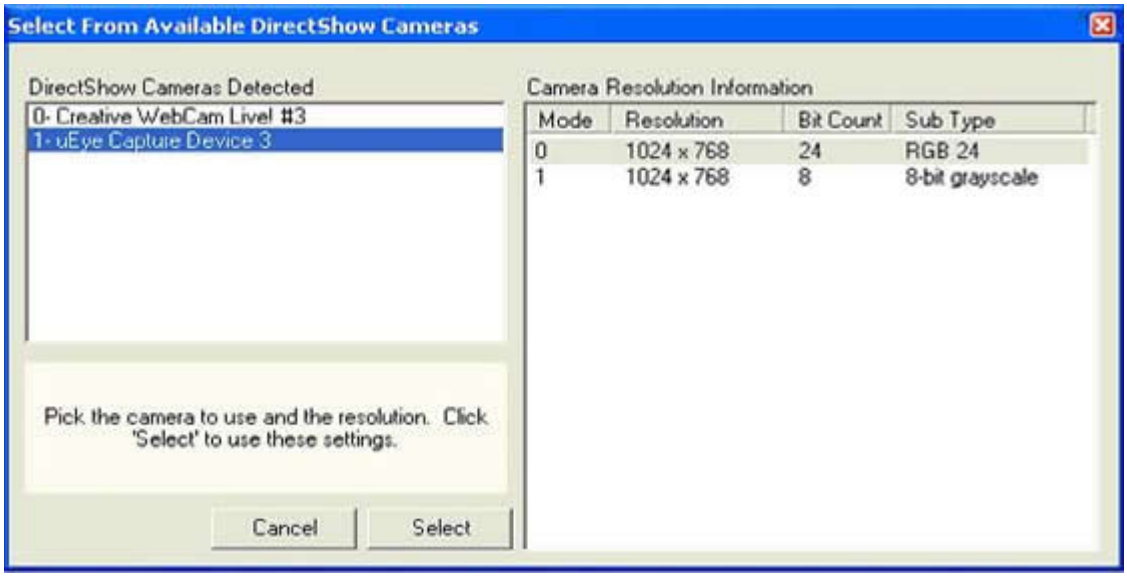


For each discovered USB camera, this panel displays the logical number (Camera buffer) that it has been assigned, together with its resolution and format, and the index number that Windows has automatically assigned. The "Enable" checkmark indicates that a camera will be exclusively used by the current instance of PreciseVision.

» If the Multiple-Instance version of PreciseVision is being executed, a camera can be released for use by another instance by unchecking ***"Enable"*** and restarting PreciseVision.

If you wish to assign a camera a different logical camera number or change its format and resolution, do the following.

» While still displaying the USB Camera Mapping panel, click on the ***"Select"*** button for the camera of interest.



If you wish to access the same camera in different formats or want to utilize more than one set calibration data, the same camera can be assigned to multiple logical camera/buffer numbers.

Subsequent Changes in Camera Assignments

If a USB camera is disconnected or a new camera is connected to the PC, the next time that PreciseVision is started, the Windows camera index numbers can all be changed. If a camera name is unambiguous, as is the case for IDS cameras that have been assigned unique camera ID numbers, PreciseVision attempts to assign the cameras to the same logical camera number/buffer as the last session independent of their Windows index number. If camera names are not unique, PreciseVision utilizes the Windows camera index numbers to determine their logical camera numbers. At the end of this process, if the camera mapping has changed relative to the previous session of PreciseVision, the following message is displayed to alert you to a possible problem.

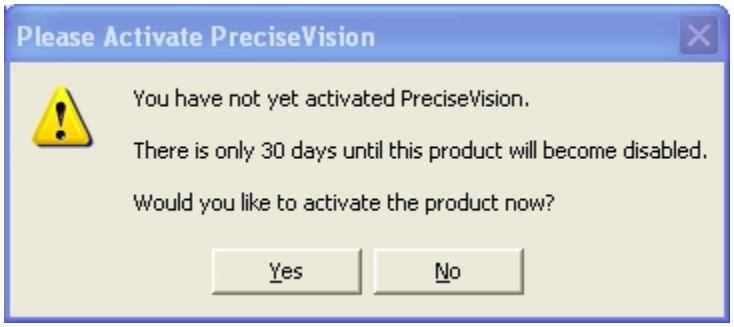
» If the error message above is displayed, review the USB camera assignment by going to ***"File > Preferences > Camera Hardware > DS Mapping"*** and correcting any errors.

This completes the IDS uEye USB configuration setup. PreciseVision can now acquire camera images via this camera(s).

Activating PreciseVision

To begin using PreciseVision, you must start the application on the PC.

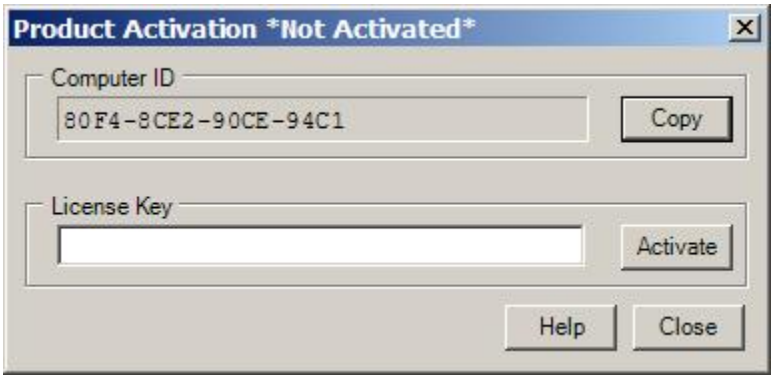
» To launch PreciseVision, click on ***"Start > Programs > Precise Automation > PV xxx > PreciseVision".*** The first time PreciseVision begins execution, you will see the following popup.



The license that you purchased for PreciseVision permits you to use this software to build and execute Vision Processes on a single PC. To complete the software activation process, you must send information concerning the PC to Precise. To allow time for sending the information and receiving a reply, PreciseVision is fully functional for 30 days without being activated.

If for some reason, you do not wish to initiate the activation process now, this popup will be displayed each time you start PreciseVision. To re-display the activation popup from within PreciseVision, click on ***"Help > Product Activation"***.

» To initiate the activation process, click ***"Yes"*** in the popup above. This will display the following activation popup.



This popup displays the "Computer ID" that you need to submit to Precise as part of the activation process.

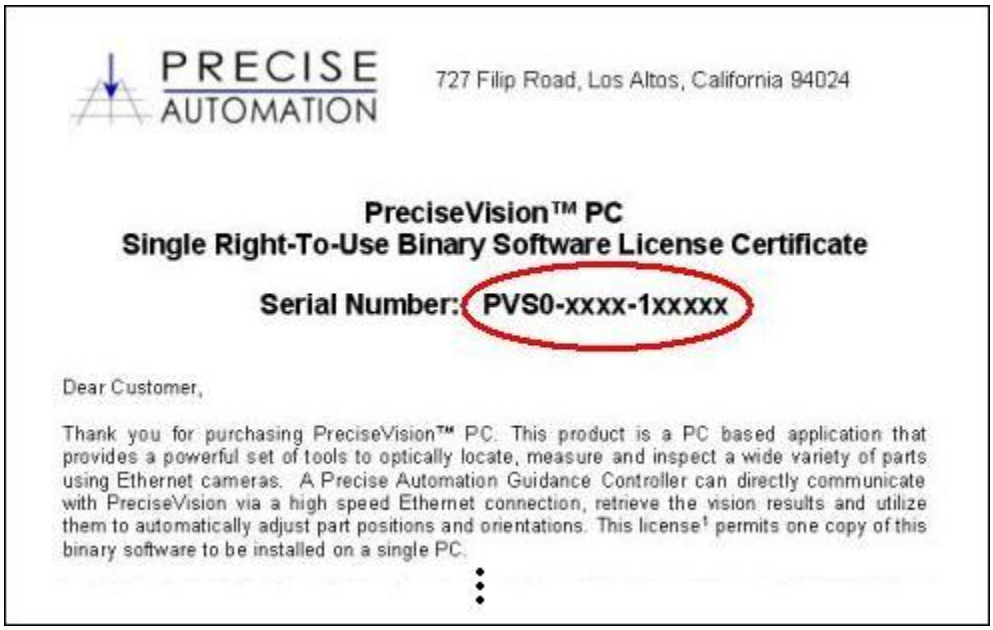
» To obtain a license key to activate PreciseVision, please send an email to Precise that contains the following information:

To: sales@preciseautomation.com
Subject: PreciseVision License Key Request

Customer Name: <your name here, optional>
Customer Company: <your company name here>
Telephone Number: <your phone number here, optional>

PreciseVision Serial Number: PVS0-0xxx-xxxxxx
Computer ID Number: xxxx-xxxx-xxxx-xxxx

» You can obtain the PreciseVision Serial Number from the PreciseVision License Certificate that you received with your order. Below is a picture of the top portion of a typical PreciseVision License with the serial number circled in red.



PRECISE AUTOMATION

727 Filip Road, Los Altos, California 94024

PreciseVision™ PC
Single Right-To-Use Binary Software License Certificate

Serial Number: PVS0-xxxx-1xxxxx

Dear Customer,

Thank you for purchasing PreciseVision™ PC. This product is a PC based application that provides a powerful set of tools to optically locate, measure and inspect a wide variety of parts using Ethernet cameras. A Precise Automation Guidance Controller can directly communicate with PreciseVision via a high speed Ethernet connection, retrieve the vision results and utilize them to automatically adjust part positions and orientations. This license[1] permits one copy of this binary software to be installed on a single PC.

» The Computer ID number should be copied from the Product Activation popup.

» As a convenience, if you are viewing this help in the *Precise Documentation Library*, you can click on the following link to get an email template: sales@preciseautomation.com.

» Alternatively, if you do not have email access, please Fax this information to 408-516-8348 together with your own fax number.

Your name and your phone number are optional. However, we recommend that you provide this information in case there is a problem with your license and we need to get in touch with you. A sample email should look as follows.

In response to your request, you will receive a license key.

» When you receive your license key, access the Product Activation popup by restarting PreciseVision or clicking in the PreciseVision menu item ***"Help > Product Activation"***.

» Enter the license key in the ***"License Key"*** box and press ***"Activate"***.

PreciseVision is now activated.

Configuring Communication Between the PC and the Controller

In order for PreciseVision to exchange information with a Guidance Controller, each system must know the Ethernet IP address of the other unit. During normal runtime execution, the Guidance Controller initiates all communications and so it must know where to send its commands. During the Robot Vision Camera Calibration procedure, PreciseVision needs to know where to send its commands to read the current robot position.

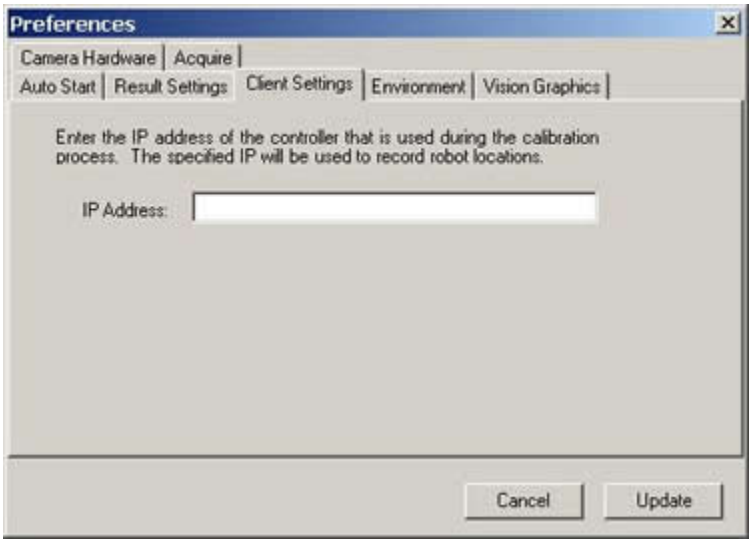
Configuring PreciseVision to Read Positions from a Controller

To permit PreciseVision to query a Guidance Controller for the current position of the robot, the IP address of the controller must be defined in the system Preferences.

» Launch PreciseVision by clicking on ***"Start > Programs > Precise Automation > PV xxx > PreciseVision".***

» Open the panel for setting the controller IP address by selecting ***"File > Preferences > Client Settings".***

The panel for setting the IP address should resemble the following:



» Enter the Guidance Controllers Ethernet IP address and press the ***"Update"*** button.

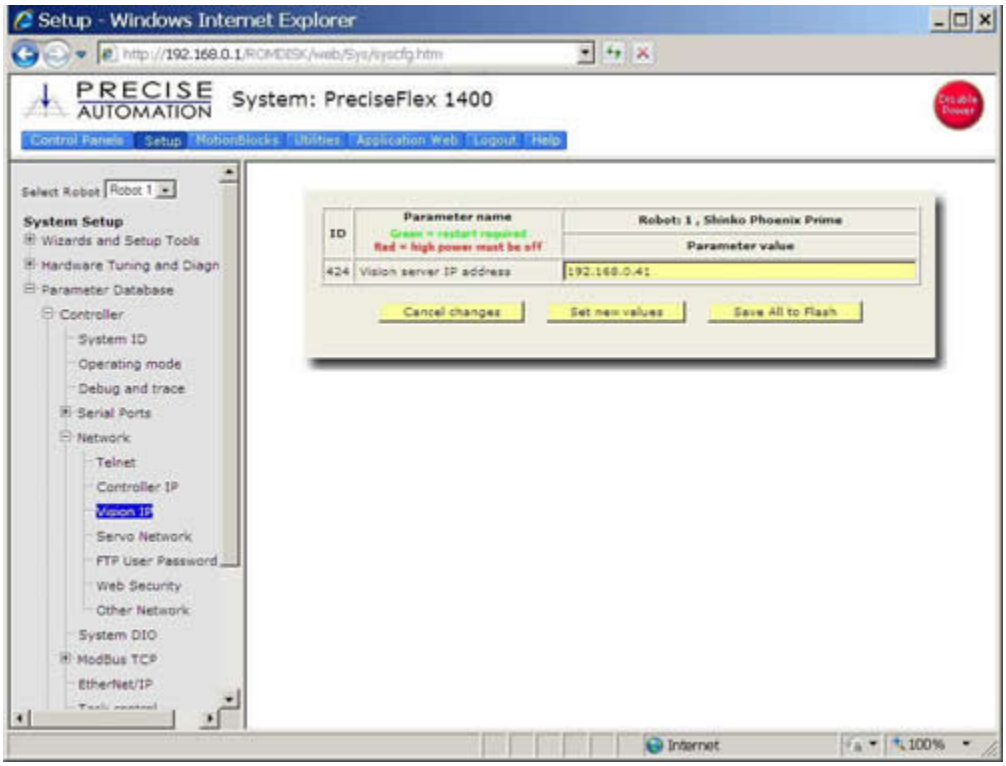
Configuring a Controller to Send Commands to PreciseVision

Whenever the Guidance Controller issues a command to PreciseVision, the controller automatically establishes an Ethernet connection utilizing an IP address that is specified in the controller's Parameter Database. This database is stored in the controller's flash disk and is preserved even if the controller is turned off and restarted. To modify this database value, perform the following operations.

» Please see the *"Guidance System Setup and Operation, Quick Start Guide"* and follow the instructions for bringing up the web based Guidance Operator Interface for the controller.

» In the web interface, open the database page for the vision IP address, ***"Setup > Parameter Database > Controller > Network > Vision IP*** ".

The web interface page should look like the following:



» Enter the IP address of the PC executing PreciseVision into the value for the ***"Vision server IP address (DataID 424)"***.

» Press the "***Set New Values***" button to store this setting into memory.

» Press the "***Save All to Flash***" button to store this setting in the flash disk. This ensures that this setting will remain in effect if the controller is restarted. The button will flash for 10-30 seconds as the data is being written. **DO NOT TURN OFF YOUR CONTROLLER WHILE THE BUTTON IS BLINKING SINCE THIS MAY CORRUPT THE FLASH DISK.**

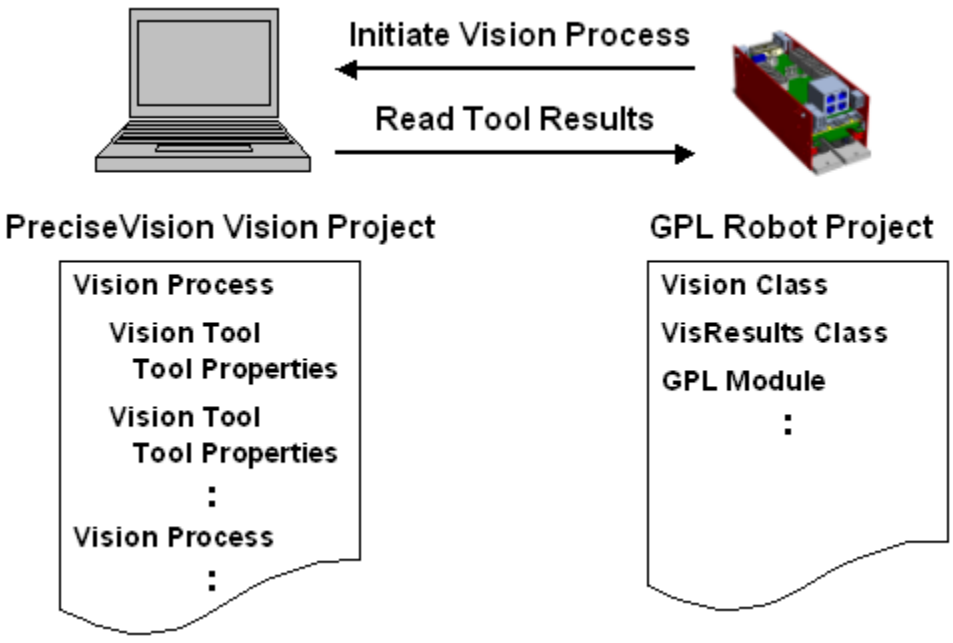
The controller is now setup to send commands to PreciseVision. It is not necessary to restart the controller.

Overview and Quick Start

PreciseVision Overview

This chapter introduces PreciseVision's principal concepts, terminology and features. For newcomers to machine vision, this chapter will provide background information prior to reading the detailed descriptions contained in the following chapters. For individuals with experience applying machine vision, the information in this chapter should be sufficient for you to start using the system.

PreciseVision has been designed to operate in combination with a Precise Guidance Controller and its embedded Guidance Programming Language (GPL) to easily produce flexible and powerful vision-guided motion applications. The following diagram illustrates the key software components in a vision guidance application.



In PreciseVision, each fundamental vision runtime function is represented as a type of **"Vision Tool"** and the collection of all possible types of tools is referred to as the **"Vision Toolkit"**. For example, the function that captures a picture from a camera and stores the result in a frame buffer is the "Acquisition Tool"; the operation that fits a line to a set of edge points is the "Line Fitter Tool"; and the function that uses an advanced, patented algorithm to identify and locate parts is the "Finder Tool".

To perform an element of a vision application, any number of Vision Tools can be combined into a **"Vision Process"**. Normally, an Acquisition Tool is the first tool in each Vision Process. This is followed by one or more tools to perform part location, metrology or inspection operations. Multiple copies of the same type of tool can be included within the same Vision Process. The individual characteristics of each tool (i.e. its size, position, algorithm parameters) are stored in Object **"Property Lists"**. After a tool is executed, its output data is stored in its Property List in a **"Tool Results"** section. To simplify their definition, tools can be positioned and sized by dragging their graphical handles on top of the display of a camera image.

A Vision Process is the basic unit of execution that can be controlled by a GPL procedure. Using a built-in **"Vision Class"**, a GPL procedure running on a Precise Guidance Controller can start the execution of a Vision Process in PreciseVision via Ethernet and monitor its progress. When the Vision Process completes execution, the GPL procedure can retrieve the Tool Results of any tool within the executed Vision Process by accessing the built-in GPL **"VisResults Class"**. In applications where PreciseVision is not connected to a Guidance Controller, another program executing in the PC or on another processor can be used to trigger a Vision Process and retrieve the results (see the appendix that describes Remote PC Control for additional information).

In most cases, only a single Vision Process is executed in order to perform the complete machine vision task. Typically, this Vision Process will take a picture and then utilize Vision Tools to locate a part and validate some key features or dimension. However, if a more complex machine vision operation is required, a GPL procedure can execute any number of Vision Processes. For convenience, multiple Vision Processes can be stored in a **"Vision Project"** in PreciseVision. At any given time, only one Vision Project can be loaded into PreciseVision and only one of its Vision Processes can be executing.

In order for PreciseVision to return Tool Results in meaningful units, e.g. millimeters, and in an appropriate coordinate system, any camera used for vision guidance must first be "Calibrated". As a convenience, PreciseVision includes a number of **"Camera Calibration"** methods that are described in a later section. Once a camera is calibrated, all data retrieved by a GPL procedure via the VisResults Class will be in millimeters and in the world coordinate system of the robot.

In a typical application, a single copy of PreciseVision will be executed on a PC and will be connected to a single robot or motion controller. If you wish to provide vision results to a number of robots or motion controllers, please see the appendix that describes Multi-Instance Execution.

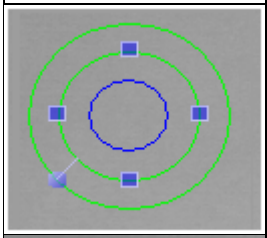
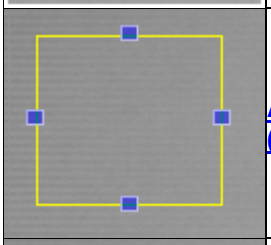
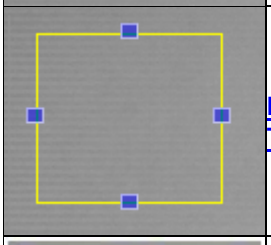
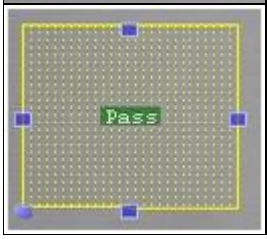
In the following sections, the Vision Tools are briefly described along with the methods for creating and editing Vision Projects, Vision Processes, and tools with their associated Property Lists.

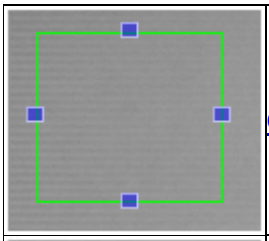
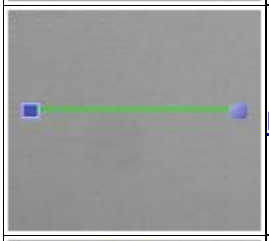
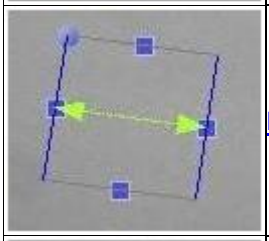
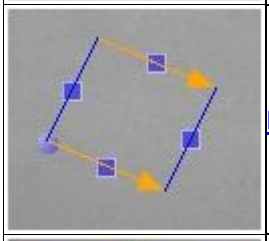
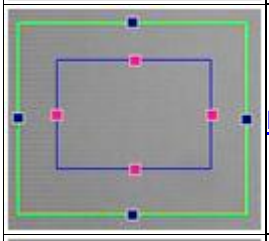
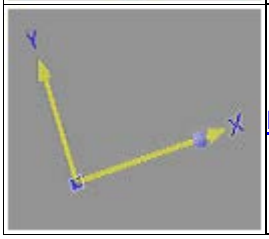
Vision Toolkit Summary

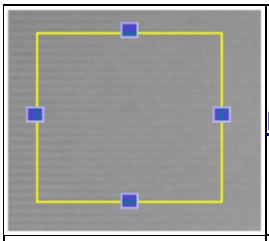
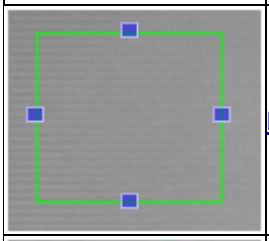
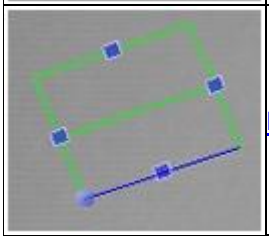
In order to effectively analyze images, PreciseVision includes a full set of vision tools for acquiring images, locating and identifying parts, performing metrology operations and executing visual inspection of key features. In this section, the function of each tool is briefly summarized and a picture displaying how the tool appears on the Camera Display Window is presented.

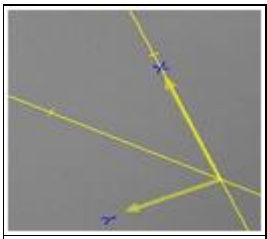
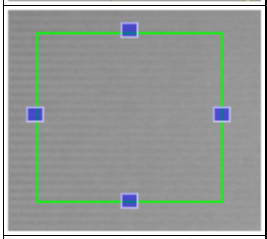
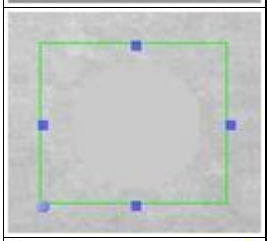
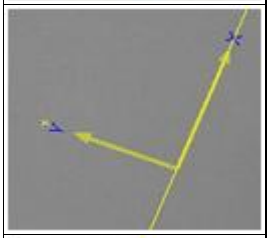
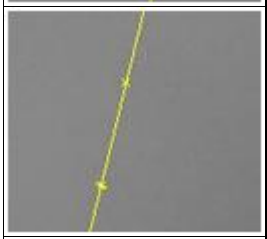
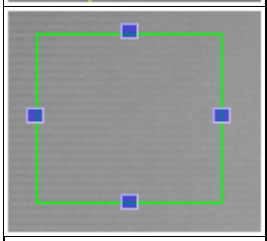
An important feature of PreciseVision is that the output of one tool can be easily utilized as the input for another tool. For example, after the Finder Tool locates one or more parts, a tool that measures the distance between two edges can be easily placed relative to the located part. This measurement tool can then test if a critical dimension of the part is within the desired tolerance. Since this measurement is performed relative to the output of the Finder, it will operate correctly even if the part's position or orientation changes. Furthermore, if the Finder locates more than one part, the measurement tool will be automatically applied to each located part. This ability allows complex measurements and tests to be easily built-up from the basic tools that are provided.

For a complete description of each tool, please read the "Vision Toolkit" chapter. (Note: The algorithms for the complete set of tools has been developed and will be released in future upgrades to PreciseVision during 2007. If you cannot find a tool that you require, please contact Precise.)

Graphical Display	Vision Tool	Description
N/A	Acquisition Tool	Captures an image from a camera or loads it from a disk file, and stores the image into a frame buffer.
	Arc Fitter Tool	Searches a specified region for arc edge points and returns the arc or circle that best fits the edges. Returns both the arc center point and radius.
	Area-Of-Interest (AOI) Tool	Copies a rectangular area-to-interest to a disk file, another region of the current image buffer, or another image buffer. Permits mathematical manipulation when copying to an image buffer.
	Barcode Reader Tool	Reads a variety of standard 1-D and 2-D barcodes and returns the barcode type and the value of the barcode.
	Clear Grip Tool	Verifies that there is no obstruction within the bounds of a defined window. Typically used to confirm that gripping a part will not result in a collision.

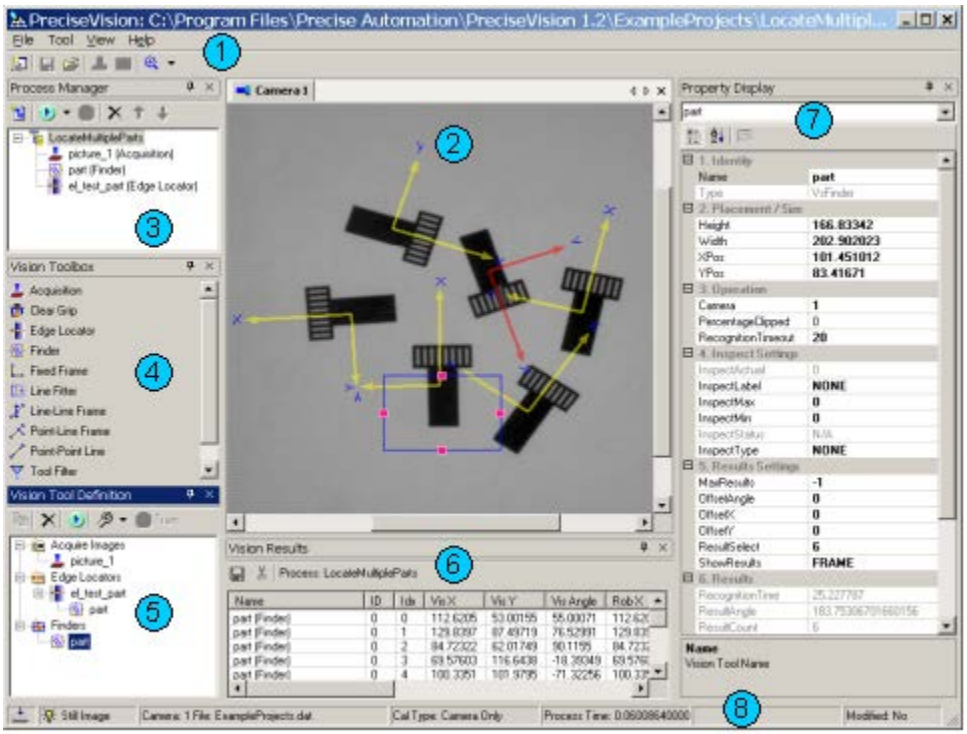
	**Connectivity Tool**	Searches a specified region for binary blobs and returns the centroid and other features of each blob. In addition, this tool can return data that defines the perimeter of any located blob.
	**Edge Locator Tool**	Detects edge points with sub-pixel accuracy along a linear path in a vision image and returns their positions. Alternately, this tool can return an array of the intensities along the linear path or a histogram of their values.
	**Find Centerline Tool**	Searches a specified region for edge points and returns the centerline between two located bounding lines.
	**Find Point Tool**	Locates the edge point that is closest to the line that defines the start of the tool's search region. The XY position of the point is returned with sub-pixel accuracy along (and pixel accuracy perpendicular to) the search direction.
	**Finder Tool**	Most powerful vision tool that identifies randomly placed parts in a camera image and returns their position, orientation and scaling to within sub-pixel accuracy.
	**Fixed Frame Tool**	Places a reference frame at a fixed image coordinate or at a constant offset relative to another vision object. Can optionally index the frame in an X and/or Y grid pattern to repeatedly execute any linked tools.

	**Image Process Tool**	Performs image filtering, edge extraction, thresholding and morphological operations on a specified area in the camera image.
N/A	**Inspect Frame Tool**	Passes through the XPos, YPos and Angle Results of a referenced tool and optionally rotates and shifts the Results based upon an inspection. The inspection can consist of the arithmetic combination of the Results of two other tools. So, this tool also provides a means for performing simple mathematical operations on the results of two tools.
N/A	**Inspect List Tool**	Passes through the XPos, YPos and Angle Results of a referenced tool and tests up to 12 different Results from multiple tools to determine if the object is good or bad. This tool provides a convenient way to perform complex testing of a located object and generating an equivalent location with a single Pass/Fail indicator.
N/A	**Inspect Region Tool**	Inspects the perimeter and the interior of an opaque object to determine if there are any flaws in the object's outline or holes in the interior. This tool was designed to inspect wafers, where slight changes in the overall size and shape of the wafer are permitted, but chips in the perimeter or interior holes or cracks are unacceptable.
	**Light Recorder Tool**	Acquires a sequence of camera images for a specified period of time, determines the brightness of each image, and returns the brightness readings in an array for subsequent analysis.
	**Line Fitter Tool**	Searches a specified region for edge points and returns the line that best fits the edges.

	**Line-Line Frame Tool**	Determines the intersection of two lines defined by vision objects and returns a reference frame. Alternately, computes and returns the midpoint of two reference frames defined by two vision objects.
	**Pixel Window Tool**	Counts edges or binary pixels or collects gray-scale statistics within a rotated rectangular or circular region. Used for quickly detecting the presence of features or collecting general intensity information about a region.
	**Pixel Color Window Tool**	Tests if the general color of a rectangular or circular region matches a trained color within a specified tolerance. Also computes the mean HSV/HSI values for the region.
	**Point-Line Frame Tool**	Takes a point and a line from two vision tools and returns a reference frame.
	**Point-Point Line Tool**	Computes a line given two points from two vision tools.
	**Sensor Window Tool**	Detects motion in a window region. Ensures the scene is stable prior to other operations or waits until motion is detected. Can be used in place of physical sensors in flexible parts feeding where parts can take time to settle.
N/A	**Text Label Tool**	Displays string constants plus inspection and results data from a referenced tool. Text is shown relative to the position of the referenced tool.

N/A	**Tool Filter Tool**	Takes the output from another tool that generates multiple sets of results, and returns a subset of the results based upon specified criteria.

Graphical User Interface Overview



This section summarizes the purpose and use of each of the major components of the PreciseVision graphical user interface (GUI). The user interface has been designed to allow you to quickly construct a Vision Process and to easily define the properties for each of the utilized vision tools.

For detailed information on each of the components of the GUI, please read the "PreciseVision User Interface" chapter. The following table contains a brief summary.

GUI Component	Description
1. Main Menu and Toolbar	Provides access to the major functions available within PreciseVision, e.g. loading and saving projects, selecting the camera output to view, etc. The toolbar includes buttons for putting the camera into "live video" mode, taking a single snap shot, and zooming the camera image up.

2. Vision Camera Display	Displays the output of the last selected camera. This can be either a frozen snapshot or a live camera image. When a Vision Process is being evaluated, the tool Results graphics will be overlaid on top of the camera image. When a tool is being edited, its outline will be displayed on top of the camera image. To reposition or resize a tool, simply drag its handles in this window.
3. Process Manager	Displays and permits editing of all of the Vision Processes defined within the currently loaded Vision Project. Each process can be expanded to show the sequence of Vision Tools that are executed within the process. Tools can be deleted or re-ordered within the process. To add a new tool, simply drag a defined tool from the Vision Tool Definition window and drop it on a process name. To preview how a Vision Process will perform, click on the "Test Process" green button in the toolbar. This will execute a process once or continuously without requiring a GPL procedure to trigger this activity.
4. Vision Toolbox	Lists all of the possible types of Vision Tools that can be created. To create a new copy of a tool, double click on the tool or right-click and select "Create tool". Each tool must be assigned a unique name. New copies of tools are automatically added to the lists in the Vision Tool Definition window.
5. Vision Tool Definition Window	Displays all of the tool that have been created. A created tool will not be utilized in a process until you drag the tool and drop it into the Process Manager window. If a tool is relative to another tool or requires other tools as inputs, they will be listed below the tool. This provides a quick visual queue as to the relationships between tools.
6. Vision Results Window	Whenever a Vision Process is executed, the output of each tool (i.e. its Vision Results) will be displayed in this window.
7. Property Display	Whenever a copy of a vision tool is selected in the Vision Tool Definition window, all of the properties of the tool will be displayed in this window. Dimmed properties are read only.
8. Status Bar	Displays status information such as the last Process execution time, live video/snapshot indicator, networking indicator, camera calibration type, etc.

Tutorial 1: Testing Processes and Editing Vision Tools

The first step in learning how to apply PreciseVision is to gain more of an understanding of how vision tools operate and how to edit tools. Fortunately, PreciseVision provides a very convenient means for experimenting with the tools without needing a camera, robot or any parts.

In the following procedure, you will be using a Sample Vision Project and camera images from disk files that are distributed with PreciseVision. One thing to note is that this Sample Project and image files were generated using standard PreciseVision facilities. So, once you are familiar with the system, you will be able to create your own Sample Projects and stored images to illustrate your work to co-workers or customers or to relay problems to Precise and our partners.

Prior to executing this tutorial, the following steps must first be performed:

- The procedure for "Installing PreciseVision on a PC" must be completed.
- No camera hardware of any sort is required.

In this first example, two Edge Locators are executed to compute a line along the side of a part.

» Start by launching PreciseVision. Click on ***"Start > Programs > Precise Automation > PV xxx > PreciseVision".***

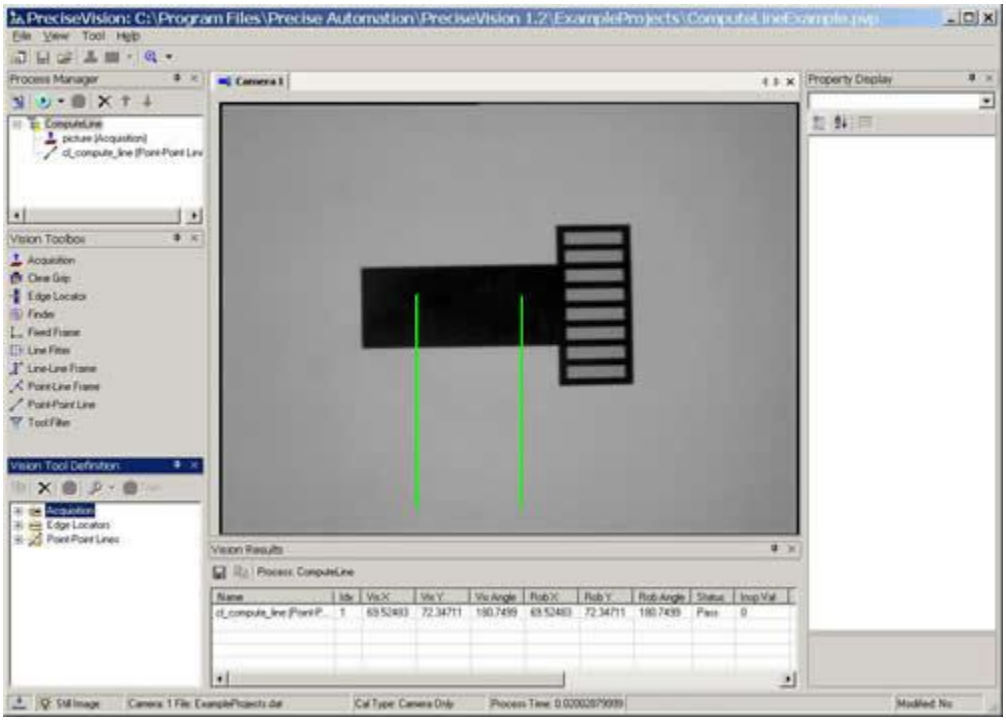
If this is the first time PreciseVision is executed, a pop-up window will be displayed requesting the specification of the camera hardware followed by a window requesting that the software be activated. Both of these actions are good to perform and you should return to the installation chapter and follow the instructions for completing these procedures. However, if you wish, you can cancel out of these pop-up's since neither of these setup steps is required to initially execute the Sample Projects.

The following "Quick Start" pop-up window should now be displayed.

If you hover the cursor over the three buttons on the left, the text in the center window panel will change to provide more information about each selection.

» Click on the ***"Samples"*** button. A standard file selection pop-up window will be displayed that lists a number of Sample PreciseVision Project files (*.pvp).

» Click on the ***"ComputeLineExample.pvp"*** file and press ***"Open"*** to load this Project.

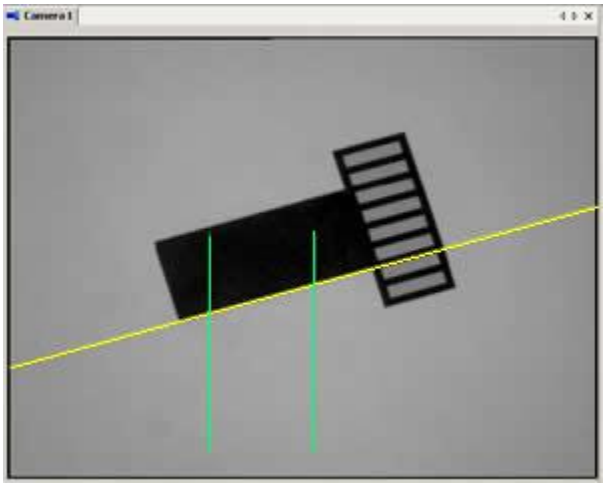


The default PreciseVision application layout should now be visible and the Camera Display should show the picture of a "T" part with two green Edge Locators as shown above.

If you examine the Process Manager, you will see that this Project has one Vision Process defined, "ComputeLine". This Process consists of an Acquisition Tool, "picture", and a Point-Point Line Tool, "cl_compute_line". The Acquisition Tool would normally be set to capture an image from a camera. In this case, it is configured to read the image from a disk file. cl_compute_line does the work of computing a line from the output of two Edge Locators. The two Edge Locators are implicitly executed and not listed in the Vision Process (more about this later).

» To test the Vision Process, click on ***"ComputeLIne"*** in the Process Manager and then click on the circular green ***"Test Selected Process"*** button in the window's toolbar.

Each time that you click on the "Test Selected Process" button, the Acquisition Tool loads an image file from the disk and then the Point-Point Line Tool executes the two Edge Locators and fits a line between their first detected edges. The first time you click the test button, the Camera Display should resemble the following. You will note that the output line of the Point-Point Line Tool is indicated in yellow.

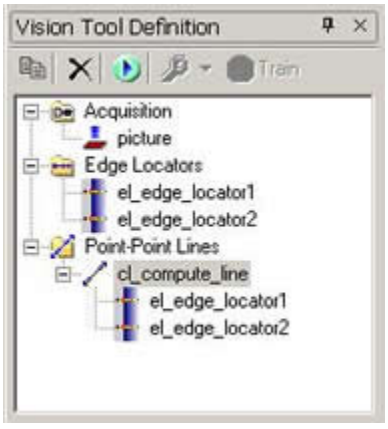


For this Sample Project, there are multiple image files and so the captured image and the results will change each time you click the test button.

» To continuously execute the Sample Project, pull down on the arrow next to the test button in the toolbar and select ***"Test Loop"***.

» To stop the execution, click the ***"Stop Test Loop"*** button in the toolbar.

» Next, to examine the defined tools, in the Vision Tool Definition Window, click on each of the ***"+"*** boxes to expand all of the displayed items.



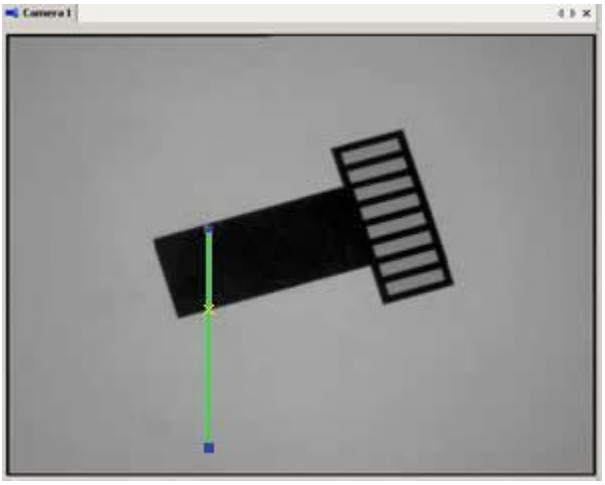
The items in the left-most column of the Vision Tool Definition Window are folders that contain all created tools of a specific type. If a type of tool has not been created, the corresponding tool folder is hidden. The top-level items listed in the tool type folders are the names of the created tools. The third and successive levels are lists of tools that are referenced by the tool above it. In this example, in addition to the "picture" and the "cl_compute_line" tools, two Edge Locators, "el_edge_locator1" and "el_edge_locator2" have been created. Note that the names of the Edge Locators appear twice, once in

the top-level Edge Locator tool type list and a second time under the "cl_compute_line" tool. This second listing indicates that the output (i.e. "Results") of these two Edge Locators are referenced by and used in the computation of "cl_compute_line".

Since the system knows that the Edge Locators are needed for the computation of the Point-Point Line Tool (more about how to specify this later), the Edge Locators are automatically executed by the Point-Point Line Tool if necessary and they need not be included in the Vision Process. Including the Edge Locators in the Vision Process before the computation of the Point-Point Line Tool is a matter of personal preference and does not effect execution time.

The Vision Tool Definition Window provides a convenient means to access all created tools whether or not the tools are included in any Vision Process. This window also provides a simple graphical representation of how tools are dependent on and connected to other tools.

» In the Vision Tool Definition Window, click on either of the ***"el_edge_locator1"*** names to select this tool for editing.



Once the Edge Locator is selected for editing, it will be displayed in the Camera Display with graphical handles (as shown above) that allow it to be repositioned and resized.

» To practice repositioning the tool, click anywhere along the green line and drag the tool to translate it.

» Click on the blue square that indicates the start of the Edge Locator and drag it to stretch or shrink the tool.

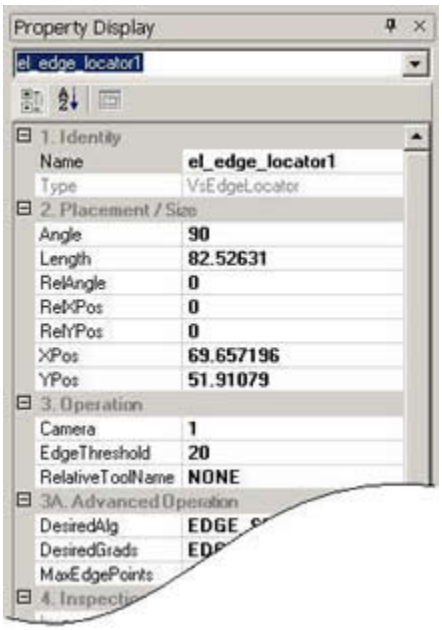
» Click on the blue circle that indicates the end of the tool and drag it to rotate the tool's orientation.

» To test the Vision Process with a new tool position and size, in the Process Manager,

click on ***"ComputeLIne"*** and then click on the circular green ***"Test Selected Process"*** button in the window's toolbar.

When the Edge Locator was selected for editing, you might have noticed that its properties were automatically displayed in the Property List Display as shown below. In addition to graphically positioning and sizing this tool, you can also enter new values into this panel. The properties are organized into groups based upon their function. Readonly values are displayed in dimmed characters.

For each of the tool types, all of the properties are described in detail in the Vision Toolkit Chapter of this document.



» In the Property List Display, try editing the ***"Angle"***, ***"Length"***, and ***"XPos"*** and ***"YPos"*** properties and note the effect in the Camera Display.

» If you wish to save your changes to the Sample Project, select ***"File > Save Vision Project As..."*** and assign a new name to your project so the original Sample is preserved.

This completes the tutorial on testing a Sample Project and editing a tool.

Tutorial 2: Adding Vision Tools

In this tutorial, we again make use of the ***ComputeLineExample*** Sample Project. In this case, we present the procedure for adding a Vision Tool to an existing Vision Process. For demonstration purposes, we will add an Edge Locator that measures the width of the stem of the "T" and validates if this dimension falls within an acceptable range. In an application that required high precision, you would probably use two Line Fitters for locating the edges of the "T" stem, but for training purposes, an Edge Locator will suffice.

Prior to executing this tutorial, the following steps must first be performed:

- The procedure for "Installing PreciseVision on a PC" must be completed.
- No camera hardware of any sort is required.

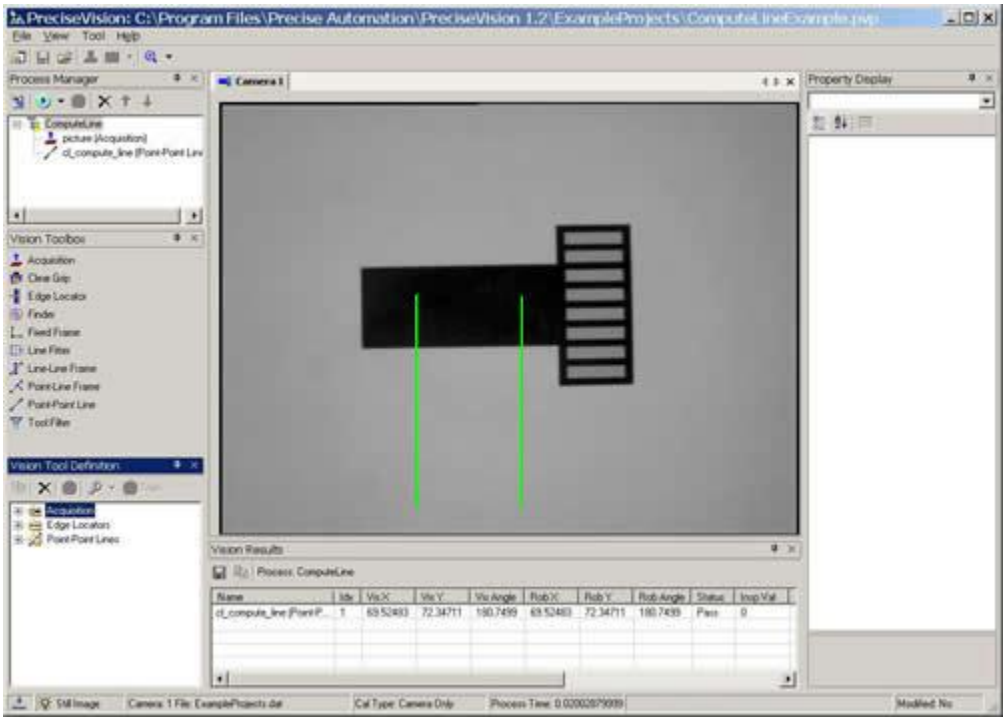
» If PreciseVision is not running, click on ***"Start > Programs > Precise Automation > PV xxx > PreciseVision".***

» If PreciseVision is already running, select ***"File > Quick Start".***

If a pop-up window is displayed requesting the specification of the camera hardware or a window is displayed requesting that the software be activated, you can cancel out of these pop-up's since neither of these setup steps is required to initially execute the Sample Projects.

» In the Quick Start window, click on the ***"Samples"*** button. A standard file selection pop-up window will be displayed that lists a number of Sample PreciseVision Project files (*.pvp).

» Click on the ***"ComputeLineExample.pvp"*** file and press***"Open"*** to load this Project.



The default PreciseVision application layout should now be visible and the Camera Display should show the picture of a "T" part with two green Edge Locators as shown above.

For robustness, the visual inspection should be designed such that the measured width of the stem is accurate independent of the orientation of the "T". To achieve this, the new Edge Locator should be oriented perpendicular to the computed Point-Point Line. If the Edge Locator was just placed at a fixed position and orientation, the width measurement would return too large a value if the "T" is rotated. To create the new tool, the following should be executed.

» Double click on the ***"Edge Locator"*** in the Vision Toolbox.

» In the Name pop-up, enter ***"el_measure_width"*** as the name of this new tool and press ***"Ok"***. The new tool will be created and will appear in the Vision Tool Definition Window.

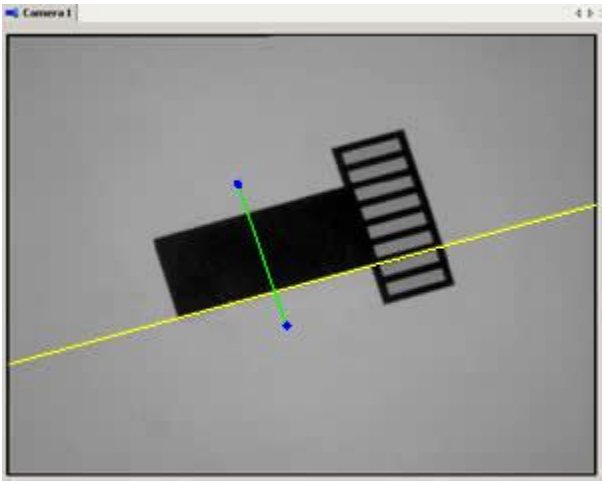
» Click on the new tool in the Vision Tool Definition Window and drag it to the Process Manager and drop it on ***"ComputeLine"***. This will evaluate the new tool after the Results of cl_compute_line are computed.

» In the Property List Display, pull down the entries for ***"RelativeToolName"*** and select ***"cl_compute_line"***. This makes the position and orientation of the new tool relative to the computed position and orientation of the Point-Point Line Tool.

» In the Camera Display Window, resize and position the new tool so that it is

approximately perpendicular to the yellow line for cl_compute_line and stretched across the stem of the "T".

The Camera Display should resemble the following picture that shows the el_measure_width tool drawn across the stem of the "T" and approximately perpendicular to the yellow line for cl_compute_line.



» To ensure that el_measure_width is exactly perpendicular to cl_compute_line, in the Property display, edit ***"RelAngle"*** to be an even multiple of 90. For example, if it has a value of -628.2, change this to -630.

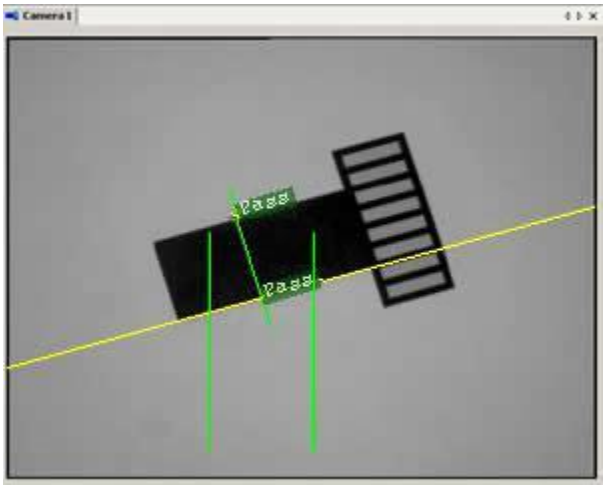
» To automatically test the width of the stem, change ***"InspectType"*** to ***"Distance_From_First_To_Last"***. This defines the inspected results as the computed distance between the first and last edge found by el_measure_width.

» To define the width dimensions that we will accept as valid, set the ***"InspectMax"*** value to ***"31.8"*** and the ***"InspectMin"*** value to ***"31.2"***.

» To display a pass or fail label on the tool in the Camera Display, change ***"InspectLabel***" to ***"Pass_and_Fail"***.

» To test the new tool, in the Process Manager, click on ***"ComputeLine"*** to select it and then click on the circular green ***"Test Selected Process"*** button in the window's toolbar.

Each time you click the test button, you should now see the new tool in addition to the three former tools. The new tool should be drawn perpendicular to the yellow computed line and should be labeled to indicate if the width dimension is acceptable. The following illustrates a typical image in the Camera Display.



As you execute the modified Process, you should review the data displayed in the Results Window to see the numerical value of the computed width. All of the Results information is also available by selecting individual tools and viewing the data in the Properties List Display.

» If you wish to save your changes to the Sample Project, select ***"File > Save Vision Project As..."*** and assign a new name to your project so the original Sample is preserved.

This completes the tutorial on adding a tool to an existing Vision Process.

Tutorial 3: Applying A Finder Tool

The Finder is the most general purpose tool available for identifying parts or sections of parts and locating their position, orientation and scale. For constrained or special situations, other tools can be combined to perform some of these functions. As an example, in the first tutorial, three tools were combined to locate a known part. But, in that case, the part was only expected to translate and rotate within a limited range. The Finder Tool utilizes an advanced patented algorithm to both identify and locate parts that are randomly positioned based upon a simple one-shot training technique.

In this tutorial, we step through the procedure for creating and teaching a Finder using a Sample Project. This tutorial also includes an example of creating a new Vision Process within an existing Vision Project.

Prior to executing this tutorial, the following steps must first be performed:

- The procedure for "Installing PreciseVision on a PC" must be completed.
- No camera hardware of any sort is required.

» If PreciseVision is not running, click on ***"Start > Programs > Precise Automation > PV xxx > PreciseVision".***

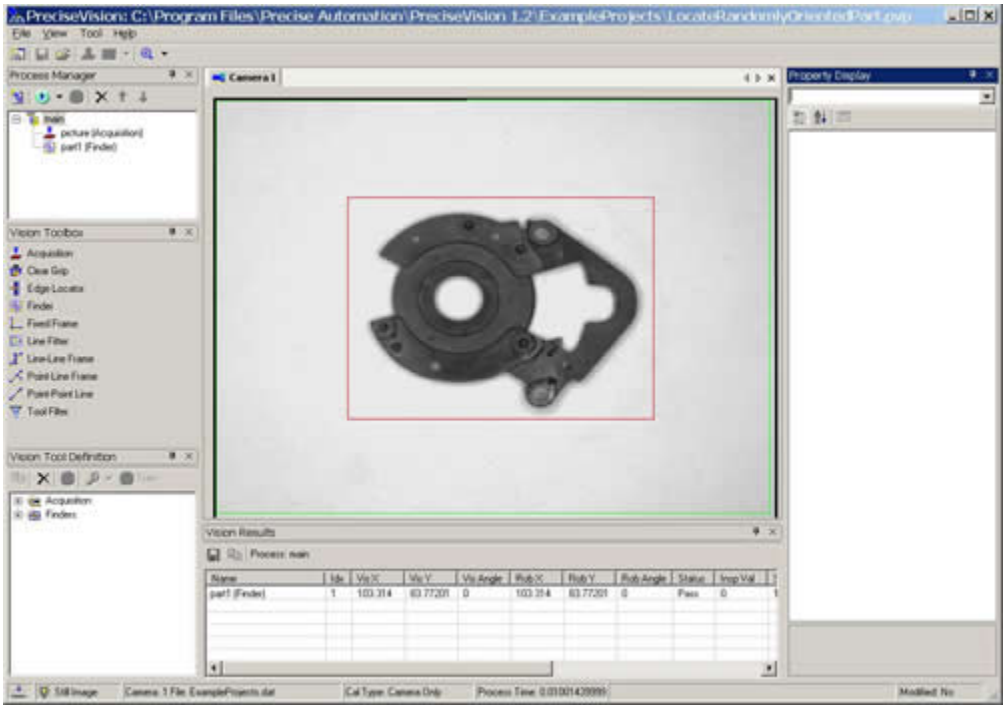
» If PreciseVision is already running, select ***"File > Quick Start".***

If a pop-up window is displayed requesting the specification of the camera hardware or a window is displayed requesting that the software be activated, you can cancel out of these pop-up's since neither of these setup steps is required to initially execute the Sample Projects.

» In the Quick Start window, click on the ***"Samples"*** button. A standard file selection pop-up window will be displayed that lists a number of Sample PreciseVision Project files (*.pvp).

» Click on the ***"LocateRandomlyOrientedPart.pvp"*** file and press ***"Open"*** to load this Project.

The default PreciseVision application layout should now be visible and the Camera Display should show the picture of a circular plastic part with a "V" shaped attachment as shown below.



This Sample Project already includes a Finder as its principal tool. In this tutorial, we will go through the steps of re-creating this Vision Process including the Finder. Before beginning, the supplied Process should be tested to see how the Finder performs on this part.

» To see the Finder in operation, click on ***"Main"*** in the Process Manager and then click on the circular green ***"Test Selected Process"*** button in the window's toolbar.

» To continuously execute the Sample Project, pull down on the arrow next to the test button in the toolbar and select ***"Test Loop"***.

» To stop the execution, click the ***"Stop Test Loop"*** button in the toolbar.

We will now begin creating a new Vision Process. In order to perform the image capture, we will reuse the existing Acquisition Tool since it has already been initialized to read the desired image disk files.

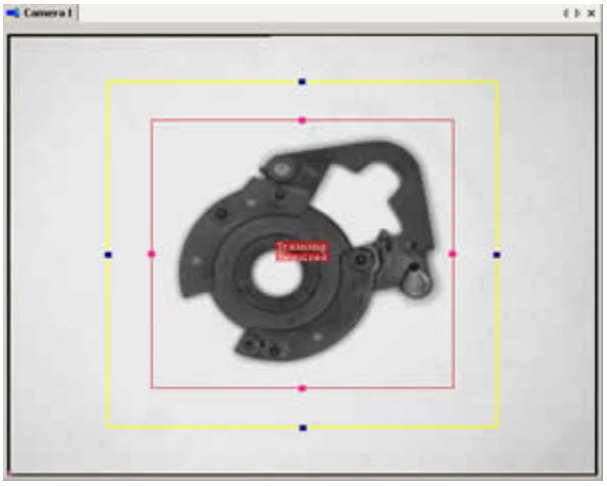
» Click on the ***"Add a new Process"*** button in the Process Manager toolbar.

» In the Process Name pop-up window, fill in ***"my_process"*** as the name and press ***"Ok"*** to create the Process.

» In the Vision Tool Definition Window, in the Acquisition folder, click on the ***"picture"*** tool and drag and drop it on top of the ***"my_process"*** in the Process Manager.

Before performing the next step of creating and training a new Finder, the following is background information and hints that will aid in applying this tool.

When you create a new Finder or re-teach ("re-train") an existing Finder, red and yellow rectangles will be drawn in the Camera Display as shown below.



The red rectangle defines the "Finder Template" for the part to be located. When PreciseVision subsequently searches for the part, it is looking for matches for the Template in the image. When the system reports the position and orientation of a matched part, it is actually stating the position of the center of the Template when the match was found and the amount by which the Template was rotated. When defining the Template, keep in mind the following guidelines.

- The Template should fully enclose the target object and should contain no other objects or parts of objects.
- The Template should include some of the background on all sides of the target object, typically at least 16 camera pixels (too many is better than too few). This allows the system to distinguish the boundary of the object from the background, which is especially important for parts with few internal features. For a 640x480 camera, 16 pixels is 1/30th of the full height of the camera image.
- If the background varies considerably (i.e. it is dirty or has blotches), the part should be trained on a uniform neutral background.
- The target object should be oriented in the camera image in what you want to define as 'Zero' degrees. That is, when the Finder later locates a part is in the trained orientation, it will return an orientation angle of 0 degrees.

The yellow outline defines the "Training rectangle". This rectangle is used to teach the system how to distinguish the Template from other objects or background clutter. When defining this rectangle, keep in mind the following guidelines.

- This rectangle must be larger than the Template and must fully enclose the Template.
- This rectangle should include one and only one complete copy of the target object. It's okay if this region includes other partial copies of the target object so long as each partial copy is less than 50% of the target object.
- This rectangle should include any other objects or clutter that are expected to be encountered during runtime to aid the system in learning to distinguish between the target object and its surroundings.
- This rectangle should be as large as possible without violating the rules above. A larger region permits the system to test more positions and orientations for the Template during the training phase and assists in optimizing the runtime search strategy.

It's now time to create the Finder and add it to the Vision Process.

» Double click on the ***"Finder"*** in the Vision Toolbox.

» In the Name pop-up, enter ***"my_finder"*** as the name of this new tool and press ***"Ok"***. The new tool will be created and will appear in the Vision Tool Definition Window.

» Follow the step-by-step instructions within the "Finder training" Application Wizard that is displayed to setup the Finder keeping in mind the hints above.

» Click on ***"my_finder"*** in the Vision Tool Definition Window and drag it to the Process Manager and drop it on ***"my_process"***. This will evaluate the Finder after the camera acquisition is executed.

» To test the new tool, in the Process Manager, click on ***"my_process"*** to select it and then click on the circular green ***"Test Selected Process"*** button in the window's toolbar.

Congratulations, you just created your first Vision Process that identifies and locates randomly parts! You can continue to click on the test button or execute your process in a continuous loop.

» If you wish to retrain the Template at anytime, click on ***"my_finder"*** in the Vision Tool Definition Window to select it. Then click on the arrow to the right of ***"Finder Operation Mode"*** icon on the window toolbar and select ***"Train Mode"***. You can also restart the Finder Training Wizard in this same pull-down if you wish.

» If you wish to save your changes to the Sample Project, select ***"File > Save Vision Project As..."*** and assign a new file name to your project so the original Sample is preserved.

» At this time, we highly recommend that you execute each of the distributed Sample Vision Projects, review their Vision Processes and edit the Tool Properties to get an understanding of more of the vision tools.

This completes the tutorial on creating a Vision Process and applying the Finder Tool.

Tutorial 4: Developing a Robot Vision Application

This tutorial illustrates how to develop a simple, but complete, robot vision guidance application. It presents both the PreciseVision Project and the GPL software that must be executed on a Guidance Controller. This tutorial requires that your robot and Guidance controller be operational and interfaced to the PC executing PreciseVision.

Prior to executing this tutorial, the following steps must first be performed:

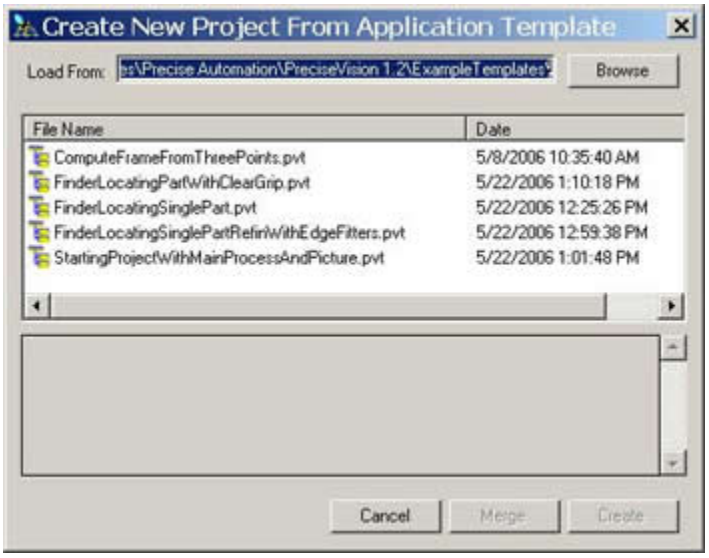
- The procedure for "Installing PreciseVision on a PC" must be completed.
- A camera must be rigidly mounted over a region that can be reached by the robot.
- The camera interface must be installed and interfaced to PreciseVision using one of the procedures in the "Hardware Installation" section of this manual.
- The robot must be fully operational and interfaced to the PC via Ethernet.
- The procedures outlined in "Configuring Communication Between the PC and the Controller" that permit the two systems to exchange data must have been performed.
- The "Robot Vision Area Camera Calibration" must have been executed to define the camera's pixel-to-millimeter conversion and the transformation from the camera's frame of reference to the robot's frame of reference.
- The Guidance Development Suite must be installed on the PC and you should have a modest ability to develop and execute GPL programs.

» If PreciseVision is not running, click on ***"Start > Programs > Precise Automation > PV xxx > PreciseVision".***

» If PreciseVision is already running, select ***"File > Quick Start".***

» In the Quick Start window, click on the ***"New Project"*** button.

You should now see the "Create New Project From Application Template" pop-up window that resembles the following.



The "Create New Project From Application Template" is a wizard that builds a complete Vision Project for a number of common tasks. For example, the "FinderLocatingPartWithClearGrip" constructs a Project that uses a Finder Tool to locate an object and then executes two ClearGrip Tools to verify that the parts can be grasped without hitting an obstruction. In particular, this wizard deletes the current Vision Project (after confirmation if the Project should be saved), creates a new Project, and automatically creates the Vision Tools and Vision Processes for the selected task. The wizard then walks you through the steps to position, size and train each of the tools for your parts. Over time, we will be adding additional templates to address even more common tasks and we will be adding the ability for you to add your own templates to simplify the use of the system by your co-workers or customers. Of course, you can always create your own Vision Projects, Processes and Tools using the basic methods described in the other tutorials. The Application Wizard is just a convenience for people learning about the system or it can serve as a starting point for those already familiar with the system.

For this exercise, we will be selecting the "FinderLocatingSinglePart" template. This takes a single picture and then employs a single Finder Tool to locate your part.

» Click on the ***"FinderLocatingSinglePart.pvt"*** file, click on ***"Create"*** and then follow all of the instructions of the wizard to create your Vision Project. At the conclusion of the wizard, the Project, Process and Tools should all be created and the Finder should be trained on your object.

» Save your Project by selecting ***"File > Save Vision Project As..."*** and assigning a file for your Project.

» To test your Project, place your object in the camera's field-of-view, click on ***"Main"*** in the Process Manager and then click on the circular green ***"Test Selected Process"*** button in the window's toolbar.

» To validate that the Vision Project and the camera calibration are working properly, using the same robot pointer you used during the camera calibration process, manually move the robot to the X/Y position indicated in the Results Window ***"RobX"*** and ***"RobY"*** values and verify that the tool is at the center of your part's Finder Template.

PreciseVision is now setup and ready to receive commands. It is not necessary to place PreciseVision in a special mode to work with the robot, it is always listening for commands from a GPL system. So, there is no further development to be done in PreciseVision.

The next step is to develop a GPL project that will execute the Vision Process "Main", retrieve the "part1" Finder Results and move the robot to the specified position. A simplified version of the required program is presented in the *"Guidance Programming Language, Introduction to GPL"* manual in the Vision section and is repeated below.

» Within GDE, the Guidance Development Environment, in the Project Manager, create a new GPL Project named ***"Robot_vision"***.

» In the new ***"Robot_vision"*** Project, double click on the ***"Main.gpl"*** file to begin editing.

» In the editor window, below the "Public Sub Main" statement, insert the following lines of text. If you are reading this exercise in the *Precise Documentation Library* (the online help file), this can be accomplished by copying and pasting if you wish to save yourself some typing.

```
Dim vis As New Vision
Dim vResult As New VisResult

Robot.Attached = 1
Move.Loc(safe, vsProfile)

vis.Process("Main") ' Run vision process "Main"
If vis.ResultCount("part1") = 0 Then
    Console.Writeline("Vision object not found")
    Goto done
End If
vResult = vis.Result("part1", 1) ' Get results
```

```
        ' Create a reference frame object and set it
        ' equal to the returned vision location
        Dim vsRefFrame As New RefFrame
        vsRefFrame.Loc.PosWrtRef = vResult.Loc

        ' Pickup point is relative to new frame
        vsRelPoint.RefFrame = vsRefFrame

        Move.Approach(vsRelPoint, vsProfile)
        Move.Loc(vsRelPoint, vsProfile)
        Move.Approach(vsRelPoint, vsProfile)

        ' Move back to safe location
        Move.Loc(safe, vsProfile)

    done:
```

» In the the ***"Robot_vision"*** Project, double click on the ***"GModule.gpo"*** file to access the global Motion Objects panel.

» Create and teach a Location named ***"safe"*** that positions the robot so it is not obstructing the camera's view of the part. Also, create a Motion Profile named ***"vsProfile"*** that will be used to slowly move the robot to the part.

» Create a Location named ***"vsRelPoint"***. This will define the robot's pickup position relative to the Location returned from PreciseVision. Normally, it would be best to manually define this Location relative to the actual value returned from the vision system. But for demonstration purposes, set its properties to the following where the Z value is a safe world coordinate value above the part.

X = 0
Y = 0
Z = Z_safe
Yaw = 0
Pitch = 180
Roll = 0

» Press the ***"Save Document"*** icon on the main tool bar to save your changes.

» To load your application, in the Project Manager, drag the ***"Robot_vision"*** Project from the PC to the top ***"Loaded Projects"*** panel and drop it.

You have now completed the development of the GPL Project and we are ready to execute the complete vision guided robot application.

DANGER: Before proceeding with this procedure, please ensure that the robot has been properly mounted, all required safety interlocks have been installed and tested, and power has been connected. For the PrecisePlace robots, this information is provided in the *"PrecisePlace 2300/2400 Robot, Hardware Introduction and Reference Manual"*.

» In the GDE Robot Control Window, set the ***"Robot Speed"*** to approximately 5%. This will force your Project to move the robot at 1/20th of its normal speed.

» **This next action will move the robot automatically to a position computed by PreciseVision, so be prepared to hit the E-Stop button or disable power if any problem occurs!**

» With your part placed in the camera's field-of-view, execute the GPL Project. When you run a GPL program for the first time, it is advisable to start the project in debug mode. This can be done by selecting ***"Debug > Start with break"*** or pressing ***"Shift-F5"***. Single step through the program one line at a time by pressing ***"F10"***.

Congratulations, you just created your first full vision guided robot application!

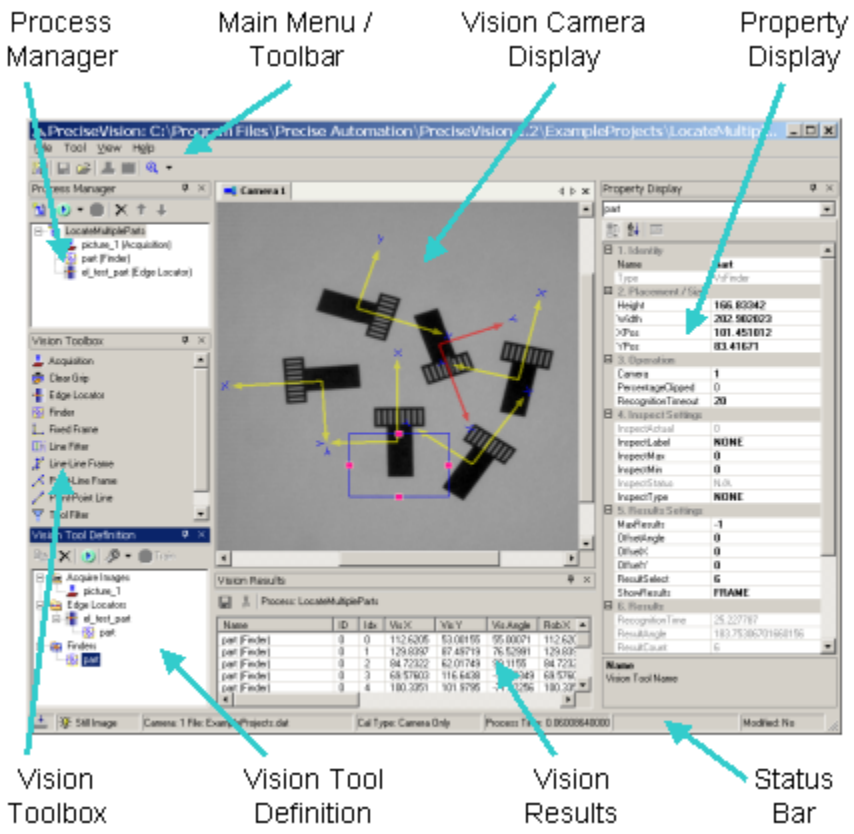
For more information on the GPL vision interface, please read the Vision Guidance chapter of the *"Guidance Programming Language, Introduction to GPL"* manual. This contains an overview of the built-in GPL vision classes, methods and properties. For more detailed information, please see the *"Guidance Programming Language, GPL Dictionary Pages".*

Graphical User Interface

PreciseVision User Interface

The PreciseVision Graphical User Interface (GUI) consists of a title bar that displays the name of the opened project, a top menu bar, a main toolbar, and a variety of dockable windows. Each of the dockable windows can be displayed or hidden, resized and repositioned into the arrangement that is most efficient for your use.

When PreciseVision is started for the first time, the following default arrangement of the various elements of the GUI is presented.



To reposition a window, you simply click in its title bar and drag it to its new location. If you drag a window on to another window, they can split the space or share the space using tab controls. If you click on the close icon (x) in the top right of a window's title bar, you can use the "View" top-level menu to

redisplay the window. If you click on the "push pin" in the top right of a window's title bar, you will either "pin" a window and fix its location or "un-pin" a window so it can share its space with another window.

Windows can be resized by grabbing a border and stretching or shrinking it to the desired dimension.

In the following sections, the functions supported by each of the major elements of the PreciseVision GUI (i.e. main menu bar, tool bars, and windows) will be described. In addition, the Preferences Panel for specifying system configuration information is also described. The graphical components are presented in order of importance rather than alphabetically. To jump to a specific description, click on a label in the picture above or on a window.

Main Menu and Toolbar

The Main Menu and the Main Toolbar provide access to general functions such as displaying the dockable windows, saving and loading Vision Projects and launching the camera calibration operations. This menu and toolbar appear as follows.

The following tables describe the operation of each of the selections within the Main Menu and are organized by the name of the top-level menu. These tables are followed by a description of each of the Toolbar buttons.

File	Description
Quick Start	Displays the pop-up Quick Start window that is shown each time PreciseVision is started. This window allows an existing Project to be loaded, sample canned Projects to be loaded, or a new Project to be constructed by a step-by-step wizard that supports many common procedures. The latter two options are very instructive for individuals who wish to learn about machine vision or PreciseVision.
New Vision Project	Displays a sub-menu with selections that create either an empty un-named Vision Project or a new Project constructed by a step-by-step wizard that supports many common procedures. An additional sub-menu selection permits the last executed wizard to be re-executed to allow editing of the previously generated Vision Process.
Open Vision Project	Opens a new Vision Project from the disk and loads it into memory replacing the current Vision Project. The standard file pop-up window is displayed to allow browsing to the desired folder. PreciseVision project files have a ".PVP" file extension.
Import Vision Project	Loads in the Vision Project contained in a file and merges its contents with the currently loaded project. If the new project contains any processes or tools whose names conflict with items

	that are already loaded, "_r" is appended to the name of the new item. Also, any camera calibration information that is contained in the new project is ignored.
Export Vision Project	Copies the currently loaded Vision Project into another directory. As part of the copy operation, the extra files associated with certain vision tools, such as Finder templates files, and the camera calibration files can also be automatically copied. Normally, PV stores camera calibration files into a central folder to permit them to be shared by multiple Projects. If you manually copy a project area, the calibration file will often not be copied or the link to the file will be broken. The Export utility can automatically copy the applicable calibration files with the other Project files and can zero the link. If the link is zeroed, when the Project is reloaded, the calibration files stored in the Project folder will be utilized. The Export function is also useful if you have multiple PV systems, each with unique camera calibrations, but you wish to execute the same Vision Project. In this case, you should Export the Project without calibration files and with zero'ed links. For each PV system, you start by loading its unique calibration files. You then load the exported Project to associate the generic Project with the individual calibration files.
Save Vision Project	Either saves the loaded Vision Project to its associated disk file or displays a pop-up file window to allow the operator to browse to the desired disk folder and assign a disk file name.
Save Vision Project As ...	
Close Project	Closes the loaded Vision Project after prompting if any changes should be saved to the Project's disk file. The close operation deletes all defined tools and Vision Processes resulting in an empty un-titled Project.
Preferences	Displays the PreciseVision Preferences panel. This panel contains a number of application setup parameters that customize the operation and appearance of PreciseVision, e.g. enable/disable the display of the Main Toolbar or tool graphics.
Acquire New Image	Takes a single snap shot from the currently selected camera and loads the image into the Camera Display window.
Reinitialize Camera Connections	Reinitializes cameras that have stopped operating after being disconnected and reconnected or due to network connectivity disruptions. PreciseVision normally tries several times to reconnect to cameras that are no longer operating, but will eventually stop retrying. This function forces the system to perform additional retries.
Image Load / Save	Diagnostic and debugging aid that loads a camera image from a disk file into the Camera Display window or saves the currently displayed image to a disk file.
Lock / Unlock Application	Locking an application prohibits users from modifying the loaded project. This is useful for avoiding situations where a user might inadvertently change a parameter or tool and cause a vision project to no longer operate correctly. Locking dims out all menu items, removes most toolbar items, and inhibits closing PreciseVision.

	Unlocking requires a password if a password has been defined in the Preferences panel.
Protect / Unprotect	Protecting a selected PV ".pvp" disk file encrypts the contents of the file and prevents the vision project from being viewed or modified when it is loaded into PV. The vision project is password protected and a duplicated encrypted file with a ".pvr" extension is created. This prevents distributed projects from being inadvertently modified and protects the vision project algorithm. Once a protected vision project is loaded, it must be started using a remote command since many of the manual operator controls are disabled.
Exit	Terminates execution of PreciseVision

View	Description
Layouts	Provides "Load Default", "Load Layout" and "Save Layout" selections for permitting custom desktop layouts to be preserved, reloaded or set back to the normal default. Also provides "Load Runtime Layout", "Load Edit Layout" and "Load Calibration Layout" for putting standard layouts into effect. These standard layouts can also be accessed from the main toolbar.
Select Camera	Selects the camera whose image is to be presented in the Camera Display window. This also specifies the camera that is utilized in the Camera Calibration methods and other functions.
Debug	Brings up a submenu that permits displaying log files that contain the bi-directional communication messages between PreciseVision and Guidance Controllers or Remote PC Servers. Generating these log files can be enable via the Preferences Pop-Up.
Process Manager	Permits any dockable window that has been closed to be re-displayed.
Vision Tool Definition	
Vision Toolbox	
Property Display	
Results Display	
Guidance Web Panels	Opens up to three web panels specified in the 'Preferences -> Web Interface' tab. This provides convenient access to the Precise Guidance Controller web interface.
Project Version	Sets and displays the version number and text description for the currently loaded vision project.
Refresh	Redraws any graphics generated by vision tools in the Camera Display window. This is convenient because the graphics can sometimes be lost when PreciseVision is minimized or if you click on certain panels.

Tool	Description

Calibrate Vision	Initiates the execution of the camera calibration wizard. This procedure determines the pixel-to-millimeter conversion for the selected camera and also the conversion from the camera's frame of reference to that of the robot.
Camera Calibration Configuration	Displays a pop-up window that associates a camera calibration file with a specific camera in the current Vision Project. The information in this pop-up is saved and reloaded with the Vision Project.
Show Camera Information	Displays a pop-up window that shows which camera is selected and the camera calibration data that is in effect.
Camera Video Properties	Displays a pop-up window that permits the brightness range and offset for the currently selected camera to be set. This is helpful during the setup procedures for the Camera Calibration methods and other functions.
Show Vision Tool Cross Reference Information	Displays a pop-up window that either: (1) lists the vision processes and vision tools that reference each defined tool or (2) lists the defined vision tools that are not referenced by any vision process or any vision tool. This function is useful for understanding the relationship between vision tools in complex applications.

Help	Description
Contents	Opens the *Precise Documentation Library* in a separate window and jumps to the specified section that is relevant to PreciseVision.
PreciseVision GUI	
Vision Tools Overview	
GPL Programming	
Product Activation	Displays the pop-up window used for activating this product.
Camera Driver Info	Presents the version and identification information for installed camera drivers.
About PreciseVision	Generates a popup window that displays the PreciseVision version and ID information along with build information for key components of this application package.

Icon	Tool Tip Title	Description
	Create new Vision Project	A new empty un-named Vision Project is created by deleting all of the tools and existing Vision Processes. If the current Project has been changed and needs to be saved to the disk, a pop-up is first presented that permits the operator to save the modifications.
	Save current Vision Project to disk	Saves the loaded Vision Project back into its disk file. If the current project has never been saved to the disk, a file pop-up window is displayed to allow the operator to browse to the desired disk folder and assign a disk file name.

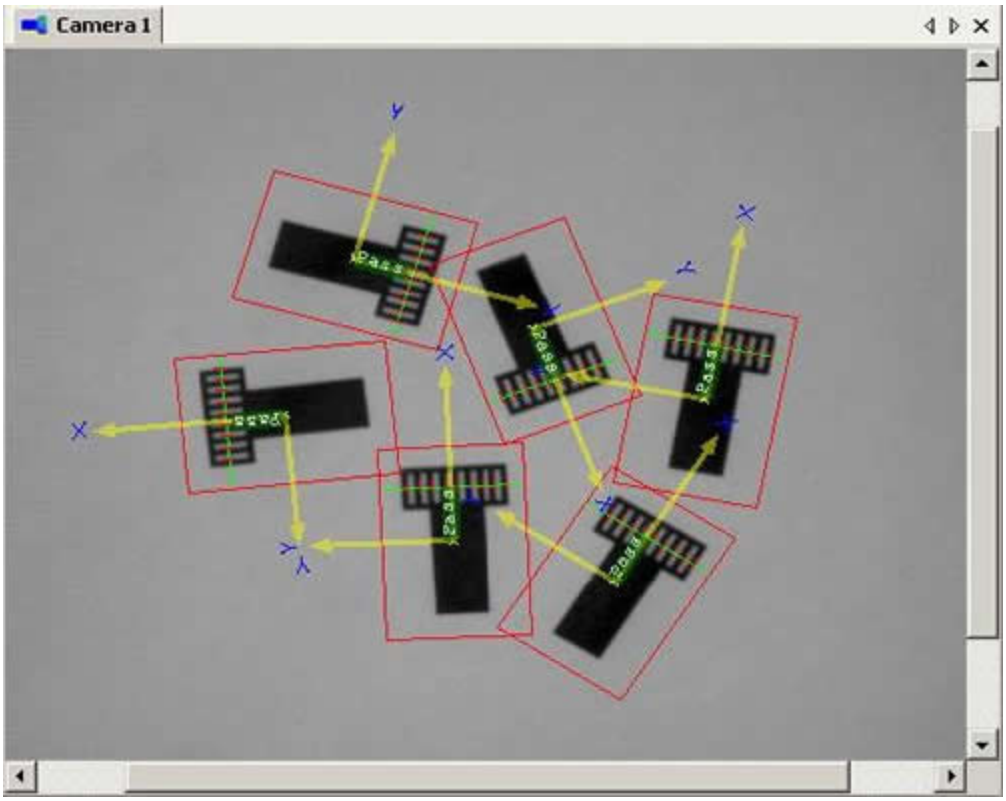
	Open a Vision Project	Opens a new Vision Project from the disk and loads it into memory, replacing the currently loaded Project. The standard file pop-up window is displayed to allow browsing to the desired folder. PreciseVision project files have a ".PVP" file extension.
	Display live video camera image	Places the currently selected camera into "Live Video" mode. Pictures will be taken continuously and displayed in the Camera Display window. This mode is very convenient for setting up the focus, f-stop, gain, and offset of a camera. If the camera is already in live video mode, pressing this button will terminate this mode.
	Take a single snap shot	Takes a single snap shot (i.e. picture) using the currently selected camera and displays the results in the Camera Display window.
	Toggle Color Display Mode	For color cameras only, toggles the Camera Display window display between color and grayscale mode. Set to color display mode by default.
	Zoom the Camera Display window	Zooms the Camera Display window view in or out. Pressing the button zooms in. A pulldown menu tied to this button zooms out or resets the window to its un-zoomed state. If "Zoom Auto" is selected, the active vision window is automatically scaled so as to best fit within the display window. The zoom factors range from 0.1 to 5 in steps of 0.1.
	Refresh Camera Display window graphics	Redraws any graphics generated by vision tools in the Camera Display window. This is convenient because the graphics can sometimes be lost when PreciseVision is minimized or if you click on certain panels.
	Switch display layout	Switches between three predefined display layouts: Edit, Run and Calibration. This icon makes changing the display layout very easy.
	Lock / Unlock Application	Locking an application prohibits users from modifying the loaded project. This is useful for avoiding situations where a user might inadvertently change a parameter or tool and cause a vision project to no longer operate correctly. Locking dims out all menu items, removes most toolbar items, and inhibits closing PreciseVision. Unlocking requires a password if a password has been defined in the Preferences panel.

Vision Camera Display

The Camera Display window shows the captured image from a selected camera with graphics to illustrate applicable vision tools. The camera image can be a "live video" display, which means that the captured

image is continuously updated, or a single snap shot. The vision tool graphics illustrate the placement of tools and their computed results. When a tool is selected for editing, the graphics also include handles that permit the tool to be easily re-positioned, re-oriented and re-sized using a mouse.

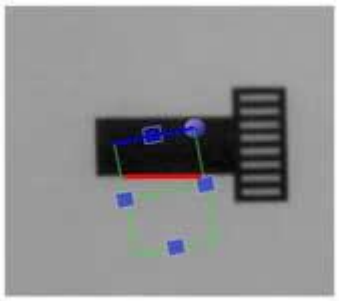
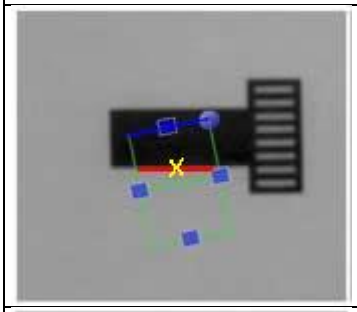
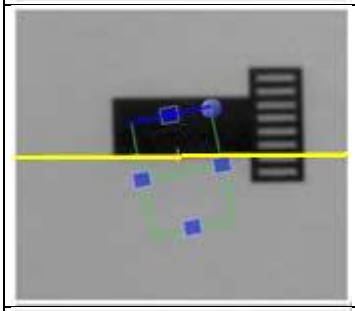
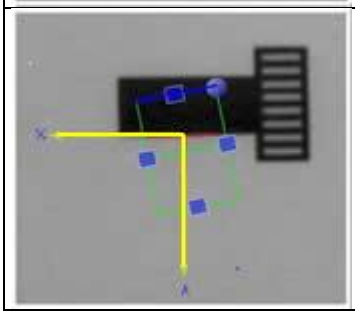
The following picture is a typical camera image with overlaid tool graphics. For all cameras, the X-axis of the camera coordinate system points from left to right and the Y-axis points from the bottom of the display to the top. The origin (0,0) of the camera coordinate system is in the lower-left corner of the camera image. If the camera is not calibrated, X/Y positions in the camera image are returned in units of raw pixels relative to the lower-left corner of the camera image. If the camera is calibration, X/Y positions are typically in units of millimeters relative to the origin.



To accommodate large or small images (those with many or few pixels), the camera display can be zoomed out or zoomed in. The control for this operation is provided on the Main Toolbar along with icons for manually placing a camera into live video mode and taking a snapshot. The Main Menu contains other commands for changing the selected camera and adjusting their gain and offset.

The following images describe how to utilize the graphical handles to place and size a typical vision tool and options for controlling how the results of a tool are illustrated in the Camera Display Window.

Graphical Display	Description

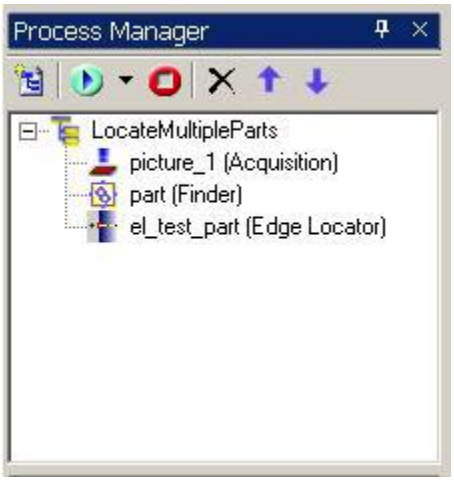
	This is a snap shot of a Line Fitter Tool placed over one edge of a part. The green and blue perimeter lines indicate the tool's search area. The blue edge marks the side of the search region that should be positioned on the dark side of the edge. The red line illustrates the results of the tool and marks the best fit line. To position the tool, click anywhere in the search region and drag the rectangle. To resize the tool, click one of the four blue squares and drag. To rotate the search region, click the blue circle and drag.
	If the output of a tool is to be utilized as an input (X, Y) point for another tool, it is informative to display the result as a single position. In this picture, the **ShowResults** property for the Line Fitter has been set to "POINT". This places a yellow "X" at the origin of the output of the Line Fitter. Note, setting **ShowResults** only affects the graphical display and does not alter the results computed by the tool.
	In this example, if the output of the Line Fitter is to be interpreted as a line, the **ShowResults** property for the Line Fitter can be set to "LINE".
	In this final example, the output of the Line Fitter is to be interpreted as a reference frame, so the **ShowResults** property for the Line Fitter has been set to "FRAME".

Process Manager Window

The Process Manager Window displays the list of all of the Vision Processes that are contained in the currently loaded Vision Project. If a Vision Process is expanded, all of the Vision Tools for the Process are displayed in the order in which they are evaluated. This window is the primary means for managing Vision Processes including their creation, renaming, deletion, modification and test execution. Any

Process that is displayed in this window is eligible for execution by a GPL procedure running on a Precise Guidance Controller.

A typical Process Manager Window is shown below.



In this example, a single Vision Process "LocateMultipleParts" is contained in the currently loaded Vision Project. This Vision Process has three tools. The Process begins with the standard Acquisition Tool that takes a picture. After the picture is captured, the system evaluates a Finder Tool named "part". The final operation is an Edge Locator "el_test_part".

It should be noted that if the Edge Locator is defined relative to the Results of the Finder Tool, the Process could have been written without the explicit execution of the Finder. When the system encounters the Edge Locator, it would automatically execute the Finder if the Finder had not been evaluated. The decision of whether to include the Finder in the Vision Process is entirely a matter of personal preference. Whether or not the Finder is included, the Results of the Process would be identical as well as the execution time.

To add a new Vision Process to a Project, click the toolbar icon to "Add a new process". Once the new Process is assigned a unique name, it will appear as a top-level (i.e. left-most) item in the window. To specify the sequence of tools to be executed in the process, simply drag Vision Tools from the Vision Tool Definition Window and drop them on the Process. New tools are automatically added to the end of the Process and can be re-ordered by using the up and down arrows in the toolbar. The order of appearance of the tools is important since the tools are evaluated top to bottom in the displayed list.

After a Vision Process has been created, it can be renamed by right-clicking on its name within the Process Manager window and selecting "Rename Process."

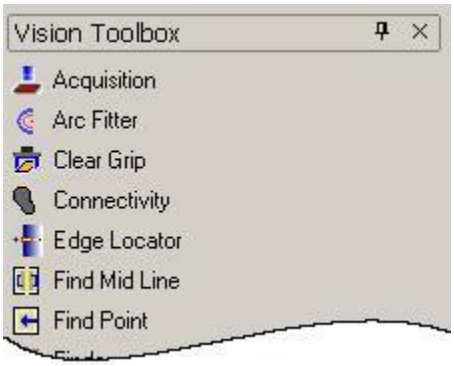
Once a Vision Process has been created, it can be triggered from GPL or can be manually tested using an icon on the toolbar. The manual test can be executed a single time or the process can be executed continuously. The continuous test mode is intended for demonstration purposes since the execution of the Process will not be synchronized with the GPL robot program and therefore not of very much practical use.

The following tables provide additional information on each of the window toolbar options.

Icon	Tool Tip Title	Description
	Add new process	Creates a new Vision Process and adds it to the Process Manager Window after it has been assigned a unique name via a pop-up window. The Process names are very important since they are used by GPL procedures to initiate the execution of a Process.
	Test selected process	Test executes the selected Process once or continuously. This is useful for debugging a Process and examining its computed results. However, in a robot guidance application, Processes are always initiated from a GPL procedure to ensure that the picture taking and evaluation are coordinated with the robot motions.
	Stop test loop	If a process is running in a continuous test mode, this button will stop the Process. This is only used while manually test executing a Process.
	Remove selected item	Removes a Process from a Project or a tool from a Process. In both cases, any referenced tools are not deleted from the Project, only the reference to the tool is removed. To delete a tool, you must use the Vision Tool Definition Window.
	Move tool up	Moves a vision tool up or down one line in a Vision Process list. The order in which the tools are listed in a Process is important because this is the order in which the tools are evaluated. If a Vision Process is selected, these button will change the order of the Processes within this window. This is a convenience feature and does not alter how Vision Processes are executed.
	Move tool down	

Vision Toolbox Window

The Vision Toolbox Window displays a list of all of the vision tool types that can be employed to construct a Vision Process. The top of this list currently looks as shown below.

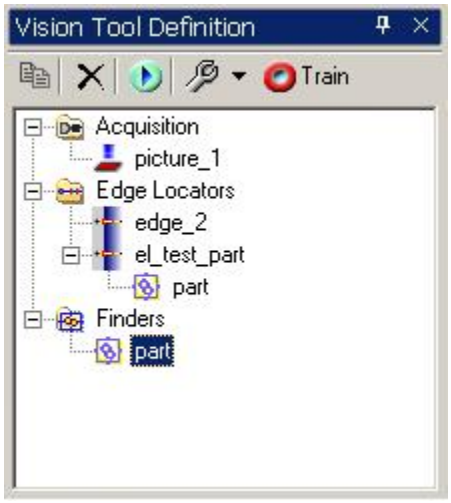


To add a new tool to an application, double click on a tool type in this list or right-click and select ***"Create Tool"***. A pop-up will be displayed for assigning a unique name to the new copy of the tool. Once a name has been assigned, the new tool will appear in the Vision Tool Definition Window. Once there, the tool can be edited or dragged into the Process Manager to add the execution of the tool to a specific Vision Process.

Vision Tool Definition Window

The Vision Tool Definition Window displays all of the tools that have been created in the currently loaded Vision Project, even those that are not referenced in any Vision Process. This window provides the primary means for adding tools to a Vision Process and for selecting tools to be edited. As a convenience, this window graphically indicates when tools are dependent upon the results of other tools.

A typical Vision Tool Definition Window will appear as shown below.



To simplify accessing tools, all of the defined tools are grouped according to their tool type and only tool types that have one or more tools defined are visible. In the example above, there is one Acquisition Tool

named "picture_1", two Edge Locators named "edge_2" and "el_test_part" and one Finder Tool named "part". The second Edge Locator has been defined with respect to the results of the Finder, so the Finder "part" appears below "el_test_part" to indicate this relationship.

There are a number of ways that a new tool can be added to the Vision Project and the Vision Tool Definition Window:

- Double click on a tool type in the Vision Toolbox Window.
- Right-click a tool or tool type in the Vision Tool Definition Window and select "New Vision Tool".
- Drag a tool type from the Vision Tool Definition Window and drop it into the Camera Display Window.
- Right-click in the Camera Display Window and select "New Vision Tool".

To copy an existing tool while maintaining its dependencies to other tools, highlight the tool in this window and click the "Copy" icon in the toolbar or drag the existing tool and drop it into the Camera Display Window.

To edit a tool, simply click on the name of the tool. This will automatically display the properties of the tool in the Property Display Window and will highlight the editing graphics and the handles for the tool within the Camera Display Window.

To test the operation of a tool without executing a Vision Process, click on the tool and click the "Play" button in the toolbar. This only executes the selected tool and any tools on which it is dependent. If the tool on which it is dependent returns multiple results, e.g. a Finder, only the first set of results is processed by the tool being executed. This is done to permit you to easily reposition the executed tool.

The following tables provide additional information on each of the window toolbar options.

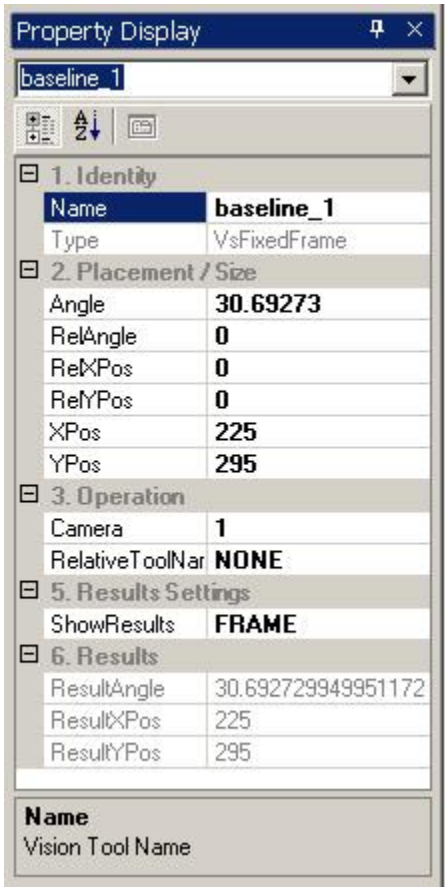
Icon	Tool Tip Title	Description
	Create copy	Copies the selected tool along with its relationships to other tools.
	Remove selected tool	Deletes the selected tool after confirmation but does not effect any tools that it references.
	Execute selected tool	Executes the selected tool and all of the tools on which it is dependent without executing a Vision Process. See above for further information.
	Finder operation mode	For Finder Tools, switches between "Find" and "Train" mode. Can also launch the Training Wizard to assist in teaching a Finder Tool and can display a trained Finder template at various resolutions as a diagnostic aid.
	Perform training operation	For tools that require training, executes the training process. In the case of the Finder Tool, this records the Finder Template and performs the analysis to determine distinctive features and the maximum levels of the search pyramid.

Property List Display

In PreciseVision, each Vision Tool is internally represented as a software "Object". The parameters for each tool are referred to as its "Properties". The properties include both the values that specify how a tool operates (e.g. its tool type, tool name, position, size, operating characteristics, etc.) as well as the values that indicate the results of executing the tool (e.g. number of objects found, number of edges located, etc.). For a given type of Vision Tool, the list of properties is always the same, but the list will vary from one type of tool to the next.

Some Vision Tools provide special buttons that are displayed between the tool name and the property grid. These buttons provide access to tool specific operations during the editing process (e.g. automatically snapping a tool into position).

The Property List Display provides a very convenient means for both viewing and editing the property values of a tool. As an example, the Property List Display for a typical Fixed Frame Tool is shown below.



The pull-down list at the top of the window selects the tool to be displayed. The displayed tool will also be automatically changed whenever a new tool is created or a tool is selected in the Vision Tool Definition Window.

Once a tool has been selected, any of its writable values can be modified by clicking on a cell and entering a new value. When a property is selected, a short description of the property is displayed at the bottom of the window. Properties that are readonly are indicated by dimmed text.

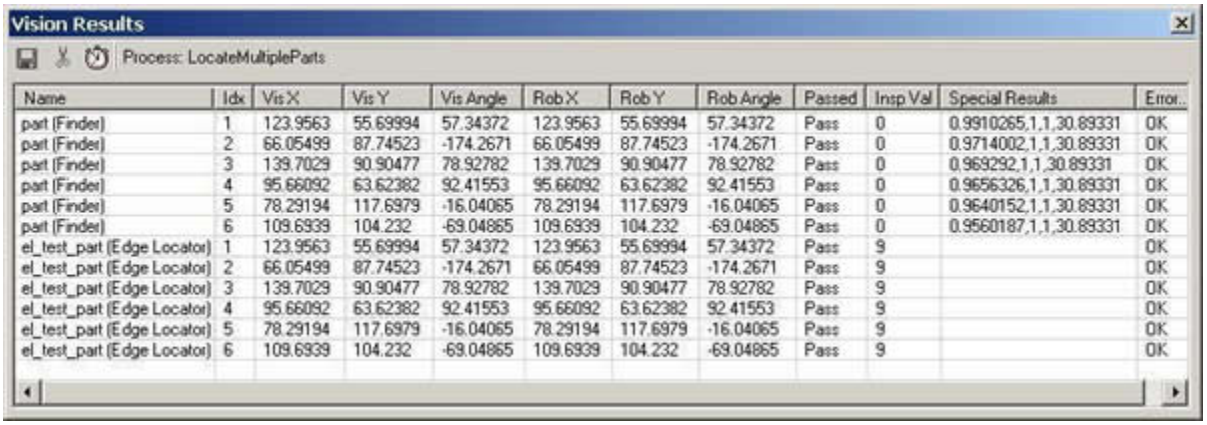
To assist in accessing these parameters, the properties are group according to their use. A description of each of the groups is presented in the following table. For detailed information on the properties for each of the tool types please see the chapter on the "Vision ToolKit".

Property Group	Description
1. Identity Properties	Specifies the name of the tool and its basic tool type.
2. Placement / Size Properties	Defines the position, orientation and size of the tool in the coordinate system of the camera. In addition, if the tool is defined relative to another tool, the relative position and orientation can also be specified. Normally, it is easiest to graphically position and size each tool by dragging on the tool's handles displayed in the Camera Display Window.
3. Operation Properties	Defines parameters that control the execution of the tool. For example, a tool might have an Operation Property that defines if it should screen out black-to-white edges or the gray scale levels above which pixels are processed. Only the commonly used Operation Properties are placed in this group.
3A. Advanced Operation Properties	Contains Operation Properties that are not normally modified and generally require a bit more experience to utilize. For these properties, their default settings will yield good tool performance for typical applications.
4. Inspection Settings Properties	Permits a single scalar result of the vision tool to be automatically tested (inspected) to verify that it is within limits. A Pass / Fail property is set accordingly and a Pass / Fail label can be optionally displayed on the tool in the Camera Display Window.
5. Results Settings Properties	Controls the results that are computed by the vision tool. For example, if multiple sets of results can be computed by the tool, these properties can limit the number of sets of data computed and can select the set of data that is displayed in the **Results Properties**.
6. Results Properties	Displays a single set of results for the vision tool. In general, the results returned varies from tool to tool. Many tools return a vision reference frame. However, some tools return additional results items such as statistical information on how well the answer fits the actual data.

Vision Results Display

The Vision Results Window displays the "Results" properties for all of the tools evaluated in the previously executed Vision Process. For position and orientation data, the results are displayed in both the vision coordinate system and the robot coordinate system utilizing the applicable camera calibration data. The information contained in this display is automatically updated each time a Vision Process completes execution.

A typical display that illustrates the results of a Vision Process that contains both a Finder and a Edge Locator Tool is shown in the following picture. For tools that return multiple sets of results (e.g. the Finder or Connectivity), *if you click on a line in the Results display, the corresponding object will be highlighted with a blue coordinate frame in the Camera Display*.



Name	Idx	Vis X	Vis Y	Vis Angle	Rob X	Rob Y	Rob Angle	Passed	Insp Val	Special Results	Error
part (Finder)	1	123.9563	55.69994	57.34372	123.9563	55.69994	57.34372	Pass	0	0.9910265,1,1,30.89331	OK
part (Finder)	2	66.05499	87.74523	-174.2671	66.05499	87.74523	-174.2671	Pass	0	0.9714002,1,1,30.89331	OK
part (Finder)	3	139.7029	90.90477	78.92782	139.7029	90.90477	78.92782	Pass	0	0.969292,1,1,30.89331	OK
part (Finder)	4	95.66092	63.62382	92.41553	95.66092	63.62382	92.41553	Pass	0	0.9656326,1,1,30.89331	OK
part (Finder)	5	78.29194	117.6979	-16.04065	78.29194	117.6979	-16.04065	Pass	0	0.9640152,1,1,30.89331	OK
part (Finder)	6	109.6939	104.232	-69.04865	109.6939	104.232	-69.04865	Pass	0	0.9560187,1,1,30.89331	OK
el_test_part (Edge Locator)	1	123.9563	55.69994	57.34372	123.9563	55.69994	57.34372	Pass	9		OK
el_test_part (Edge Locator)	2	66.05499	87.74523	-174.2671	66.05499	87.74523	-174.2671	Pass	9		OK
el_test_part (Edge Locator)	3	139.7029	90.90477	78.92782	139.7029	90.90477	78.92782	Pass	9		OK
el_test_part (Edge Locator)	4	95.66092	63.62382	92.41553	95.66092	63.62382	92.41553	Pass	9		OK
el_test_part (Edge Locator)	5	78.29194	117.6979	-16.04065	78.29194	117.6979	-16.04065	Pass	9		OK
el_test_part (Edge Locator)	6	109.6939	104.232	-69.04865	109.6939	104.232	-69.04865	Pass	9		OK

The following tables provide additional information on the window toolbar options and the data provided in each column of the results table.

Icon	Tool Tip Title	Description
	Save	Stores the results data in a comma delimited file (*.CSV) that can be read into a spreadsheet application for further processing. A standard popup file window is displayed to allow the file name and disk directory path to be specified.
	Copy to clipboard	Copies the contents of the results table to the system clipboard in a comma delimited (*.CSV) format. This information can then be pasted into another application.
	Timer	Displays the execution time for each top-level tool in the previously executed vision process in place of the tool results. The incremental time for each tool is displayed as well as the cumulative execution time for all tools in the process, in milliseconds.

Data Column	Description
Name	Specifies the name of the vision tool that generated the results.
Idx	For tools that return multiple sets of results, this is the index for the set of results that is displayed on a line. The index value goes from 1-n.
Vis X	For tools that return a position and orientation, this is the X & Y position in millimeters and the angle in degrees. These values are in the coordinate system of the camera's field-of-view.
Vis Y	
Vis Angle	
Rob X	For tools that return a position and orientation, this is the X & Y position in millimeters and the angle in degrees. These values are in the robot's world coordinate system and are the values that are transferred to a GPL program in a vision guidance application.
Rob Y	
Rob Angle	
Passed	Pass / Fail results of the tool Inspection. This is the same value as the **InspectPassed** property.
Insp Val	Actual property value that was tested by the tool Inspection. This is the same value as the **InspectActual** property.
Special Results	Each vision tool may have additional results that are unique to the tool type. For example, the Connectivity Tool returns the area of located blobs (**ResultBlobArea**). The Special Results column provides a place to display up to 20 tool specific results per vision tool. These special results are the same values that can be retrieved by a GPL program using the *VisResults.Info* method and are indicated in the documentation for each tool by *[GPL: VisResult.Info(x)].* When a Connectivity Tool is executed, the blob area can be fetched as *VisResult.Info(3)* and will be displayed as the 3rd number in the Special Results column.
Errors and Warnings	Specifies if an error or warning condition occurred when the tool was executed. This is the same value as the **ResultErrorCode** property.

Application Status Bar

The bottom of the PreciseVision application window contains the Status Bar. This area continuously displays status information on the execution of the application. The Status Bar typically resembles the following.

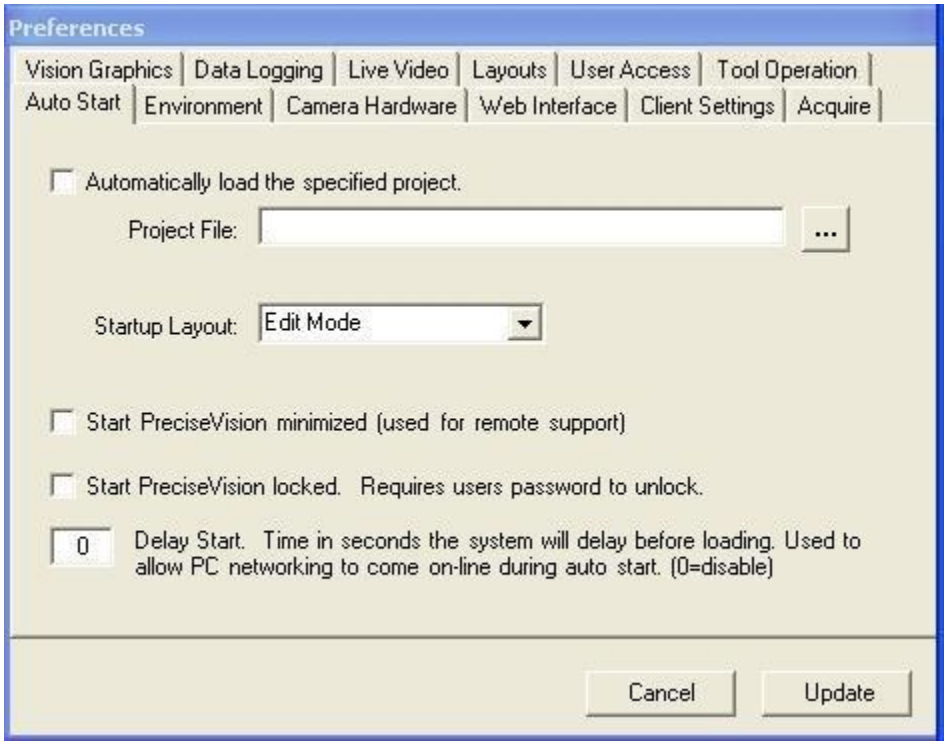
The information contained in each of the columns of the Status Bar is described in the following table.

Col #	Data	Description
1	Server status	Indicates Ethernet traffic between PreciseVision and a Guidance Controller. Each time a message is received, the icon will flicker.
2	Image status	Indicates if the selected camera is in "live video" mode (bright light bulb) or a single still image has been captured.
3	Camera calibration file	Displays the name of the camera calibration file for the currently selected camera.
4	Camera calibration type	Displays the type of camera calibration that is in effect for the currently selected camera, e.g. Camera Only, Robot Vision, etc.
5	Process execution time	Indicates the total time required to execute the last Vision Process in seconds.
7	Camera coordinates and intensity reading	Shows the X/Y position of the mouse cursor in calibrated camera coordinates and the gray-level intensity value of the associated camera image pixel. This is useful for setting threshold and edge level values.
8	Project modified flag	Indicates if the currently loaded Vision Project has been modified in any way and must be saved to the disk in order to preserve the changes.

Preferences Panel

The Preferences Panel contains a number of application setup parameters that customize the operation and appearance of PreciseVision. Once saved, these preferences are automatically reloaded and put into effect each time the application is restarted.

The Preference Panels appears as shown below.



When any data in the Preferences has been modified, the ***"Update"*** button must be pressed before the change will take effect and be preserved.

In the following table, the configuration information presented on each tab of this panel is briefly described.

Tab	Description
Acquire	Defines the default disk file path for storing camera images when an Acquisition Tool is executed. This is a debugging and diagnostics convenience for logging images.
Auto Start	Specifies a Vision Project that will be automatically loaded each time PreciseVision is restarted. Once a Project is loaded, its Vision Processes are immediately available for execution by a GPL procedure executing on a Guidance Controller. This tab also contains the 'Startup Layout' specification. The specified layout will be automatically loaded each time PreciseVision is started. For production systems this would normally be set to the 'Run Mode' layout. Options are also provided to start up PreciseVision minimized and / or in a locked mode that prevents operators from

	making changes to the loaded processes. To set the lock password, see the User Access tab. A parameter is provided that will delay PreciseVision from starting execution for a specified number of seconds. This is beneficial if PreciseVision is automatically started when the PC boots up, but the PC requires time to initialized its network cards and camera interfaces.
Camera Hardware	Specifies the type of camera hardware to be used to capture images, e.g. Ethernet or USB, and defines each camera's logical number. Some of the data specified on this panel and its sub-panels are specific to the type of camera being employed. Normally, these parameters are defined during the software and camera installation process. For more information on this panel, please see the Camera Installation sections of this manual for detailed information on configuring each type of supported camera.
Client Settings	Defines the Ethernet IP address for the Guidance Controller that is accessed during the camera calibration process. For normal operations, the controller IP address is automatically deduced from the commands that GPL sends to PreciseVision.
Data Logging	Enables the logging of Vision Tool Results to a disk log file for diagnostic purposes. Each time a tool is executed, its Results are time stamped and stored in a specified file in a comma delimited text format (*.csv). This file can be easily imported to Excel or another spreadsheet application for analysis. Also provides a checkbox to enable logging of bi-directional communications between PreciseVision and Guidance Controllers and Remote PC Servers. *Normally, this should only be enabled for debugging purposes since it consumes resources.*
Environment	Controls whether the Main Toolbar and the Status Bar are displayed. Also permits adjustment of some diagnostic and demonstration parameters. In addition, a checkbox is provided to permit PreciseVision to be minimized to an icon in the System Tray as opposed to an item on the Windows Start Bar. An "Auto Zoom" feature is also provided that automatically scales the active camera image to best fit within the camera display window.
Layouts	Defines three user-interface layouts of the dockable panels. These layouts provide an easy way to quickly switch between several display formats that are appropriate for different operating needs: production systems, editing sessions and calibration processes.

	If the "Layout Filename" is left blank, PreciseVision will automatically use a predefined default layout stored in the "Layout" folder of the installation directory. If the field contains a valid layout file, it will override these defaults. Click the "Current" button in each selection to record the size of the window (width and height) automatically.
Live Video	Controls the rate at which the camera image is updated in Live Video mode. This permits the Ethernet network traffic to be balanced when using PreciseVision. The update rate ranges from 1 to 30 frames per second (fps) with a default setting of 10. For cameras interfaced at 100Mb Ethernet, a rate of 1 to 15 fps is typical. For cameras interfaced at 1000Mb Ethernet, the refresh rate can be as high as 30 fps. This mode is NOT available for USB cameras. The frame rate is set to the maximum defined by the camera. Some USB cameras do not operate properly if the live frame rate is set to certain values.
Tool Operation	Enables and disables the **ResultOnNotFound** property for all required tools. This is typically set if an **Inspect List Tool** is defined that analyzes the output of a tool that generates multiple results, such as a **Finder**, and the **Inspect List** includes multiple inspection operations. Enabling this property forces all tools to generate results even if their inspection tests fail. This is important for generating complete, consistent sets of results for the **Inspect List** to test. This property is normally disabled since most vision processes prefer to ignore any object that fails an inspection test This property is disabled by default.
User Access	Defines the password used to unlock access to PreciseVision.
Vision Graphics	Enables and disables the display of Vision Tools and their Results in the Camera Display Window. Drawing these graphics has relatively little impact on the performance of the system, so the graphics are normally displayed.
Web Interface	Enables up to 3 web panels to be configured into the PV user interface as dockable controls. These panels can be utilized to integrate web pages of the Guidance Controller into the PV environment. Either the complete path for each web must be specified, e.g. "http://192.168.0.1/ROMDISK/web/Opr/master/masterfs.html" or just the path with the Client Guidance Controller without the Client's IP address, e.g. 'ROMDISK/web/Opr/master/masterfs.html'. In the latter case, the IP address of the Client Guidance Controller will be automatically assumed when displaying the panel.

Vision Toolkit Descriptions

Vision Toolkit Introduction

This chapter provides detailed information on each of the tools in the PreciseVision vision toolkit. These tools perform all of the runtime operations that are needed to acquire and process an image captured from a camera.

Each instance of a tool is a vision object with a graphical front-end, a property list for configuring the tool for your desired operation, and a read-only list of results properties that can be used by other vision objects to create more complex machine vision and logical operations. The graphical front-ends allow the vision tools to be easily positioned over a captured image and greatly simplifies their training.

ToolKit Summary

The following table briefly summarizes the primary function of each of the vision tools. The "Type" broadly categories each tool as to whether it "Acquires" an image, "Finds" an object, "Measures" features, "Inspects" a region, "Computes" additional information, "Filters" image or object data or serves as a "Motion Sensor". Please see the detailed descriptions on the following pages for more ways in which these tools can be utilized.

Vision Tool	Type	Description
Acquisition Tool	Acquires	Captures an image from a camera or loads it from a disk file, and stores the image into a frame buffer.
Arc Fitter Tool	Measures	Searches a specified region for arc edges and returns the arc that best fits the edges. Returns both the arc center point and radius.
Area-Of-Interest (AOI) Tool	Copies	Copies a rectangular area-to-interest to a disk file, another region of the current image buffer, or another image buffer. Permits mathematical manipulation when copying to an image buffer.
Barcode Reader Tool	Inspects	Reads a variety of standard 1-D and 2-D barcodes and returns the barcode type and the value of the barcode.
Clear Grip Tool	Inspects	Verifies that there is no obstruction within the bounds of a defined window. Typically used to confirm that gripping a part will not result in a collision.

Connectivity Tool	Finds	Searches a region for binary blobs and returns the centroid and other features of each blob. In addition, this tool can return data that defines the perimeter of any located blob.
Edge Locator Tool	Measures	Detects edge points with sub-pixel accuracy along a linear path in a vision image and returns their positions. Alternately, this tool can return an array of the intensities along the linear path or a histogram of their values.
Find Centerline Tool	Measures	Searches a specified region for edge points and returns the centerline between two located bounding lines.
Find Point Tool	Measures	Locates the edge point that is closest to the line that defines the start of the tool's search region. The XY position of the point is returned with sub-pixel accuracy along (and pixel accuracy perpendicular to) the search direction.
Finder Tool	Finds	Most powerful vision tool that identifies randomly placed parts in a camera image and returns their position, orientation and scaling to within sub-pixel accuracy.
Fixed Frame Tool	Computes	Places a reference frame at a fixed image coordinate or at a constant offset relative to another vision object. Can optionally index the frame in an X and/or Y grid pattern to repeatedly execute any linked tools.
Image Process Tool	Enhances image	Performs image filtering, edge extraction, thresholding and morphological operations on a specified area in the camera image.
Inspect Frame Tool	Computes	Passes through the XPos, YPos and Angle Results of a referenced tool and optionally rotates and shifts the Results based upon an inspection. The inspection can consist of the arithmetic combination of the Results of two other tools. So, this tool also provides a means for performing simple mathematical operations on the results of two tools.
Inspect List Tool	Computes	Passes through the XPos, YPos and Angle Results of a referenced tool and tests up to 12 different Results from multiple tools to determine if the object is good or bad. This tool provides a convenient way to perform complex testing of a located object and generating an equivalent location with a single Pass/Fail indicator.
Inspect Region Tool	Computes	Inspects the perimeter and the interior of an opaque object to determine if there are any flaws in the object's outline or holes in the interior. This tool was designed to inspect wafers, where slight changes in the overall size and shape of the wafer are permitted, but chips

		in the perimeter or interior holes or cracks are unacceptable.
Light Recorder Tool	Detects light pattern	Acquires a sequence of camera images for a specified period of time, determines the brightness of each image, and returns the brightness readings.
Line Fitter Tool	Measures	Searches a specified region for edge points and returns the line that best fits the edges.
Line-Line Frame Tool	Computes	Determines the intersection of two lines defined by vision objects and returns a reference frame. Alternately, computes and returns the midpoint of two reference frames defined by two vision objects.
Pixel Window Tool	Inspects	Counts edges or binary pixels or collects gray-scale statistics within a rotated rectangular or circular region. Used for quickly detecting the presence of features or collecting general intensity information about a region.
Pixel Color Window Tool	Inspects	Tests if the general RGB color of a rectangular or circular region matches a trained color within a specified tolerance. Also computes the mean HSV/HSI values for the region.
Point-Line Frame Tool	Computes	Takes a point and a line from two vision objects and returns a reference frame.
Point-Point Line Tool	Computes	Computes a line given two points from vision objects.
Sensor Window Tool	Detects motion	Detects motion in a window region. Ensures the scene is stable prior to other operations or waits until motion is detected. Can be used in place of physical sensors in flexible parts feeding where parts can take time to settle.
Text Label Tool	Labels	Displays string constants plus inspection and results data from a referenced tool. Text is shown relative to the position of the referenced tool.
Tool Filter Tool	Filters	Takes the output from another tool that generates multiple sets of results, and returns a subset of the results based upon specified criteria.

Standard Tool Properties

To simplify their application, as much as practical, the vision tools make use of a common group of property names and associated operations. The following tables describe the standard tool properties that are common to many of the tools.

In these tables and the detailed tool descriptions, properties that are "read-only" have their name and data type dimmed. In addition, properties that have a counterpart

property or method within the Guidance Programming Language include the GPL member name underlined and in italics bounded by square brackets preceded by "GPL:" and the class name, e.g. "*[GPL: Vision.ResultCount]*".

The **Identity Properties** specify the name of the tool and its basic tool type.

1. Standard Identity Properties			
Property Name	**Data Type**	**Range**	**Description**
Name	**String**	n/a	This is the unique name that must be assigned to each instance (copy) of a general tool type. The name must follow the GPL variable naming conventions. That is, a name can be mixed case (upper and lower case characters) but when referenced by other tools, the names are not considered case sensitive (i.e. Abc and aBc refer to the same tool). Names must start with either a letter or an underscore "_" and can be followed by up to 127 additional letters, numbers and underscores. *In GPL, this is the name by which a tool is referenced.*
Type	**String**	n/a	This read-only property displays the basic type for the tool, e.g. VsLineFitter.

The **Placement / Size Properties** define the position, orientation and size of the tool in the coordinate system of the camera. In addition, if the tool is defined relative to another tool, the relative position and orientation can also be specified. Normally, it is easiest to graphically position and size each tool by dragging on the tool's handles displayed in the Camera Display Window. In the range specifications, "AOI" is an abbreviation for "Area of Interest" and refers to the size of the portion of the camera image that is being analyzed.

2. Standard Placement/Size Properties			
Property Name	**Data Type**	**Range**	**Description**
Angle	**Single**	-360 to 360	These values specify the orientation (in degrees) and the position (in calibrated units, mm) of the center of the vision tool in the coordinates of the vision image. These values are automatically updated when the tool is graphically re-positioned and re-oriented during training and if the vision tool is placed relative to another vision tool during runtime.
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	

Height	**Single**	0 - AOI (mm)	These values define the height and width of the vision tool in calibrated units (mm). These values are automatically updated when the tool is adjusted with the mouse or can be manually entered for more precise adjustments.
Width	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	These values specify the orientation (in degrees) and the position (in calibrated units, mm) of the center of the vision tool relative to the vision object specified by **RelativeToolName**. If the vision tool is not relative to another vision object, these values are zero.
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	

The **Operation Properties** define parameters that control the execution of the tool. For example, a tool might have an Operation Property that defines if it should screen out black-to-white edges or the gray scale levels above which pixels are processed. Only the commonly used Operation Properties are placed in this group.

3. Standard Operation Properties			
Property Name	**Data Type**	**Range**	**Description**
Camera	**Integer**	1 - 6	Defines the number of the camera (and its associated frame buffer) on which the tool operates.
Relative-ToolName	**List**	n/a	If a tool is to be positioned relative to another tool, this is the name of the referenced tool. Tools can be relative to any vision tool that returns a vision coordinate position. By default tools are not relative to another tool and this property is blank.

The **Advanced Operation Properties** contain Operation Properties that are not normally modified and generally require a bit more experience to utilize. For these properties, their default settings will yield good tool performance for typical applications.

3A. Standard Advanced Operation Properties			
Property Name	**Data Type**	**Range**	**Description**

The **Inspection Settings Properties** permits a single scalar result of the vision tool to be automatically tested (inspected) to verify that it is within limits. A Pass / Fail property is set accordingly and a Pass / Fail label can be optionally displayed on the tool in the Camera Display Window.

4. Standard Inspection Settings Properties			
Property Name	**Data Type**	**Range**	**Description**
InspectType	**List**		Specifies the single scalar result that is to be inspected. The items in the list vary from one tool to the next. Select "None" to disable inspecting a result.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	Indicates if a Pass / Fail label or a numeric value is to be displayed on the tool in the Camera Display Window based upon the results of the inspection. The items in the list indicate if no label should be displayed, only a Pass label, only a Fail label, either a Pass or Fail label, or a real or integer numeric value. The Pass indicator is displayed in green and the Fail is written in red. This selection is used for display purposes only and has no effect on the results of the inspection.
InspectMax	**Single**		Maximum and minimum values permitted for the item specified in **InspectType** to "Pass" the inspection. Values outside of this range are marked as "Failed".
InspectMin	**Single**		
InspectActual	**Single**		This is a read-only property that indicates the value of the result specified by **InspectType** that was tested during the inspection. *[GPL: VisResult.InspectActual]*
InspectPassed	**List**	Pass / Fail	Indicates if the inspection criteria was satisfied (Pass) or was not (Fail). Set to 'Pass' if no inspection is selected. If the inspection fails, any tool that references this tool, for example via the **RelativeToolName** property, will also have a failed InspectPassed although that tool will still be evaluated. *[GPL: VisResult.InspectPassed]*

The **Results Settings Properties** control the results that are computed by the vision tool. For example, if multiple sets of results can be computed by the tool, these properties can limit the number of sets of data computed and can select the set of data that is displayed in the **Results Properties**.

5. Standard Results Settings Properties

Property Name	Data Type	Range	Description
MaxResults	**Integer**	-1 or 1 to n	For tools that provide multiple sets of results, this property can limit the maximum number of sets of results that are computed and returned. A negative 1 (-1) indicates that all possible results should be collected.
OffsetAngle	**Single**	degrees	These Offset values permit the results of a vision tool to be shifted and rotated from their standard location. This allows you to re-position the results of a tool to a position and orientation that is more meaningful to you. The orientation change is in degrees and the shift is in calibrated units (mm). See the **Fixed Frame Tool** for an alternate method for implementing this offset.
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	If TRUE, indicates that the "Return results if not found" Preferences property is asserted. Normally, this property is FALSE and only tools that are successful will generate results and continue processing an object. This is the correct behavior if you only wish to process objects that pass all inspection tests. However, if a tool is referenced in an **Inspect List**, it is important that both successful and failed results be returned so that all of the inspections in the list are aligned and are with respect to the same located object. So, if an **Inspect List** is being utilized, this Preference property should be set to TRUE.
ResultSelect	**Integer**	1 - n	For tools that return more than one set of results, this property selects the set of results that is displayed in the **Results Properties**. This does not affect the actual results of the tool.
ShowResults	**List**	NONE / POINT / LINE / FRAME	Alters how the results of a tool is graphically displayed. Each vision tool has a default method for display, e.g. a line or a frame. This property allows the graphical display to be changed. For example, a Fixed Frame Tool is normally displayed as a

			reference frame. However, if it is being used as a datum line or a point, its display can be changed to a line or a point for visual clarity. This property does not affect the actual results of the tool.

The **Results Properties** display a single set of results for the vision tool. In general, the results returned varies from tool to tool. Many tools return a vision reference frame. However, some tools return additional results items such as statistical information on how well the answer fits the actual data. If the tool computes more than one set of results and **MaxResults** is set to greater than 1, the displayed set of results will correspond to the setting of the **ResultSelect** property.

6. Results Properties			
Property Name	**Data Type**	**Range**	**Description**
ResultCount	**Integer**	0 - n	Number of sets of results the vision tool has returned. *[GPL: Vision.ResultCount]*
ResultErrorCode	**Integer**		Standard error code that indicates if the tool executed without an error. This is different from the **InspectPassed**, which indicates if the result was computed properly but its value was inside or outside of a specified limit. If **ResultErrorCode** is non-zero, any tool that references this tool, for example via the **RelativeToolName** property, will also fail with this same error code and will not be evaluated. *[GPL: VisResult.ErrorCode]*
ResultAngle	**Single**	-360 to 360	Orientation (in degrees) and position (in calibrated units, mm) of the center of the results computed by the vision tool in the coordinates of the vision image taking into consideration the value of the **Offset**'s in the **Results Settings Properties**. For robot guidance applications, these values are transformed by the vision-to-robot calibration conversion routines and then presented as *[GPL: VisResult.Loc.X, VisResult.Loc.Y and VisResult.Loc.Roll]*.
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Acquisition Tool

Vision tool that captures an image from a camera or loads it from a disk file, and stores the image into a frame buffer.

Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
GainBlue	**Single**	0-12 (set to -1 if not a color camera)	These properties scale the intensity values for each of the three color layers as they are read from the camera. This permits the color intensities to be balanced.
GainGreen	**Single**		
GainRed	**Single**		
3A. Advanced Operation			
AcquireBuffer-Image	**Boolean**	True / False	If True, a copy of the image is stored into a secondary buffer when the image is acquired. This must be set True if the original image is to be restored by a subsequent Acquisition tool that has its **AcquireMode** set to "RESTORE_ORIGINAL". For resolutions greater than 640x480, setting this value to False, if not needed, can save a few milliseconds of execution time. Set False by default.
Acquire-Extension	**List**	BMP / PNG	If this tool is set to automatically save a camera image to a disk file, this property specifies whether the image should be saved as a non-compressed bitmap (BMP) or as a compressed image in a lossless format (PNG) or as a compressed image in a lossy format (JPG). The

			PNG and JPG file formats are smaller, but requires more CPU time to create.
AcquireFill-GrayLevel	**Integer**	0 - 255	If the **AcquireMode** is defined as "CLEAR_BUFFER", all of the pixels in the image buffer are set to the intensity value specified by this property.
AcquireMax-CurrentCount	**Integer**	0 - 99999	When the **AcquireMode** is set to ACQUIRE_AND_SAVE or SAVE_ONLY, the **AcquireMaxSaveImages** property defines the largest index that is appended to the **AcquirePrefix** to generate the name of the file in which the image is stored. If **AcquireMaxSaveImages** is -1, the next highest unused index values is assigned up to the maximum value of 99999. So, existing image files will not be over-written. If **AcquireMaxSaveImages** is set >0, **AcquireMaxCurrentCount** will indicate the next index. When the limit specified by **AcquireMaxSaveImages** is reached, **AcquireMaxCurrentCount** is automatically reset to 0. This permits the last **AcquireMaxSaveImages** files to be saved without consuming more and more disk space (but existing image files will be over-written). In this mode of operation, **AcquireMaxCurrentCount** can be manually changed or automatically altered by a robot control program to reset the next index to a specific value. The value of **AcquireMaxCurrentCount** is not stored with the vision project and is reset to 0 each time the vision project is loaded.
AcquireMax-SaveImages	**Integer**	-1 - 99999	
AcquireMode	**List**	NORMAL_ACQUIRE / ACQUIRE_AND_SAVE / CLEAR_BUFFER / PLAY_FROM_DISK / LIVE_VIDEO /	"NORMAL_ACQUIRE" - acquires a single image from the specified camera and places it into a frame buffer. "ACQUIRE_AND_SAVE" - performs

		SAVE_ONLY / RESTORE_ORIGINAL	the same operation as "NORMAL_ACQUIRE" and then saves the image to a disk file. "CLEAR_BUFFER" - Sets all of the pixels in a buffer to the intensity value specified by **AcquireFillGrayLevel**. This is useful before an AOI copy is performed. "PLAY_FROM_DISK" - restores an image from a disk file. The disk file is specified by the **AcquirePath** and **AcquirePrefix** (see below). If there are multiple numbered image files with the same prefix, each time the Vision Process is executed, the contents of the next sequential file will be copied to the image buffer. "LIVE_VIDEO" - continuously updates the vision display with the latest image from the camera. This is a system setup mode that facilitates setting the camera f-stop, focus, **VideoGain**, and **VideoOffset**. "SAVE_ONLY" - stores the contents of the current frame buffer to a disk file. This enables a GPL program to execute a vision process that saves selected frame buffers for later access. "RESTORE_ORIGINAL" - restores the contents of the current frame buffer to its original value at the time that the image was first captured. Requires that **AcquireBufferImage** is set to True when the image is first captured.
AcquirePath	**String**	n/a	String that defines the path to the disk file when **AcquireMode** is set to "ACQUIRE_AND_SAVE" or "PLAY_FROM_DISK"'. If blank (""), the path defined in the Preferences will be used. If the path does not contain a ':' (i.e. C:\...), **AcquirePath** will be appended to the path where the PreciseVision application is

			stored, e.g. "C:\Program Files\Precise Automation\PreciseVision #.#\"
AcquirePrefix	**String**	n/a	String that defines the disk file name (excluding the required .BMP extension) when **AcquireMode** is set to "'ACQUIRE_AND_SAVE" or "PLAY_FROM_DISK". If blank (""), the file name defined in the Preferences will be used. This string is combined with AcquirePath, a numerical index, and the ".BMP" file extension to generate the disk file name for saving or loading vision images.
ActiveLayer	**List**	Monochrome / RedLayer / GreenLayer / BlueLayer	Specifies the type of data to be loaded into the image frame buffer for processing by subsequent vision tools. For grayscale cameras, only "Monochrome" image buffers are available and each pixel in the frame buffer has a value from 0 to 255 that defines its grayscale intensity. For color cameras, the frame buffer can be loaded with one of four types of data: monochrome, red, green or blue. The monochrome mode produces the same results as a grayscale camera. The red, green and blue modes load the intensity data for a single color. For example, if "RedLayer" is selected, the image frame buffer will contain values that indicate the intensity (response) of only the red receptors of the camera. Tools that are applied to this image will only be operating on the red color layer of the camera. The one exception is the "Pixel Color Window" Tool. When this tool is executed, it automatically accesses all three of the RGB color layers of an image and is unaffected by the setting of the **ActiveLayer** property.
AoiAcquire	**Boolean**	True / False	For cameras (such as most IDS uEye USB models) that support acquiring
AoiHeight	**Single**	0 - FOV (mm)	

AoiWidth	**Single**	0 - FOV (mm)	only a portion of the full field of view, these parameters define the rectangular section of the image to be acquired. If a reduced size AOI is acquired, the camera acquisition time will be reduced. Different Acquire tools can have different sized and positioned AOI's. However, when you switch between different AOI's, there is typically a significant time delay. To acquire an AOI, set **AoiAcquire** True and define the height, width and position of the AOI. The AOI dimensions and position are specified in calibrated units (mm) like most tools instead of pixels.
AoiXpos	**Single**	0 - FOV (mm)	
AoiYpos	**Single**	0 - FOV (mm)	
FlatFieldEnabled	**Boolean**	True / False	If True, each captured image is automatically compensated for uneven background lighting by adding a sample image that was previously saved. The saved image is automatically inverted and normalized such that when it is added to a new image, a very light uniform (flat) background is produced. This can make it easier to distinguish objects when there are background lighting variations, especially if binary vision tools are utilized. See below for a description of the Special Feature Buttons for capturing the sample image.
ImageFile	**String**	n/a	Read-only string that displays the currently selected vision image file. This value is only displayed when the **AcquireMode** is set to "PLAY_FROM_DISK".
LoadFirstImage	**Boolean**	True / False	This is a convenience feature for setting the display of a sequence of images back to the first image and is only valid during the "PLAY_FROM_DISK" mode. That is, when this property is set to TRUE, the disk file that satisfies **AcquirePath** and **AcquirePrefix** and has the lowest index values will be loaded into the image buffer.
3B. Trigger			

TriggerActive	**List**	Low_to_High / High_to_Low	If **TriggerEnable** is True, this property defines if the picture is to be when the digital input signal transitions from low to high or high to low.
TriggerEnable	**Boolean**	True / False	If this property is True, the image capture is delayed until a digital input signal of the correct state is received by the camera.
TriggerTimeout	**Integer**	1 - 30	If **TriggerEnable** is True, this property specifies the time in seconds that the system will wait for the trigger to occur before an error is generated (-4019 Vision Process Failed).
4. USB Cameras			
BackLight Compensation	**Integer**		These properties change the operation of USB cameras. The properties that a camera supports and the range of values for each property will vary from one camera model to the next and the software driver used to access the camera. When a camera is attached by PreciseVision, its supported properties, the range of allowed values and the default values are automatically sampled. The range for each supported property is displayed by the "Adjust Video Properties" Special Features Pop-up. Special attention should be paid to the **Exposure** and **PixelClock** properties since these affect the time it takes to acquire an image. Please see the documentation provided with the camera for specific information on the setup of its supported properties.
Brightness	**Integer**		
ColorEnable	**Integer**		
Contrast	**Integer**		
Exposure	**Integer**		
ExposureTime	**Single**		
Focus	**Integer**		
Gain	**Integer**		
GainBoost	**Integer**		
Gamma	**Integer**		
Hue	**Integer**		
Offset	**Integer**		
PixelClock	**Integer**		
Saturation	**Integer**		
Sharpness	**Integer**		
WhiteBalance	**Integer**		
Zoom	**Integer**		

Remarks

This tool performs the basic image capture operation from a color or monochrome camera and stores the image in a frame buffer. Consequently, this tool is normally the first tool in each Vision Process.

Multiple cameras can be accessed by this tool and their gains and offsets can be controlled to optimize the brightness range for the field of view. To facilitate setting up a camera's gain, offset, focus, and f-stop, the camera image can also be continuously acquired and displayed in the Camera Display Window.

Normally, when this tool is executed, a camera image is immediately captured. However, if **TriggerEnable** is True, executing this tool primes an image capture, but the picture is not immediately taken. The image capture is delayed until a digital signal, which is directly connected to the camera, is asserted. For applications where the timing of the image capture relative to an external event is critical, trigger mode can significantly reduce latency and jitter. As a debugging convenience, if a Vision Process is manually executed using the PreciseVision GUI and the process contains an **Acquisition** that has triggering enabled, a pop-up is display to confirm if the camera should wait for an external trigger or if the picture should be taken immediately. This pop-up is not displayed if the Vision Process is remotely initiated from a Guidance Controller.

Camera triggering is not supported on all cameras. Please consult your camera's hardware documentation for specific information on whether it supports external triggering and how this is implemented.

Most vision tools analyze the data that is stored in a frame buffer by an Acquisition tool, generate results, and leave the frame buffer unmodified. However, if an "Image Process" tool is executed (e.g. a low pass filter), the data stored in the frame buffer is altered so that all subsequent tools operate on the modified image data. In these cases, it is sometimes convenient to restore the original contents of the frame buffer. This can be done by executing an Acquisition Tool with the **AcquireMode** set to "RESTORE_ORIGINAL". This operation restores the original image data from an in-memory buffer, and is therefore much faster than storing an image to a file and then reloading. In order for this mode to operate properly, the original Acquisition tool must have its **AcquireBufferImage** property set to True. When this property is true, a copy of the camera image is generated in memory when the acquisition is performed.

If the background lighting is not uniform, **FlatFieldEnabled** can be set to automatically add a previously saved inverted normalized image of the background lighting. This will produce a "flat" background that will help to highlight the features of parts within the field of view. Enabling this function adds about 1 msec of processing time to a 752x480 grayscale image. (See the Special Feature Buttons below.)

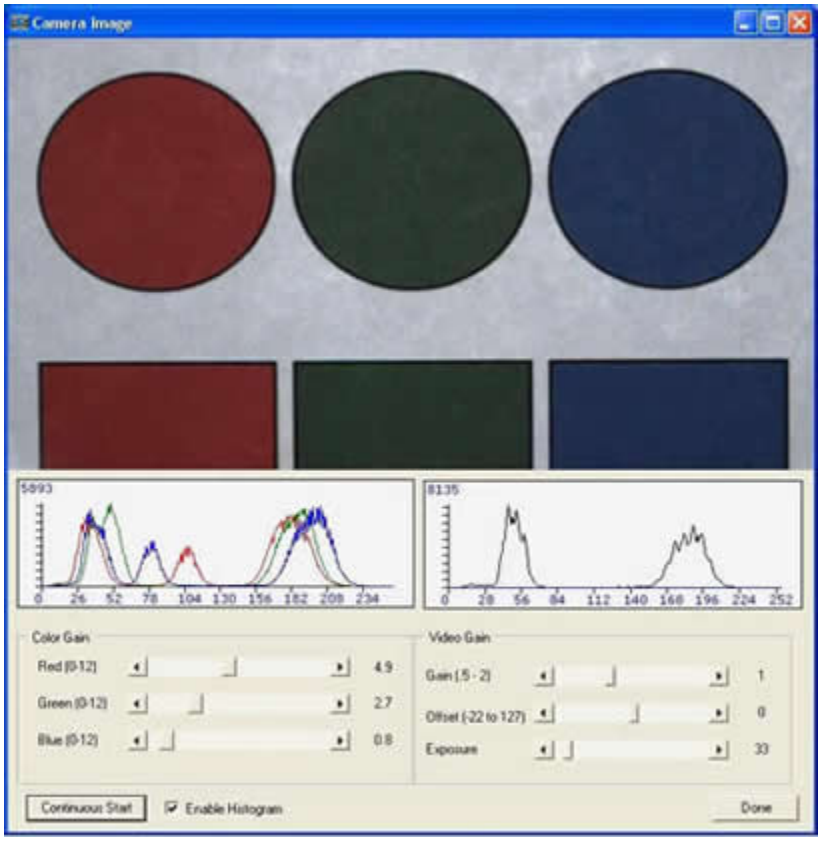
As both a demonstration feature and as an aid in remotely diagnosing problems, this tool can be used to easily store captured images to a disk file and to reload images stored to the disk. To store images, the **AcquireMode** must be set to "ACQUIRE_AND_SAVE" or "SAVE_ONLY". To load files, the mode must be set to "PLAY_FROM_DISK". For any of these modes, the disk file name is constructed by combining the **AcquirePath** with the **AcquirePrefix** and an optional numerical index followed by the .BMP file extension. Each time that an **Acquisition Tool** is executed with "ACQUIRE_AND_SAVE" or "SAVE_ONLY" set, the system automatically increments the numerical index to create a new disk file. Likewise, each time that the **Acquisition** is executed with "PLAY_FROM_DISK", the system automatically searches the file folder for

the file with the next larger numerical index. This automatic indexing allows a series of images to be conveniently generated or replayed.

Special Feature Buttons (located above the property editor)

Adjust Video Properties

Pressing this button displays a pop-up window that permits the property values for the monochrome image (**VideoGain, VideoOffset, ExposureTime**) and the color image (**GainRed, GainGreen, GainBlue**) to be manually adjusted while dynamically viewing the effect of the new settings. In addition, histograms of the various values can be simultaneously displayed.

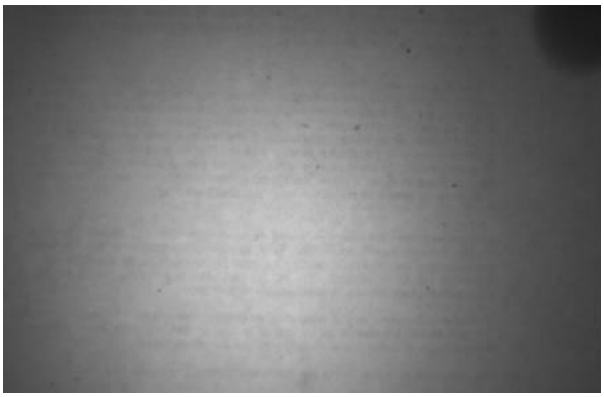


Select Acquire Path

Pressing this button displays a pop-up window that permits you to browse the PC's file structure to select the **AcquirePath** that specifies where image disk files are to be stored and retrieved.

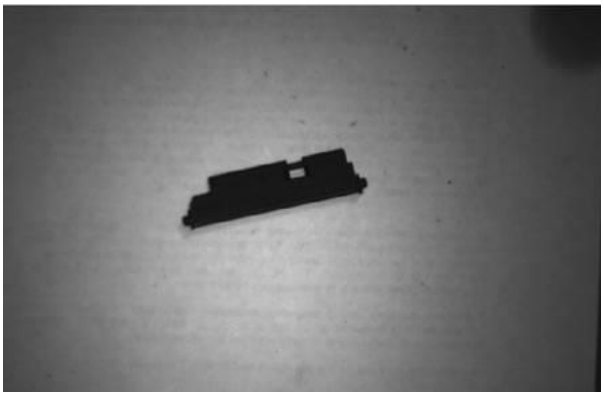
Train Flat Field / Show Flat Field

If the **FlatFieldEnabled** property is True, all captured images are automatically compensated for uneven background lighting by adding a sample inverted image that was previous saved. For example, in the following picture, there is a distinct blooming of the light intensity in the center of the image.

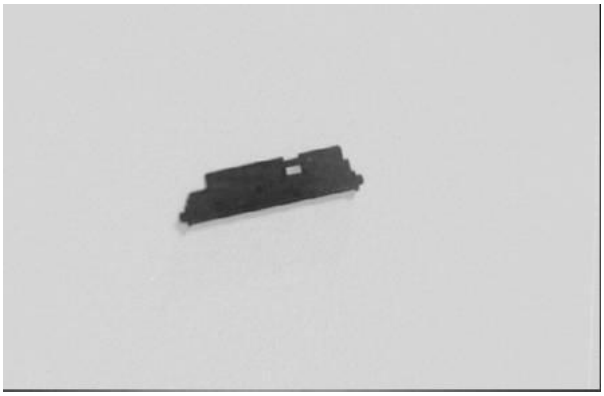


While viewing this background, if you press the **Train Flat Field** button, a copy of the image is inverted, normalized and saved both in memory and in a disk file.

If **FlatFieldEnabled** is False and an object is viewed with this lighting, it will appear as shown below. While the object is relatively distinct, there are variations in the background light intensity that can confuse some vision tools, especially binary tools such as blob finders.



However, if a sample background image is saved and **FlatFieldEnabled** is True, the same captured image will appear as shown below. As can be seen, the object is much more distinctive and the image could even be processed using binary tools without great difficulty.



Examples

If the **AcquireMode** is set to "PLAY_FROM_DISK", and the **AcquirePath** is set to "ExampleProjects", and the **AcquirePrefix** is set to "part2", and the folder that contains the PreciseVision application contains a folder "ExampleProjects" that has the following files:

part2_0001.bmp, part2_0002.bmp, part2_0003.bmp

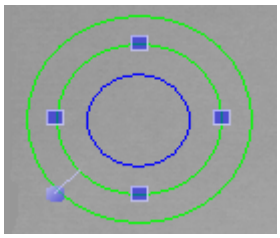
Then, each time that the **Acquisition Tool** is executed, it will load the next successive file in the list above and will restart when it encounters the end of the list.

See Also

Vision Toolkit | **Sensor Window Tool** | **AOI Tool**

Arc Fitter Tool

Vision tool that searches a specified region for arc edge points and returns the circle or arc that best fits the edges.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the tool and the center of the search region.
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
Radius	**Single**	mm	Specifies the nominal radius of the arc or circle to be located. This defines the mid-range value for the radius search.
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
AngularSize	**Integer**	0-360	Specifies the angular size in degrees of the arc to be located. A value of 360 indicates that a complete circle is to be located. Values less than

			360 specify that a partial circle (arc) is to be located.
DarkOutside	**Boolean**	True / False	Defines whether the pixels outside of the arc or circle are darker than the interior pixels. For example, to locate a circle with light interior pixels surrounded by darker pixels, this parameter must be set to True.
EdgeThreshold	**Integer**	0 - 254	Specifies the threshold below which weak (low contrast) gradient edges are ignored. The lower this property is set, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
FindWhat	**List**	FITARC_FIND_CENTER / FITARC_FIND_RADIUS / FITARC_FIND_BOTH	Defines if the radius should be fixed and only the center of the arc varied to find the best match or if the center is to be fixed and the radius varied or if both should be varied.
SearchRadiusRange	**Single**	mm	This property defines the limits of the center point and radius search regions. The search region is centered about the **XPos**, **YPos** and **Radius** values. So, if the **Radius** property is set to a value of N and the **SearchRadiusRange** is set to M, the radius can vary from N-M/2 to N+M/2 and the center point can vary by M in X and Y.
UseWhichEdge	**List**	FITARC_MIDDLE_EDGES / FITARC_INNER_EDGES / FITARC_OUTER_EDGES / FITARC_MAX_GRADIENTS	If multiple arcs exist within the search region, this parameter specifies which arc should be returned. The selected arc can be the one that is closest to the nominal radius, the arc closest to the

			minimum or maximum search radius or the arc that has the greatest contrast (maximum gradients).
3A. Advanced Operation			
DrawEdgePoints	**List**	..._NO_EDGE_POINTS / ..._ALL_EDGE_POINTS / ..._USED_EDGE_POINTS	If set, draws yellow dots to indicate ALL edge points that are selected after the **EdgeMode** is applied or only those edges that are USED to fit arcs after the **EdgeMode** is applied and all outliers are eliminated. This is very helpful for understanding the effect of changing various property settings. This value is NO_EDGE_POINTS by default.
MaxEdgePoints	**Integer**	3 - n	Maximum number of edge point searches to perform. Higher numbers produce greater accuracy at the expense of execution speed. The minimum value of this parameter is 3 and this value is automatically limited to the number of pixels across the width of the tool. The default value is 30.
MaxIterations	**Integer**	0 - n	Maximum number of iterations for filtering. The filtering algorithm repeatedly removes outlier points, refitting the arc each time, until no more points need to be removed or the maximum number of iterations have occurred. A value of zero disables filtering. The default value is 5.
MinFilterDistance	**Single**	0 - n.nn	Absolute minimum filter distance in pixels. No edge points closer to the fitted arc than this are discarded during the iterative process. A minimum distance is needed because

			the standard deviation of distances to the fitted arc can be less than a pixel with a good image of a clean edge. The default value is 1.5.
SigmaFilter	**Single**	0 - n.nn	Filter width in units of standard deviations. This value is multiplied by the standard deviation of the edge points' distances to the fitted arc to compute the distance threshold beyond which edge points are removed during the iterative fitting process. The default value is 2.5.
4. Inspection Settings			
InspectType	**List**	NONE / ..._RMS / ..._RADIUS / ..._NUM_EDGES_FOUND / ...NUM_EDGES_USED	Standard Inspection Settings properties
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

ResultNum-EdgesFound	**Integer**		Number of edge points found. *[GPL: VisResult.Info(1)]*
ResultNum-EdgesUsed	**Integer**		Number of edge points remaining after the filtering process. *[GPL: VisResult.Info(2)]*
ResultRadiusFound	**Single**		Radius of the located arc in mm. *[GPL: VisResult.Info(4)]*
ResultRMS	**Single**		Root mean square of the arc fit. *[GPL: VisResult.Info(0)]*

Remarks

The Arc Fitter searches the region defined by its green and blue outlined arc for edge points and fits an arc to the edge points using a least squares technique. By computing the edge positions to a sub-pixel accuracy and by employing multiple such edges in its computation, this tool is able to generate very accurate sub-pixel results.

This tool is very useful for accurately locating holes or curved edges of objects or fiducials. Once found, these fit arcs can be used to compute center points between important features, for generating reference frames that can accurately locate an object or features in an object or for measuring the diameters of holes.

In order to make this tool more discriminating, the Arc Fitter only utilizes edges whose dark and light sides have roughly the same orientation as the dark (blue) and light (green) sides of the search area. So, when positioning this tool, you should place the center (nominal) arc of the search region at approximately the position and orientation that you expect to find the arc with the blue side on the anticipated dark side of the object. To aid in this operation, there is an 'Auto Size / Position' feature button above the property editor. This button will analyze the coarse position of the tool and snap it to the nearest arc/circle. The radius search range is automatically adjusted as well.

The operation of this tool is basically performed in two steps. First, all of the edges are extracted in the search region. Secondly, an arc is fit to the edges. Of the two operations, the tool spends most of its time extracting the edges. So, the speed and the accuracy of the Arc Fitter can be traded-off by adjusting **MaxEdgePoints**. This property determines how densely the tool scans to detect edges. This parameter must be set to at least 3 in order to detect the minimum number of edges to define an arc. At most, one edge search is performed for each pixel along the width of the Fitter. Execution time increases roughly in proportion to the value of **MaxEdgePoints**. On the other hand, the accuracy of the tools increases approximately as the square root of **MaxEdgePoints**. That is, if you increase **MaxEdgePoints** by a factor of 4, the tool execution time will quadruple but the accuracy will only double.

Once the edges are found, the Arc Fitter will optionally perform an iterative fitting process to increase the robustness of the results. During each pass, edge points that are too distant from the arc (outliers) are discarded. This filters out edges that are not a part of

the arc and whose inclusion would incorrectly offset the result. **MaxIterations** specifies the maximum number of fits to perform. After each arc fit, edges that are beyond **SigmaFilter** standard deviations are rejected, unless the edge is within **MinFilterDistance** pixels of the fit arc. The iterative process stops if (1) **MaxIterations** are performed, (2) no further edges are discarded, or (3) only two edge points remain.

At the conclusion of this tool, the center position and radius of the best fit arc is returned along with statistical data on the number of edges finally used and the root mean square of the arc fit, which indicates the quality of the final fit. This statistical data can be tested via the "Inspect Settings" to yield a pass/fail indication for the operation.

Special Feature Buttons (located above the property editor)

Adjust Edge Threshold

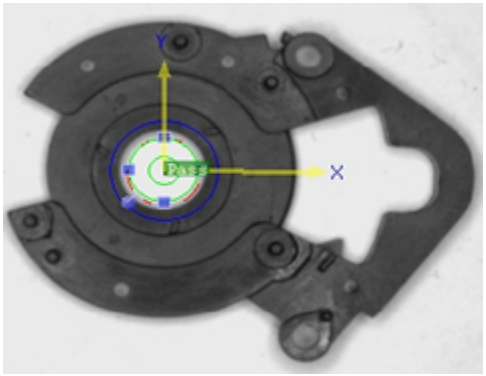
Pressing this button displays a pop-up window that permits the value of the **Edge Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. The display indicates the edges that satisfy the threshold criterion in white.

Auto Size / Position

Provides an easy way to position the tool. This button will compute the currently found arc/circle and automatically snap the tool to the exact coordinates of the found arc/circle. In addition the radius search range is adjusted proportionally to the arc/circle size.

Examples

In the following example, an Arc Fitter is utilized to accurately determine the position of a hole within an object. The best fit radius of the hole is computed and the resultant is compared to the acceptable range of values to generate a pass/fail inspection result.

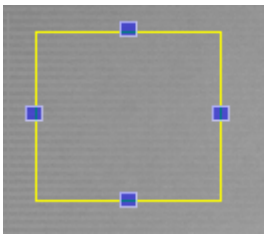


See Also

Vision Toolkit | Edge Locator | Line Fitter

Area-Of-Interest (AOI) Tool

Vision tool that copies a rectangular area of the current image into a file or another region of the image or a different image buffer. When the area-of-interest (AOI) is copied, various processing operations can be optionally performed.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Height	**Single**	0 - AOI (mm)	Standard Placement / Size properties. The X and Y values define the center of the region to be copied. The Height and Width define the dimensions of the region. The AOI is a rectangular region and cannot be rotated.
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
Destination-ToolName	**List**	n/a	If the AOI is not to be copied to a file, another vision tool must be specified to define the destination frame buffer for the copy operation and the center X, Y position of the AOI in the destination. Only the destination tool's XPos, YPos, and Camera are utilized. The destination tool is not executed and the Height, Width and Angle are

			ignored. The destination tool only serves to provide information required for the copy operation.
Extension	**List**	BMP / PNG	If the AOI is to be copied to a file, these properties specify the folder path to the file, the name of the file, and the file extension and type.
FilePath	**String**	n/a	
Prefix	**String**	n/a	
Operation	**LIST**	NONE / SAVE_TO_DISK / COPY_TO_BUFFER / MATH_ADD / MATH_AND / MATH_MAX / MATH_MIN / MATH_XOR / MATH_OR / MATH_SUBTRACT / MATH_SUBTRACT_CLIP	Specifies whether the AOI is to be copied to a disk file or copied to an image buffer. If the AOI is to be copied to an image buffer, various operations can be performed during the copy operation. For example, MATH_ADD copies an AOI to an image buffer and adds the value of each pixel of the AOI to the current contents of the target region in the destination image buffer.
5. Results Settings			
ShowResults	**List**	NONE / FRAME / LINE / POINT	Standard Results Settings properties
6. Results			
ResultErrorCode	**Integer**		Standard Results properties X and Y position for the centroid of the processed area-of-interest.
ResultAngle	**Single**	0 (non-rotating)	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

This tool copies an area-of-interest (AOI) from the current image to a file, another region of the current image, or another image buffer. This is useful for combining several small AOI's into a single image buffer for additional processing or storage.

This tool is similar to the AOI feature of the **Acquisition Tool** in that only a portion of the full image is copied. However, the AOI feature of the **Acquisition Tool** is performed by the camera and reduces the amount of data that the camera sends to PreciseVision (and

reduces the acquisition time as well). The **AOI Tool** operates on an image that has already been acquired.

If you need to only save a portion of an image to a disk file, the **AOI Tool** operates more quickly and generates a smaller file than utilizing the image save feature of the **Acquisition Tool**.

While most vision tools analyze the data in the camera image and leave the data unmodified, this tool can alter the raw data in the image buffer. All subsequent vision tools will operate on the modified image data and not the raw information captured by the camera.

If you wish to revert back to the original raw captured image data, special properties must be set in the **Acquisition Tool** at the time that the picture is first taken.

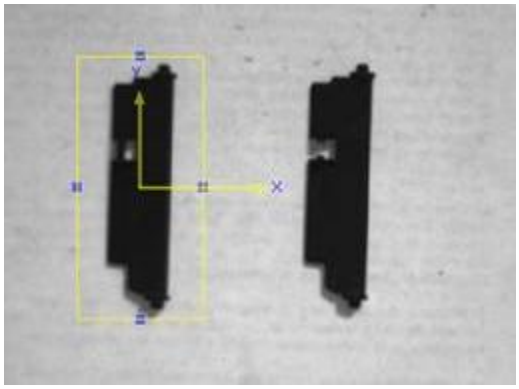
Special Feature Buttons (located above the property editor)

Select Path

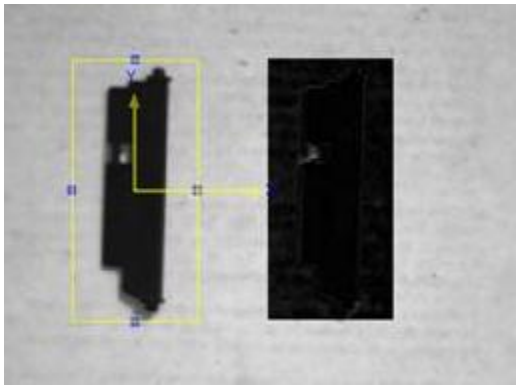
Pressing this button displays a pop-up window that permits you to browse the PC's file structure to select the **FilePath** that specifies where the area-of-interest is to be stored if the **Operation** is set to SAVE_TO_DISK.

Examples

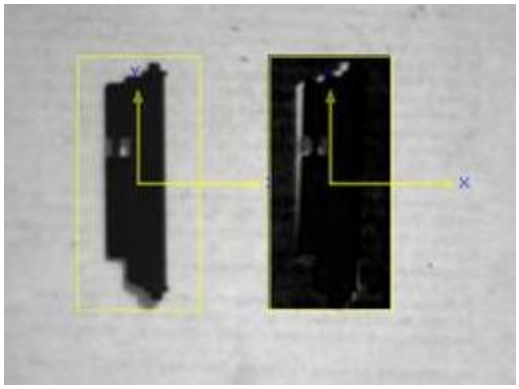
The various mathematical operations permit a number of interesting processes to be performed In the following image, a picture is taken of what should be two identical parts.



If we use the **AOI Tool** to subtract the area on the left from the region on the right, any non-black pixels left in the destination region indicate differences (defects) between the two parts.



Likewise, if the position or orientation of one of the two parts is out of alignment, after subtracting the AOI in the left with the region on the right, any alignment differences will appear as non-black regions.

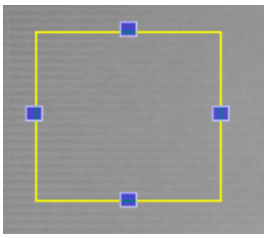


See Also

Vision Toolkit | **Acquisition Tool**

Barcode Reader Tool

Vision tool that reads a variety of standard 1-D and 2-D barcodes and returns the barcode type and the value of the barcode.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	0 (*non-rotating*)	Standard Placement / Size properties. The X and Y values define the center of the region to be analyzed. The Height and Width define the dimensions of the region. The AOI cannot be rotated, but rotated barcodes can be identified (see **SkewTolerance**).
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	0	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
ProcessLevel	**Integer**	0 - 5	Controls the speed with which regions are processed verses accuracy. A lower value will process images more quickly but will result in a lower successful read-rate. The default value is 2.
SkewLineJump	**Integer**	1 - 9	In order to recognize barcodes that are not aligned along the horizontal or vertical axes of the camera, the
SkewTolerance	**Integer**	0 - 5	

			SkewTolerance and **SkewLineJump** must be adjusted. The approximate permitted angular deviations of a barcode from a horizontal or vertical orientation for various settings of the **SkewTolerance** are as follows: 0 - up to 5 degrees 1 - " " 13 " 2 - " " 21 " 3 - " " 29 " 4 - " " 37 " 5 - " " 45 " The **SkewLineJump** determines how many lines are skipped during the portion of the scanning process that is specifically for locating skewed barcodes. 1 means that every line in the region is tested. Lower values for this parameter will increase the processing time but may be useful for poor quality images. If the barcode is aligned along the X or Y axes, the **SkewTolerance** should be set to 0 and the **SkewLineJump** should be set to 9 to minimize the tool's processing time.
3A. Advanced Operation			
AllowDuplicates	**Boolean**	TRUE / FALSE	If a barcode is badly damaged, the same barcode maybe reported twice within the same image. If this parameter is FALSE, duplicate results are only reported a single time. If TRUE, the same value can be returned multiple times in the same image.
Bottom_To_Top	**Boolean**	TRUE / FALSE	These parameters specify the directions in which the software will scan looking for barcodes. If the barcodes always appear in the same orientation, turning off the unneeded scanning directions will reduce execution time. Also, see **SkewTolerance** for the ability to identify barcodes that are not oriented vertically or horizontally.
Left_To_Right	**Boolean**	TRUE / FALSE	
Right_To_Left	**Boolean**	TRUE / FALSE	
Top_To_Bottom	**Boolean**	TRUE / FALSE	
5. Results Settings			

MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
ShowResultType	**Boolean**	TRUE / FALSE	If **ShowResultValue** is TRUE, the identified barcode value will be displayed on top of the barcode in the camera window. In addition, if **ShowResultType** is TRUE, the display will include the type of the barcode that was found.
ShowResultValue	**Boolean**	TRUE / FALSE	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties X and Y position represent the centroid of the located barcode.
ResultAngle	**Single**	0 (non-rotating)	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultCode	**String**		The type of the barcode, e.g. UPCA. *[GPL: VisResult.InfoString]*
ResultCodeDirection	**Integer**		Indicates the scanning direction used to locate the barcode *[GPL: VisResult.Info(1)]* 1 - Left to right 2 - Bottom to top 4 - Right to left 8 - Bottom to top
ResultCodeValue	**String**		The value of the barcode. *[GPL: VisResult.InfoString]*

Remarks

This tool reads a variety of popular types of 1-D and 2-D barcodes. More than one barcode can be located within a single AOI and the barcodes can be of different types (although searching for multiple types of barcodes increases execution time). The value of the barcode as well as its type are returned by this tool and can optionally be displayed in the camera window on top of the identified barcode.

This tool can locate barcodes that are oriented in any direction (although barcodes that are approximately horizontal or vertical are processed most quickly). This tool will operate on grayscale or color images, with grayscale images being processed more rapidly and more reliably due to the single image plane and higher quality edges.

In order to reliably detect barcodes and to determine their correct values, the following should be kept in mind:

- As a general rule-of-thumb, the thinnest bar should be at least 3 pixels wide. Likewise, the smallest gap between bars should be at least 3 pixels wide.
- If a barcode is not horizontal or vertical but is tilted (skewed), the minimum bar and gap width should be increased to at least 5 pixels since diagonal lines look like stair steps and alternately are thick and thin.
- Even if the barcode's position is well known, white space should be included in the AOI around the barcode so that the software can detect the start and end of the bars or squares.

The Barcode Reader Tool can detect a number of different codes and can correctly operate even when the barcode is at any orientation or the image is somewhat degraded or the barcode is slightly defective. However, the execution speed of this tool can be optimized by reducing the generality of the property settings, such as reducing the size of the AOI to be searched, disabling unneeded barcode types, reducing the skew tolerance, etc.

Special Feature Buttons (located above the property editor)

Select Codes

Pressing this button displays a pop-up window that can restrict the types of barcodes that are detected during the scanning process. Disabling unneeded types will reduce the execution time of the tool. Currently, variations of the following types of 1-D and 2-D barcodes can be detected: *Code 39, Code 93, Code 128, Code 25 (interleaved and non-interleaved), Codabar, EAN-8, EAN-13, UPC-A, UPC-E, PDF-417, Data Matrix, Databar, and Patch Codes*.

Advanced Settings

Pressing this button displays a pop-up window that presents more advanced properties of the barcode tool whose default values are normally acceptable. For convenience, this pop-up also contains duplicates for some of the common properties that are contained on the standard Property Display. The following describes the operation of the unique Advanced Settings.

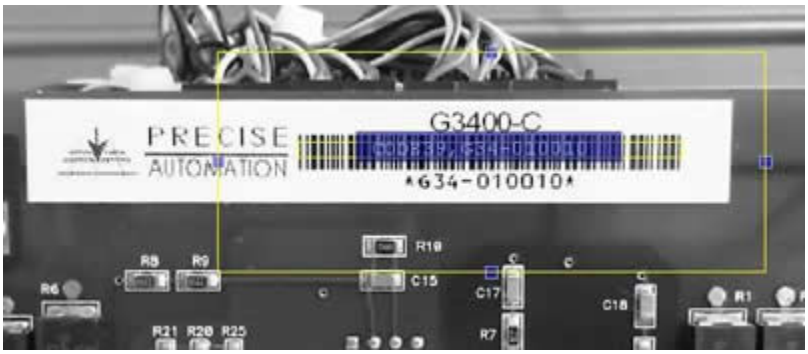
Property Name	Data Type	Range	Description
Convert UPC-E to EAN-13	**Boolean**	TRUE / FALSE	If TRUE, UPC-E barcodes are automatically converted into EAN-13. This is FALSE by default.
GammaCorrection	**Integer**	1 - n	If this value is not 100, a Gamma Correction equal to **GammaCorrection**/100 will be applied to adjust the overall illumination of the AOI. By default,

			this value is 100, which means don't apply a correction.
Line Jump	**Integer**	1 - 9	This property determines how many lines are skipped during the initial scan for a barcode. 1 means that every line in the region is tested. Lower values for this parameter will increase the processing time but may be useful for poor quality images. This value is 1 by default.
Median Filter	**Boolean**	TRUE / FALSE	If TRUE a median filter is applied to the AOI before scanning for the barcode. This can eliminate small flaws in high resolution images. It should not be applied to low resolution images since this will blur the transitions between bars and gaps. This is FALSE by default.
Max Length	**Integer**	1 - 999	Specifies the largest number of characters in the barcode including any checksum characters. Set to 999 by default.
Min Length	**Integer**	1 - 999	Defines the minimum acceptable number of characters in a valid barcode value. Set to 4 by default.
Min Space Width	**Integer**	0 - n	Defines the minimum acceptable width of spaces between bars. Spaces that are smaller than this size are ignored. By default, this value is 0, which implies that the system will automatically determine the minimum acceptable width.
Noise Reduction	**Integer**	0 - n	If greater than 0, the AOI is filtered before being scanned to eliminate marks that are unlikely to be part of the barcode. Larger values will reduce larger marks. This can be helpful in poor quality images. A typical value is 10. By default, this is set to 0.
Numeric Barcode	**Boolean**	TRUE / FALSE	If TRUE, only numeric barcodes are identified. By default, this value is FALSE.
Quite Zone	**Integer**	0 - n	When a line in the image is scanned, regions that are not

			preceded by this number of white pixels are ignored. Set to 0 by default.
Show Check Digits	**Boolean**	TRUE / FALSE	If TRUE, the barcode check digit will be included in the barcode value. This property only applies to barcode types with built-in check digits, such as Code 128. Set FALSE by default.
Use Oversampling	**Boolean**	TRUE / FALSE	If TRUE, 3 sequential lines at a time are scanned and their average pixel value is used to determine the barcode. This is useful for images that contain both white and black speckles. Set FALSE by default.

Examples

The following two examples demonstrate the Barcode Reader Tool identifying the same character pattern presented as both a 1-D Code 39 and a 2-D Data Matrix pattern. In each picture, the darker blue boxes display the barcode type and value as determined by this tool.

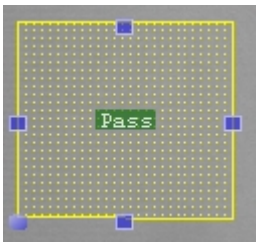


See Also

[Vision Toolkit]

Clear Grip Tool

Vision tool that verifies that there is no obstruction within the bounds of a defined window. This tool is typically used to confirm that gripping a part will not result in a collision.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the verification window. The Height and Width define the dimensions of the window.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
EdgeThreshold	**Integer**	0 - 254	Specifies the threshold below which weak (low contrast) gradients edges are ignored. The smaller the number, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
4. Inspection Settings			

InspectType	**List**	NONE / EDGE_COUNT	Standard Inspection Settings properties
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
ShowResults	**List**	NONE / FRAME / LINE / POINT	Standard Results Settings properties
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultEdgeCount	**Integer**		Number of edge pixels found. *[GPL: VisResult.Info(0)]*
ResultTotalArea	**Integer**		Total number of pixels in the tested rectangular region. *[GPL: VisResult.Info(1)]*

Remarks

This tool specifies a rectangular region of an image and tests it to ensure that it is free of obstacles. This tool is typically used to verify that the fingers of a robot's gripper can reach in and pick up a part without hitting another part. This visual tool verifies that the region is clear by extracting edges within the region and counting the number of edge pixels. If an excessive number of edge pixels exist, this indicates that a part or obstruction is in the region.

The Clear Grip region is normally defined relative to another vision tool, e.g. a Finder, that is used to locate the part. In fact, it is common for two or more clearance regions to be defined for a part, one for each of the gripper's fingers. When the Clear Grip tool is processed, all of the pixels within the region whose gray-scale gradient value exceeds the **Edge Threshold** are counted as edge pixels. By setting the **InspectType** to EdgeCount, if the number of edges falls outside of the **InspectMax** and **InspectMin** limits, this tool will signal a failure.

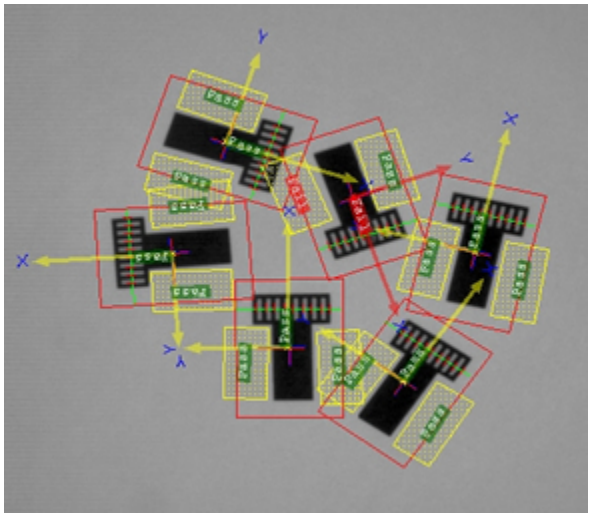
Special Feature Buttons (located above the property editor)

Adjust Edge Threshold

Pressing this button displays a pop-up window that permits the value of the **Edge Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. The display indicates the edges that satisfy the threshold criterion in white.

Examples

In the following example, the Clear Grip tool is used to test if randomly oriented and positioned parts can be safely grasped. As each part is located by a Finder, two Clear Grips tools are tested, one on each side of the part. If the grip zone does not contain a portion of another part, the zone is marked as passed. However, in the sample picture, one zone in the center of the image is partially occluded by another part and that zone is marked as failed. This would prevent the robot from attempting to pick up the part in the center and potentially damaging the adjacent part or the gripper.

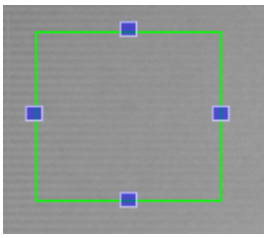


See Also

Vision Toolkit | **Finder Tool**

Connectivity (Blob) Tool

Vision tool that searches a specified region for binary blobs (and/or holes) and returns the centroid and other features of each blob (and/or hole). In addition, this tool can return data that defines the perimeter of a located blob.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	0 (*non-rotating*)	Standard Placement / Size properties. The X and Y values define the center of the tool and the center of the search region. The Height and Width specify the size of the search region.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	0	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
BlobFindMode	**List**	BLOBS_ONLY / BLOBS_AND_HOLES / HOLES_ONLY	Specifies if only blobs that have the correct **ObjectColor** are to located or if blobs and/or holes are to be located. A hole is a region of the opposite **ObjectColor** that is contained within a blob. By

			default, only blobs are located.
MaxArea	**Integer**	0 - n	Specifies the maximum and minimum number of pixels that a valid blob must contain. Any blob that consists of a greater or lesser number of pixels will be ignored.
MinArea	**Integer**	0 - n	
ObjectColor	**List**	DARK_OBJECT / LIGHT_OBJECT	After the image area is converted to binary, defines whether connected groups of black or white pixels should be considered as blobs.
Threshold	**Integer**	0 - 254	Specifies the threshold below which gray-scale pixels will be set to black and above which pixels will be set to white. When this tool is created, the system will automatically set this property to a reasonable value for the full vision window. This value can also be updated using the special feature button described below or the **AutoThresh** property.
3A. Advanced Operation			
AutoThresh	**Integer**	0 or <>0	If set non-zero, the **Threshold** property is automatically computed. This is equivalent to pressing the special feature button that is described below.
IgnoreClipped	**Boolean**	TRUE / FALSE	If TRUE, any blobs that are not completely inside of the **Width** and **Height** search area are ignored. If FALSE, even if a portion of a blob is clipped by the search area boundaries, it is still returned as a result.
MaxHoleArea	**Integer**	0 - n	

MinHoleArea	**Integer**	0 - n	Maximum and minimum number of pixels required before a hole in a blob is retained and not filled in. If the minimum value is set to the total area of the image in pixels, holes will always be deleted.
MaxRadiusLength	**Single**	mm	If either of these parameters is non-zero, the **ResultMaxRadiusLength** and the **ResultMinRadiusLength** of a blob must fall within these limits. Any blobs whose radiuses fall outside of this range will be ignored.
MinRadiusLength	**Single**	mm	
SampleRate	**Integer**	0 - n	When the tool is set to return PERIMETER information, this properly specifies how many pixels to skip along the perimeter between each returned "result" position. A value of '0' will not skip any pixels and every perimeter point is returned.
4. Inspection Settings			
InspectType	**List**	NONE / NUM_BLOB_FOUND / MAX_RADIUS / MIN_RADIUS / MAX_RADIUS_ANGLE / MIN_RADIUS_ANGLE / MAJOR_AXIS_ANGLE / MINOR_AXIS_ANGLE / BLOB_AREA / MAJOR_LENGTH / MINOR_LENGTH / PERIM_LENGTH	Standard Inspection Settings properties
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		

InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties. **MaxResults** defines the number of blobs or perimeter points to find each time this tool is executed. By default, this property is set to -1, find all possible blobs or perimeter points. See below for a description of **ResultMode**.
OffsetAngle	**Single**	degrees	
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultMode	**List**	BLOB_CENTER / MIN_RADIUS / MAX_RADIUS / MAJOR_AXIS / MINOR_AXIS / PERIMETER	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
ResultSort	**List**	SORT_NONE / SORT_MAX_AREA / SORT_MIN_AREA / SORT_CENTER_MOST / SORT_LEFT_MOST / SORT_RIGHT_MOST / SORT_TOP_MOST / SORT_BOTTOM_MOST	Specifies the order in which located blobs and holes are returned in the results.
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties **ResultCount** indicates the number of blobs located in non-PERIMETER mode. In PERIMETER mode, this is sum of all perimeter points found for all blobs.
ResultErrorCode	**Integer**		
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultBlobArea	**Integer**	pixels	Total area of the blob in pixels excluding any holes within the blob. *[GPL: VisResult.Info(3)]*
ResultNumPerBlobs	**Integer**	0 - n	For PERIMETER mode, this is the total number of blobs located and **ResultCount** indicates the sum of all perimeter points available. For non-PERIMETER mode, this

			value is undefined and **ResultCount** indicates the total number of blobs located. *[GPL: VisResult.Info(2)]*
ResultObjectIdx	**Integer**		Each blob in a region is assigned an index. In PERIMETER mode, each returned *VisResult* represents a single perimeter point. If multiple blobs are located, this property can be monitored to determine when the perimeter points pertain to the next blob. *[GPL: VisResult.Info(0)]*
ResultMajorAxisAngle	**Single**	-360 to 360	Angle *[GPL: VisResult.Info(8)]* and Length *[GPL: VisResult.Info(9)]* of the major axis of the best fit ellipse for the blob. This axis provides a means for characterizing the orientation of an arbitrary shape unambiguously within +-90 degrees. By definition, this ellipse has the same 2nd moments of inertia as the blob.
ResultMajorAxisLength	**Single**	mm	
ResultMinorAxisAngle	**Single**	-360 to 360	Angle *[GPL: VisResult.Info(10)]* and Length *[GPL: VisResult.Info(11)]* of the minor axis of the best fit ellipse for the blob. This axis provides a means for characterizing the orientation of an arbitrary shape unambiguously within +-90 degrees. By definition, this ellipse has the same 2nd moments of inertia as the blob.
ResultMinorAxisLength	**Single**	mm	
ResultMaxRadiusAngle	**Single**	-360 to 360	Angle *[GPL: VisResult.Info(6)]* and Length *[GPL: VisResult.Info(7)]* of the
ResultMaxRadiusLength	**Single**	mm	

			maximum radius for the blob. This radius is measured from the blob centroid to the point on the perimeter that is furthest from the centroid. This provides a means for characterizing the orientation of an arbitrary shape. However, the uniqueness of this radial line is a function of the shape.
ResultMinRadiusAngle	**Single**	-360 to 360	Angle *[GPL: VisResult.Info(4)]* and Length *[GPL: VisResult.Info(5)]* of the minimum radius for the blob. This radius is measured from the blob centroid to the point on the perimeter that is closest to the centroid. This provides a means for characterizing the orientation of an arbitrary shape. However, the uniqueness of this radial line is a function of the shape.
ResultMinRadiusLength	**Single**	mm	
ResultPerimeterLength	**Single**	mm	The length of the perimeter of the blob. *[GPL: VisResult.Info(12)]*

Remarks

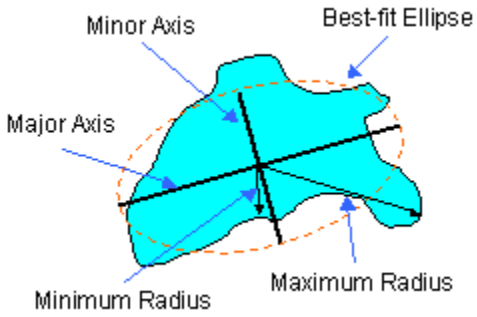
The Connectivity tool converts the region defined by its green outlined rectangle to a binary image and groups adjacent ("connected") pixels with the correct **ObjectColor** into blobs. Any blobs that do not satisfy the specified size limits as well as any interior blobs (i.e. blobs within blobs) are not returned. For all valid blobs, the position of their centroid and other key features are computed and returned as the results of this tool.

This tool is similar to the Finder Tool in that multiple objects can be identified and located. However, this is a less discriminating and less powerful tool than the Finder, but it is nonetheless very useful. In particular, Connectivity should be used for locating objects that do not have a fixed boundary and in applications that have high contrast with visually simple images. Parts that vary in size and shape are easily located using this tool. In addition, this tool has the ability to return the perimeter of any blob as a series of individual results points. This perimeter retrieval feature allows an application to easily trace an object, which is useful in the textile or automated cutting industries.

If the vision system must differentiate between objects that differ by subtle features, locate objects that touch other objects, locate objects that are not clearly differentiated from the background, or operate when thresholding may not be appropriate or robust, the Finder Tool should be utilized in place of Connectivity.

With regard to the performance of the Connectivity tool, even though this tool operates on binary images, the X & Y position of the centroid of blobs is normally accurate to within a fraction of a pixel. This is due to the fact that the centroid is computed by taking into consideration the positions of all of the perimeter pixels. In general, the larger the blob, the more accurate the centroid position. Consequently, the centroid position is a very good measure of where the blob is located.

In addition to the blob centroid, each blob has several other features computed as indicated in the following drawing.



These additional features can be used to determine the orientation of the blob. Depending upon the exact shape of the object, some of these features will be more useful than others. For example, if the blob is a rectangle, all four corners will be at the same maximum radius from the centroid, so the **ResultMaxRadiusAngle** can return any one of four values, all of which are correct. In many cases, the orientation of the major axis of the best-fit ellipse (**ResultMajorAxisAngle**) can provide a very good orientation estimation. But, it should be kept in mind that this parameter always has a 180 degree ambiguity due to the symmetry of an ellipse.

The **ResultMode** property selects which of the properties is utilized to compute the orientation of each blob or if the orientation is to be fixed (BLOB_CENTER).

If the **ResultMode** property is set to PERIMETER, this tool returns a series of results, one for each desired point on the perimeter of each located blob. Each *VisResult* contains the X & Y position of a point on the perimeter with the orientation angle set to be perpendicular to the perimeter of the blob. In this mode, the **SampleRate** specifies if some pixels on the perimeter are to be skipped. For example, if the **SampleRate** is set to 3 and two blobs are located, the coordinates of every 3rd pixel on the perimeter of the first blob will be returned in the *VisResult* and **ResultObjectIdx** will be set to 1. After the last perimeter position of the first blob is transmitted, every 3rd pixel of the perimeter of the second blob will be returned with the **ResultObjectIdx** set to a value of 2.

Special Feature Buttons (located above the property editor)

Automatic Threshold

The Connectivity tool operates on a thresholded binary image. This requires a specific gray level to be entered in the **Threshold** property to define which pixels are to be converted to black (below the **Threshold**) and white (above the **Threshold**). To help set this value, there is an "Automatic Threshold" button available above the property editor. When this button is pressed, the **Threshold** property is automatically set to the system's best estimate for the tool's search region.

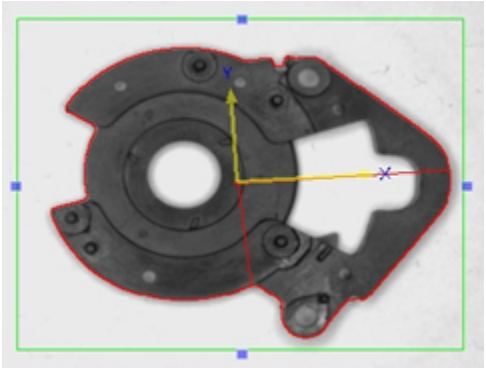
If the **Threshold** is properly set, a thin red boundary should will be drawn along the perimeter of each located object. If required, manually adjust the **Threshold** value to ensure the displayed red boundary is properly aligned with the object boundary.

Adjust Threshold

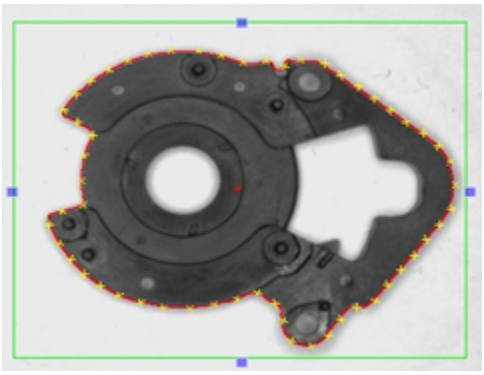
Pressing this button displays a pop-up window that permits the value of the **Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. The displayed image shows all pixels converted to black or white, where the white pixels are the ones that are above the threshold.

Examples

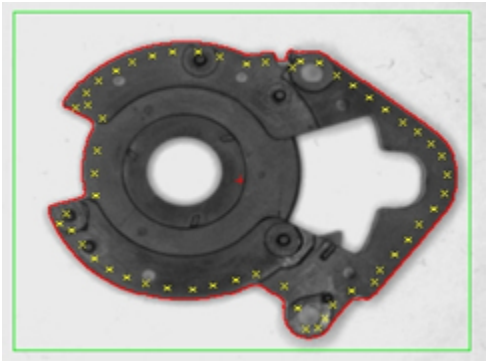
In the following example, a single part is located using a Connectivity (blob) tool. The maximum radius is used to determine the orientation of the part (**ResultMode** set to MAX_RADIUS). The red outline illustrates the perimeter of the object as computed by the tool. The intersection of the X and Y axes denotes the computed centroid of the blob. The X-axis is aligned with the radial line to the perimeter point that is furthest away from the centroid. For this part, the maximum radius point yields a unique orientation angle for the object.



In the following example, the **ResultMode** is set to PERIMETER and the Connectivity tool returns the locations of the points along the perimeter of the object. In this example, the **SampleRate** has been set to 20, so a yellow "X" is drawn every 20 pixels along the perimeter representing each of the returned values.



In this following example, a Fixed Frame tool has been added to the vision process. The Fixed Frame has its **RelativeToolName** set to the name of the Connectivity tool. The Connectivity tool is once again set into PERIMETER mode. Since the orientation of each of the perimeter results is perpendicular to the perimeter of the blob, the Fixed Frame can be used to create a series of points that are a fixed distance inside of the boundary of the blob. In the picture below, the display of the blob perimeter points has been turned off and the position of each Fixed Frame is being displayed as a yellow "X".

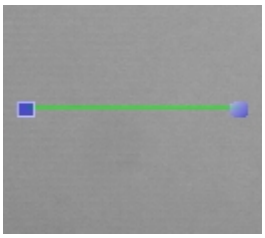


See Also

Vision Toolkit | **Finder**

Edge Locator Tool

Vision tool that detects edge points with sub-pixel accuracy along a linear path in a vision image and returns their positions. Alternately, this tool can return an array of the intensities along the linear path or a histogram of their values.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the tool.
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
Length	**Single**	0 - AOI (mm)	Defines the length of the linear path that is tested for edges.
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
EdgeThreshold	**Integer**	0 - 254	Specifies the gray scale threshold value used to detect edges or (in the case of gradient values) the threshold for the rate of change of the gray scale values. The smaller the number, the more sensitive the system is in locating edges. The default value is 30.
3A. Advanced Operation			

DesiredAlg	**List**	EDGE_BINARY / EDGE_GRAYTHRES / EDGE_SUBPIX / EDGE_HIGH_-PRECISION_SUBPIX	EDGE_BINARY: identifies edge transitions in a binary image. EDGE_GRAYTHRESH: identifies an edge position whenever the gray scale value crosses a specified threshold value. EDGE_SUBPIX: computes the positions of edges to a sub-pixel accuracy based upon the rate of change (gradient) of the gray scale values. This is the default setting. EDGE_HIGH_PRECISION_SUBPIX: similar to the sub-pixel mode except that a more precise interpolation technique is employed that provides more accurate sub-pixel edge locations at the cost of a small increase in execution time. In rare cases where very fine features (e.g. 2 or 3 pixel bumps) are contained in the edge and need to accurately accounted for, EDGE_SUBPIX may produce more accurate results since that method uses a smaller pixel region to compute its results.
DesiredGrads	**List**	EDGE_ALLGRADS / EDGE_POSGRADS / EDGE_NEGGRADS	This parameter specifies whether all edges are to be detected (EDGE_ALLGRADS) or only edges that occur when the gray scale value goes from dark-to-light (EDGE_POSGRADS) or only edges when the gray scale value goes from light-to-dark (EDGE_NEGGRADS). Detecting only negative or positive edges helps disambiguate the edge being detected or can eliminate double edges that occur when thin dark or light features are encountered. Processing starts at the square resizing handle and ends at the round rotation handle.
MaxEdgePoints	**Integer**	1 - n	Maximum number of edge points to return.
4. Inspection Settings			

InspectType	**List**	NONE / NUMBER_OF_POINTS / DISTANCE_TO_-FIRST_POINT / DISTANCE_TO_-LAST_POINT / DISTANCE_TO_-MID_POINT / DISTANCE_FROM_-FIRST_TO_LAST / GRAY_AVERAGE / GRAY_MIN / GRAY_MAX	Standard Inspection Settings properties. All of the inspected results should be self explanatory except perhaps the "GRAY_...", which test the average or min/max grayscale values along the length of the entire tool.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties
OffsetAngle	**Single**	degrees	
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ResultSelect	**Integer**	1 - n	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
ResultMode	**List**	RESULT_EDGE_POINTS / RESULT_FIRST_POINT / RESULT_LAST_POINT / RESULT_MID_POINT / RESULT_PASS_-THRU_SOURCE / RESULT_HISTOGRAM / RESULT_GRAYLEVEL	Defines whether all edge points or selected edge points are to be returned in the results. The "RESULT_PASS_THRU_SOURCE" is a special mode where this tool returns the "Results" of its parent tool pointed to by **RelativeToolName**. This permits a process to add one or more Edge Locators after it has computed the position of a part and use the Locators to perform additional visual verification tests, but still have the process return the "Results" location

			of the parent tool. Please see the "PassFailEdgeCount" demonstration program that was shipped with PreciseVision for an example of the use of this feature. The "RESULT_GRAYLEVEL" and "RESULT_HISTOGRAM" are special modes that return an array containing either the intensity (grayscale) value for each pixel along the length of the tool or the computed histogram of the distribution of intensity values. If either of these modes are selected, edge points are not detected.
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties
ResultErrorCode	**Integer**		
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

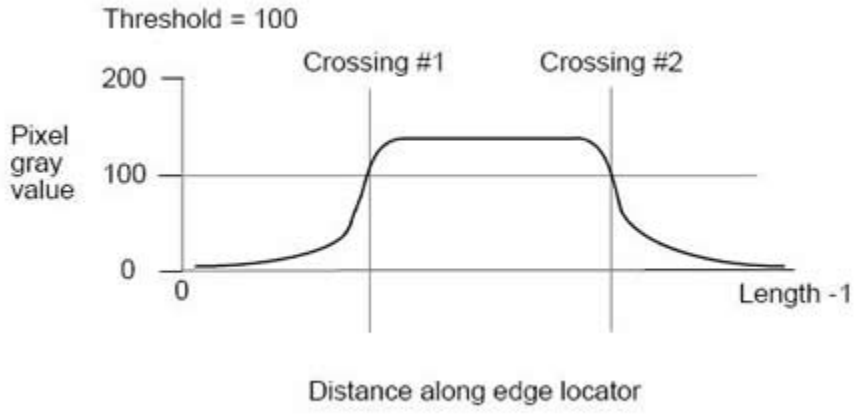
Remarks

This tool finds object boundaries, i.e. edges, along a linear path in an image. The edge positions can be used to measure the dimensions of objects or detect if key features are present or absent. The edges can be detected in a number of different modes (e.g. threshold, sub-pixel, etc). Alternately, the raw pixel values or a histogram of the grayscale values of the pixels along the path can be returned. In other vision systems, this type of tool is sometimes called a "caliper" or "ruler".

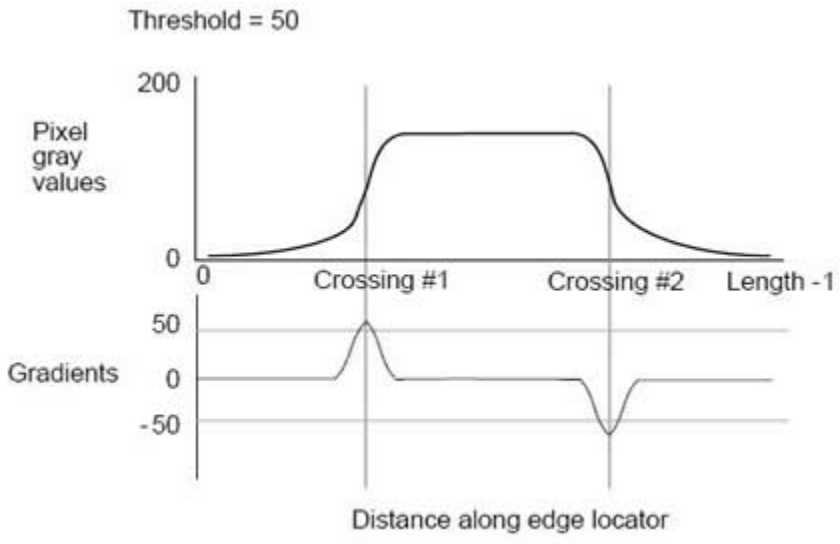
Locating Edge Points

When locating edge points, this tool operates by analyzing pixels along the specified line. Processing starts at the square resizing handle and ends at the round rotation handle. In cases where you wish to compute the position of a single straight or smoothly curved edge, the Line Fitter and Arc Fitter tools will yield more accurate results because they compute an edge position utilizing many more pixel values.

The primary mode of operation for this tool is specified by the **Desired Alg** property. If this parameter is set to "EDGE_GRAYTHRESH", an edge is detected whenever the gray scale value crosses the threshold specified by **EdgeThreshold**. The graph below shows a typical example of how the gray scale values might appear along the length of an Edge Locator. In this case, if the threshold is set to 100, two edges will be detected (Crossing #1 and #2) where the gray scale values cross the specified threshold value.



If the **Desired Alg** property is set to "EDGE_SUBPIX", edge positions are computed to a sub-pixel accuracy by first computing the rate of change ("gradients") of the gray scale values along the length of the Edge Locator. For each region where the gradients exceed the **Edge Threshold** value, the peak of the gradient is computed with sub-pixel accuracy and returned as the location of the edge.



If **Desired Alg** is set to "EDGE_HIGH_PRECISION_SUBPIX", a higher-resolution sub-pixel position is computed than the ""EDGE_SUBPIX" method by employing an even more accurate (and costly) sub-pixel determination method. This method analyzes a longer gradient neighborhood along the path of the Edge Locator.

Finally, if **Desired Alg** is set to "EDGE_BINARY", 0-to-1 or 1-to-0 transitions are detected in a binary image.

After all of the edges have been determined, this tool can return all of the located edges or a selected edge based upon the setting of the **ResultMode** property. Also, the standard "Inspection Settings" can be used to yield Pass / Fail results based upon the number of edges detected, various distance relationships or the average grayscale value along the entire length of the tool.

Computing Intensity Histogram

If the **ResultMode** is set to "RESULT_HISTOGRAM", rather than computing edge points, this tool counts the number of pixels along the specified line that has each of the possible 256 grayscale values and returns the pixel counts. This count of the grayscale distribution of pixels is called an intensity "Histogram". It is useful for determining the prevalent grayscale values and the intensity differences between the peak values.

If you are executing GPL version 2.0 or later, an array of 256 counts (0[black] - 255[white]) can be fetched using the **VisResults Info** property. This array of results is also displayed in the Vision Results panel of the PV user interface in the "Special Results" section and can be recorded if "logging" is enabled.

Whenever this mode is selected, the **ResultXPos** and **ResultYPos** properties represent the center of the specified line and can be used to further propagate the vision tree.

Returning Grayscale Values

If the **ResultMode** is set to "RESULT_GRAYLEVEL", rather than computing edge points, this tool returns the grayscale (intensity) value for each pixel along the length of the specified line. Each grayscale value can range from 0 (black) to 255 (white).

If you are executing GPL version 2.0 or later, an array of up to 1500 pixel values can be fetched using the **VisResults Info** property. This array of results is also displayed in the Vision Results panel of the PV user interface in the "Special Results" section and can be recorded if "logging" is enabled.

Whenever this mode is selected, the **ResultXPos** and **ResultYPos** properties represent the center of the specified line and can be used to further propagate the vision tree.

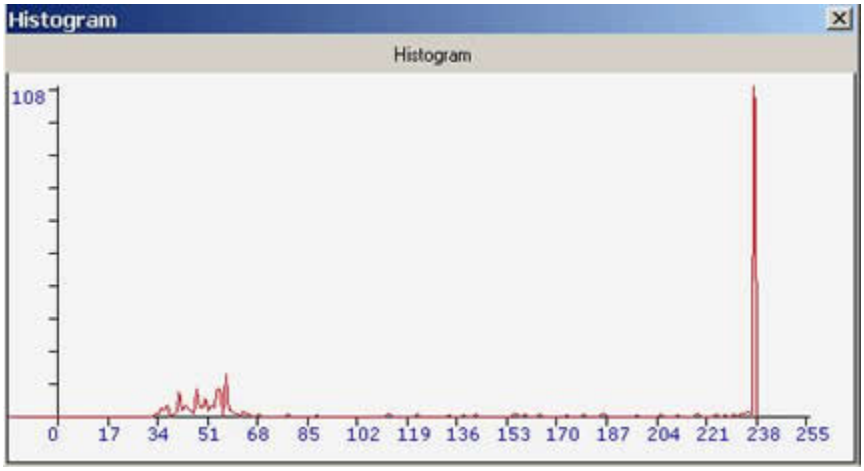
Special Feature Buttons (located above the property editor)

Adjust Threshold

Pressing this button displays a pop-up window that permits the value of the **Edge Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. If the **DesiredAlg** mode is "EDGE_BINARY", the display illustrates the pixels above the threshold as white. Otherwise, for the grayscale modes, the display indicates the edges that satisfy the threshold criterion in white.

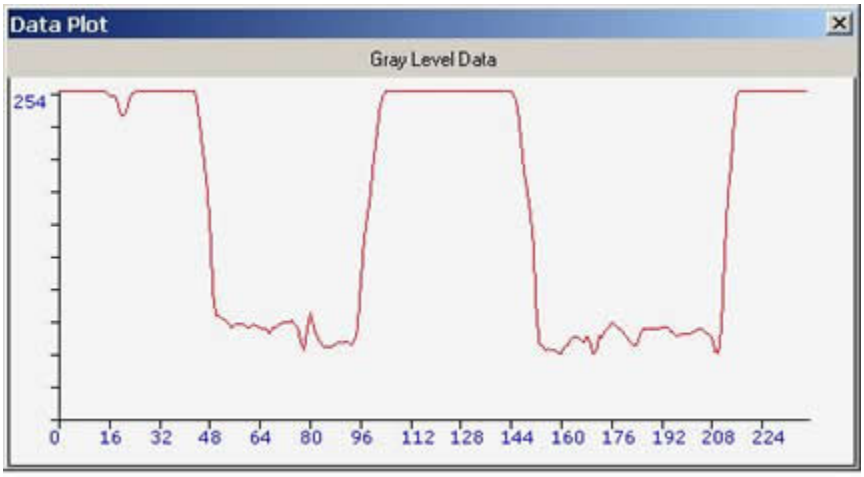
Show Histogram

If "RESULT_HISTOGRAM" mode is selected, this button will pop-up a window that displays the histogram for the specified line in the current image. A sample graph is shown below. This graph corresponds to the tool placement described in the Example below. You will note the large number of white pixels with grayscale values around 238 and a distribution of dark pixels with grayscale values between 34 and 68.



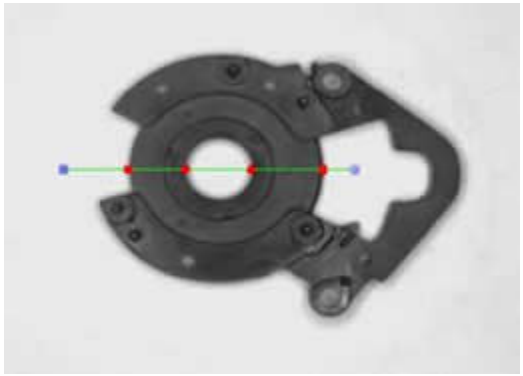
Show Gray Level

If "RESULT_GRAYLEVEL" mode is selected, this button will pop-up a window that displays the grayscale levels for each pixel along the specified line in the current image. A sample graph is shown below. This graph corresponds to the tool placement described in the Example below. Note the low intensity values for the pixels that correspond to the black sections of the object.



Examples

In the following example, an edge locator starts on the left of the image (at the square resizing handle) and processes moving to the right (towards the round rotation handle). In this case, four edges are detected and can be utilized to measure the diameter of the center hole and the thickness of the two sides. (Note, an Arc Fitter could have been used to determine the diameter of the center hole and would have resulted in a more accurate measurement.)

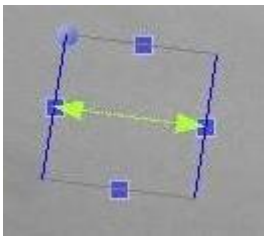


See Also

Vision Toolkit | **Arc Fitter** | **Find Point** | **Line Fitter**

Find Centerline Tool

Vision tool that searches a specified region for edge points and returns the centerline between two located bounding lines.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the tool and the center of the search region. The Height and Width specify the size of the search region.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
EdgeMode	**List**	..._NEAREST_DARK / ..._NEAREST_NOMINAL / ..._NEAREST_LIGHT / ..._MAX_GRADIENTS / ..._MAX_CONCENTRATION	If several possible lines exist within the search region, this parameter helps to distinguish which two are selected as the bounding lines. In addition to selecting the lines closest to the dark sides, light sides, nominal (center line), or greatest concentration of

			edges, you can also select the lines with the highest contrast. The concentration mode is particularly useful if there are two or more distinct non-colinear groups of edges per possible boundary line and a boundary line is attempting to span multiple groups. If concentration mode is specified, for each bounding line, this tool will select the largest cohesive group of edges as the basis for fitting the line.
EdgeThreshold	**Integer**	0 - 254	Specifies the threshold below which weak (low contrast) gradients edges are ignored. The smaller the number, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
3A. Advanced Operation			
BiasFilter	**List**	..._BIAS_NONE / ..._BIAS_TO_LIGHT / ..._BIAS_TO_DARK	For boundary lines with an irregular or jagged edge, this parameter can be set to provide extra weight (importance) to edges that are closest to the light or dark side of the search region. For example, for a saw toothed pattern, this property could be set to fit each boundary line to the tips of the high or low points of the pattern.
DrawEdgePoints	**List**	..._NO_EDGE_POINTS / ..._ALL_EDGE_POINTS / ..._USED_EDGE_POINTS	If set, draws yellow dots to indicate ALL edge points that are selected after the **EdgeMode** is applied or only those edges that are USED to fit boundary lines after the **EdgeMode** is applied and all outliers are eliminated. This is very helpful for

			understanding the effect of changing various property settings. This value is NO_EDGE_POINTS by default.
MaxEdgePoints	**Integer**	2 - n	Maximum number of edge point searches to perform across the width of this tool. Higher numbers produce greater accuracy at the expense of execution speed. The minimum value of this parameter is 2. This value is automatically limited to the number of pixels along the height of the tool. The default value is 30.
MaxIterations	**Integer**	0 - n	Maximum number of iterations for filtering per boundary line. The filtering algorithm repeatedly removes outlier points, refitting a boundary line each time, until no more points need to be removed or the maximum number of iterations have occurred. A value of zero disables filtering. The default value is 5.
MinFilterDistance	**Single**	0 - n.nn	Absolute minimum filter distance in pixels. No edge points closer to a boundary line than this are discarded during the iterative process. A minimum distance is needed because the standard deviation of distances to a fit line can be less than a pixel with a good image of a clean edge. The default value is 1.
SigmaFilter	**Single**	0 - n.nn	Filter width in units of standard deviations. This value is multiplied by the standard deviation of the edge points' distances to a boundary line to compute the distance threshold beyond which edge points are

			removed during the iterative boundary line fitting process. The default value is 1.5.
4. Inspection Settings			
InspectType	**List**	NONE / ..._RMS / ..._ANGLE / ..._NUM_EDGES_FOUND / ..._NUM_EDGES_USED / ..._LINE_THICKNESS	Standard Inspection Settings properties The LINE_THICKNESS is the average distance between the two bounding lines measured at the center of the tool.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultLine-Thickness	**Single**		Average distance between the two bounding lines measured at the center of the tool. *[GPL: VisResult.Info(3)]*
ResultNum-EdgesFound	**Integer**		Number of edge points found during the boundary line fitting. *[GPL: VisResult.Info(1)]*
ResultNum-EdgesUsed	**Integer**		Number of edge points remaining after the filtering process for the boundary lines. *[GPL: VisResult.Info(2)]*

ResultRMS	**Single**		Root mean square of the boundary line fits. *[GPL: VisResult.Info(0)]*

Remarks

This tool searches the region defined by its gray and blue outlined rectangle for edge points, computes the best fit for two bounding lines, and returns the centerline between the bounding lines. This is a convenience routine that internally executes two **Line Fitter** tools to accurately locate the bounding lines using least squares fitting and sub-pixel edge extraction techniques. The search area for each **Line Fitter** is identical to the region specified for the **Find Centerline** tool. The orientation of the two **Line Fitters** differs by 180 degrees. After the **Line Fitters** have completed execution, the center line between the fit boundary lines is returned by this tool.

This tool is very useful for accurately locating a center line between two outer edges of an object. Once found, the center line can be used to compute distances between important features or for generating reference frames that can accurately locate an object or features in an object. This tool also returns the average thickness between the two boundary lines measured at the center of the tool.

When positioning this tool, you should place the center (nominal) line of the search region (as indicated by the green double headed arrow) at approximately the position and orientation that you expect to find the center line.

For a detailed explanation of the operation of the boundary line fitting operation and the effect of this tool's various properties on the fitting process, please refer to the Remarks section of the **Line Fitter** tool.

At the conclusion of this tool, the center position and orientation of the computed center line are returned. In addition, statistical data on the number of edges used in the line fitting operations and the root mean square of the line fits are returned. This statistical data indicates the quality of the final fits and can be tested via the "Inspect Settings" to yield a pass/fail indication for the tool's operation.

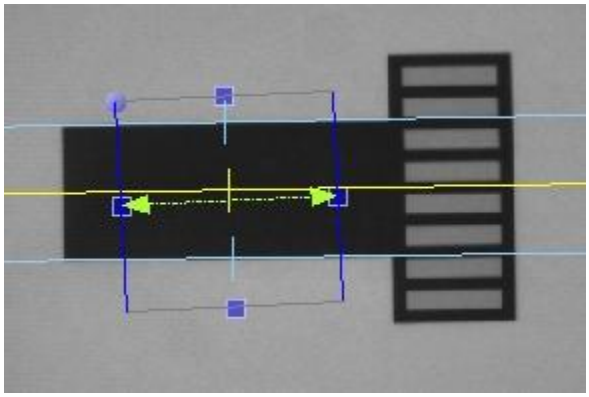
Special Feature Buttons (located above the property editor)

Adjust Threshold

Pressing this button displays a pop-up window that permits the value of the **Edge Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. The edges that satisfy the threshold criterion are displayed in white.

Examples

In the following example, a single **Find Centerline** tool is utilized to accurately determine the orientation of the primary axis of an object.

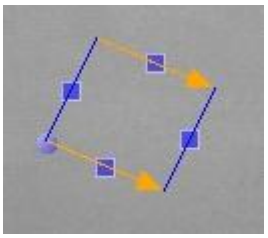


See Also

[Vision Toolkit](#) | [Line Fitter](#) | [Edge Locator](#)

Find Point Tool

Vision tool that locates the edge point that is closest to the line that defines the start of the tool's search region. The XY position of the point is returned with sub-pixel accuracy along (and pixel accuracy perpendicular to) the search direction.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the tool and the center of the search region. The Width defines the distance that is searched looking for the first edge point. The Height defines the length of the reference search line.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
Threshold	**Integer**	0 - 254	Specifies the threshold below which weak (low contrast) gradients edges are ignored. The smaller the number, the more sensitive the system is in locating edge points. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
3A. Advanced Operation			

ToolDisplayMode	**List**	NoDisplay / DisplaySearch	DisplaySearch illustrates all of the edges that were located within the search region.
4. Inspection Settings			
InspectType	**List**	NONE (reserved for future use)	Standard Inspection Settings properties.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

This tool searches a region of interest for the edge point that is closest to the line that defines the start of the rectangular search region. The distance between the edge point and the line is measured along the perpendicular to the starting line that passes through the point. The direction of search is indicated by two arrows that mark the boundary of the search region.

This tool is useful for locating the closest position between a part's boundary and a reference line. This tool is very convenient if the boundary is not a smooth curve or has points of interest such as corners or cusps.

Candidate edge points are located using the same method as the sub-pixel mode of the Edge Locator. So, the XY position of the located edge point is computed with sub-pixel accuracy along the direction of search and to within a pixel perpendicular to the direction

of the search. The position of the returned point can be used in combination with other tools to establish the position or thickness of an object.

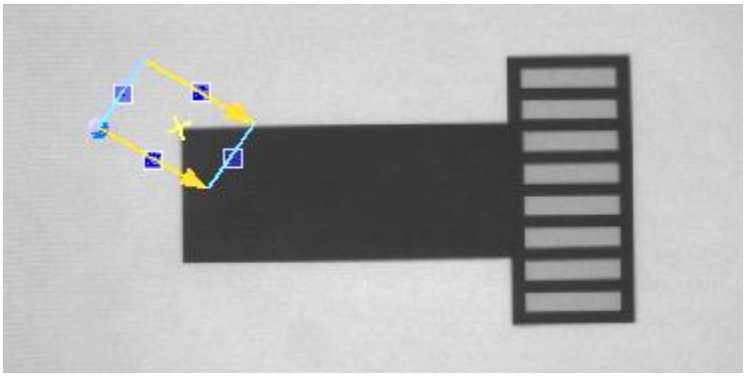
Special Feature Buttons (located above the property editor)

Adjust Threshold

Pressing this button displays a pop-up window that permits the value of the **Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. The display indicates the edges that satisfy the threshold criterion in white.

Examples

In the following example, this tool locates the edge point of the black object that is closest to the reference line that defines the start of the rectangular search region. The search is performed in the direction indicated by the tool's arrows. In this case, the closest edge is a corner point.

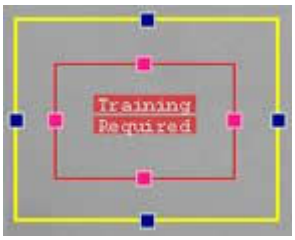


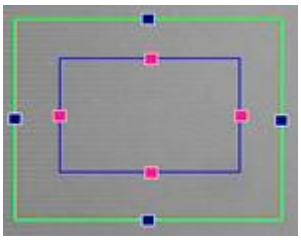
See Also

Vision Toolkit | **Edge Locator**

Finder Tool

Most powerful vision tool that identifies randomly placed parts in a camera image and returns their position, orientation and scaling to within sub-pixel accuracy.





Prerequisites

None

Properties

This tool employs a trained object model ("template") to recognize objects. To simplify the use of this tool, it has two Property Lists. The "**Template Settings**" Property List controls the automatic generation of the template and its properties. The "**Runtime Settings**" Property List defines how the template is applied to an image to locate objects. Special Feature buttons above the Property Lists switch between the two lists.

The following describes the "**Template Settings**" Property List. When a property is modified that invalidates the template, the system automatically indicates that Re-training is required.

Property Name	Data Type	Range	Description
1. Identity			
Type	String	n/a	Standard Identity properties
2. Placement/Size			
Height	**Single**	0 - AOI (mm)	Standard Placement / Size properties. The X and Y values define the center of the rectangular template and the Height and Width define its dimensions.
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
MinMaxAngle	**Integer**	0 - 180	This parameter defines the maximum amount that the template is permitted to rotate when it is locating objects. Normally, this property is set to 180 degrees to detect parts in any orientation. *However, if a part can only rotate a limited amount relative to its*

			taught orientation, reducing this value to the smaller range of orientations will speed up processing. In addition, if the part has some type of orientation symmetry (e.g. a circle or a square), this value should be set to restrict the range of orientations. This speeds processing and prevents a robot from making unnecessary changes in orientation to grasp a part. This property should be set as follows for common types of symmetries: Circle: 0 deg Square: 45 deg Rectangle: 90 deg Equilateral triangle: 60 deg Hexagon: 30 deg By default, this property is set to 180 degrees (any orientation).
ScoreAutoSetMin	**List**	NO / YES / THIS_TIME_ONLY	When a template is compared against a region in a camera image, the "recognition score" indicates how well the two match. A value of 0 represents a total mismatch and a value of 1 represents a perfect match. The **ScoreHiResMin** and **ScoreLoResMin** properties specify the minimum scores for a match to be considered good. The **ScoreLoResMin** is utilized at the lowest image resolution to determine if a possible match is a good enough candidate to be further analyzed at higher resolution levels. The **ScoreHiResMin** specifies if a match at the full camera resolution is acceptable as a final, good match. If **ScoreAutoSetMin** is set to YES, the **ScoreHiResMin** and **ScoreLoResMin** values are automatically set to computed conservative values when the template is trained. If **ScoreAutoSetMin** is set to THIS_TIME_ONLY, the **ScoreHiResMin** and **ScoreLoResMin** values are only automatically set the next time the template is trained. If **ScoreAutoSetMin** is set to NO, these two properties are not
ScoreHiResMin	**Single**	0 - 1	
ScoreLoResMin	**Single**	0 - 1	

			automatically set. In all cases, the values of **ScoreHiResMin** and **ScoreLoResMin** can be manually adjusted after training is completed. In general, the **ScoreLoResMin** is set equal to or lower than **ScoreHiResMin**. In fact, **ScoreLoResMin** can be set very low or even to 0. Since the **Finder** sorts all of the candidate matches at the lowest resolution and pursues the best candidates first, setting **ScoreLoResMin** low just results in more candidates being pursued to the full resolution level. If a specific number of parts is being located, having additional candidates is not a problem since the finder will stop when the desired number of parts are located. However, if an unknown number of parts is being searched for, having this property set too low can result in excessive execution time. On the other hand, the setting for **ScoreHiResMin** is very important in terms of the quality of the **Finders** results. Only matches that exceed this value are returned by this tool. Setting this property higher results in higher quality matches being returned at the risk of rejecting acceptable matches.
Speed	**Integer**	1 to 99	The **Speed** property controls how quickly the tool executes at runtime versus how meticulously it attempts to find matches. If this value is set to 1, the Finder executes more slowly but with the highest reliability. If it is set to 99, it executes at the fastest speed but may fail under non-ideal conditions. If **SpeedAutoSet** is set to YES, the **Speed** is automatically set to a conservative value that achieves a good balance of speed and reliability. If **SpeedAutoSet** is set to THIS_TIME_ONLY, the **Speed** value is only automatically set the next time the template is trained. If **SpeedAutoSet** is set to NO, the **Speed** is not automatically
SpeedAutoSet	**List**	NO / YES / THIS_TIME_ONLY	

			set. In all cases, the **Speed** can be manually adjusted but will typically invalidate the template.
3A. Advanced Operation			
Accuracy	**Integer**	1 - 100	This property controls the amount of effort and time that is spent on performing the final refinement of the location and orientation of a matched template. Smaller values result in less accuracy but faster execution times. Values above 10 provide higher sub-pixel accuracy at the expense of execution time. As this value is increased, additional key match points will be automatically added to the template. By default, this property is set to 10.
Depth	**Integer**	1 - n	As a significant speed optimization, at runtime the **Finder** initially searches for matches to the template model in images that have been reduced in resolution. Each level of reduced resolution is a factor of 2 smaller in width and height. This parameter is the maximum pyramid search level to use. During training, PreciseVision automatically computes the maximum number of levels to use based upon the loss of distinctive features and other criterion and sets this property value. After training, the **Depth** can be lowered to limit the number of reduced resolution levels considered. This is beneficial if the system is too aggressive and selects too many levels. The number of levels does not affect the accuracy of the final results of the **Finder** since the last match is always performed at the full image resolution. Higher levels will result in much fast execution times but may miss some possible matches. If **DepthAutoSet** is set to YES, the **Depth** is automatically set to the highest value that is appropriate for the template. If **DepthAutoSet** is set to THIS_TIME_ONLY, the **Depth** value is only automatically set the next time the
DepthAutoSet	**List**	NO / YES / THIS_TIME_ONLY	

			template is trained. If **DepthAutoSet** is set to NO, the **Depth** can be manually set to a lower value.
GainLimit	**Single**		To enhance robustness, each time the template is compared to a region of a camera image by the **Finder Tool**, the comparison is automatically normalized to eliminate differences due to the overall brightness of the image. In particular, each pixel value is effectively adjusted as follows: New_value = Gain*image_pixel + Offset This normalization process is controlled by the following properties: **GainLimit** restricts the range of the Gain value. If non-zero, the "Gain" will be restricted to fall between 1/**GainLimit** and **GainLimit**. By default, this property is set to 4. **OffsetLimit** restricts the amount by which the overall brightness is adjusted up or down, i.e. this property specifies the maximum absolute value of the "Offset". By default, this property is set to 64.
OffsetLimit	**Integer**	0 - n	
MaxScale	**Single**	0.8 to 1.2	These properties can be set to locate parts that have a variable scaling (size). This can occur if the part can be at different distances from the camera while still in acceptable focus. Allowing scaling variations can be very useful, but it does require additional execution time. By default, these values are set to 1.0, which indicates no scaling is permitted.
MinScale	**Single**	0.8 to 1.2	

The following list is displayed when the **"Switch to Runtime Settings"** Special Features Button is pressed. These properties control the runtime execution of the template matching operation.

Property Name	Data Type	Range	Description
1. Identity			

Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the rectangular search region and the Height and Width define its dimensions.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
PrecentClipped	**Integer**	0 - 50	This property allows parts to be recognized even if the matching template area is partially outside of the search region. Normally, this property is set to 0 and the **Finder** only searches for X/Y template center positions that have the template fully within the search area. In this case, this tool will not return a match position where the center of the template is closer than 1/2 the template's width or height to a boundary of the search region. If the **PercentClipped** is set to 50%, all template center positions within the search area will be considered. As this property value is increased, the recognition match score that is compared to **ScoreHiResMin** and **ScoreLoResMin** will be reduced somewhat, but not in proportion to the value of this property. Also, this property only applies to template matches that protrude beyond the search space and does not deal with occlusion due to overlapping parts. To allow for overlapping parts, the **ScoreHiResMin** and

			ScoreLoResMin should be reduced. By default, this property is set to 0.
ReTrain	**Boolean**	True / False	If you suspect that the recognition strategy for the Finder has become corrupted or invalid for any reason, this property can be set to True. When set, the original image that was used to train the Finder is automatically reloaded into the camera image buffer and the Finder is re-trained. This property can be manually or remotely set to True.
3A. Advanced Operation			
MaxFinds	**Integer**	-1 or 1 to n	This property is only valid if the Finder is called multiple times by a repeat tool such as a **Fixed Frame**, which is set to execute in a pattern. This property limits the number of matches found each time that the same Finder is executed by a repeat tool. It is similar in operation to **MaxResults**, but **MaxResults** limits the number of matches found for the entire image. For example, if **MaxFinds** is set to 2 and a **Fixed Frame** executes the Finder in 4 quadrants of an image, at most 2 matches will be found in each quadrant. On the other hand, if **MaxResults** is set to 2, after a total of 2 matches is found in any quadrants, the Finder will return no further matches in any subsequent quadrants. A value of -1 indicates that a unlimited number of matches should be found.
MaxRecognition Time	**Single**		Limits the total time the tool will execute, in milliseconds. This can be helpful if the Finder is looking for a specific number of objects (such as 1) but the

			expected objects may or may not be present. In this case, this property will stop the Finder before it exhaustively tests all possibilities. However, if this value is set too low, it can generate incorrect results due to the Finder not completing its full analysis and verification process. Normally, this parameter is set to its default value of 10000 (10 sec).
4. Inspection Settings			
InspectType	**List**	NONE / RESULT_ANGLE / RESULT_SCORE / RESULT_ID	Standard Inspection Settings properties
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties. **MaxResults** defines the number of matches to find each time that this tool is executed. This tool will execute more quickly if a known number of matches are located and this property is set accordingly. This is due to the fact that execution terminates as soon as the expected number of successful matches is found. Until all expected matches are found, this tool will continue to exhaustively test all possible matches. By default, this
OffsetAngle	**Single**	degrees	
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ResultSelect	**Integer**	1 - n	
ShowResults	**List**	NONE / FRAME / LINE / POINT	

			property is set to -1, exhaustive matching.
ShowResult-TemplateGraphics	**Boolean**	TRUE / FALSE	If TRUE, each matched part in the Camera Display is overlaid with dots that indicate the position of the key features of the Finder Template that are utilized for the object finding process. This permits the alignment of a matched part to be visually compared with the key features of the Template. The yellow dots indicate the primary pixels that should match prominent features of the full resolution image.
ShowResult-TemplateScore	**Boolean**	TRUE / FALSE	If TRUE, each matched part in the Camera Display is overlaid with its numerical "recognition score" that indicates how well the template matched the located part. A value of 0 represents a total mismatch and a value of 1 represents a perfect match. Accepted scores will all be greater than or equal to the **ScoreHiResMin** property value for the associated template. Especially for scenes with multiple instances of the same type of part, this feature is very useful for tuning the **ScoreHiResMin** value to correctly locate only valid parts.
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties
ResultErrorCode	**Integer**		
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultObjectID	**Integer**	1 - n	If a single **Finder** is executed multiple times within a Vision Process, this property identifies the instance of the **Finder** that produced each result. This is particularly useful when a

			Fixed Frame Tool has an index that steps a **Finder** around in a pattern. In this case, this property indicates the step in which each result was generated. *[GPL:VisResult.Info(4)]*
ResultRecognition-Time	**Single**	msec	Total time required for the last execution of this tool in milliseconds. *[GPL: VisResult.Info(3)]*
ResultScaleX	**Single**		These values indicate the X and Y scale factors that were computed when matching the template to the camera image. If scaling was not enabled, these values will be equal to 1. *[GPL: VisResult.Info(1), VisResult.Info(2)]*
ResultScaleY	**Single**		
ResultScore	**Single**	0 - 1	This is the "recognition score" that indicates how well the template matched the located part. A value of 0 represents a total mismatch and a value of 1 represents a perfect match. Accepted scores will all be greater than or equal to the **ScoreHiResMin** property value for the associated template. *[GPL: VisResult.Info(0)]*

Remarks

The **Finder** is the most powerful tool in the toolkit and provides the most general method for identifying and locating parts in grayscale images. It can locate randomly placed parts with both simple and complex shapes and will return their part type, position, orientation and scale. There are other tools in the toolkit that are appropriate for specialized situations such as binary images or non-rotated parts. Also, for some simplified or constrained cases, it is relatively easy to combine **Line Fitters** and other tools to locate parts. However, the **Finder** can handle the widest range of parts and situations, has a very simple one-shot teaching method, and returns accurate sub-pixel results.

To address applications like locating a secondary object within a primary object, the **Finder** can be placed relative to the results of any other tool including another **Finder**. The one restriction is that the **Finder** search window is always orthogonal with the camera frame of reference (although the matched objects can be in any orientation).

Typical uses for this tool include: finding fiducial marks on a printed circuit board, locating parts randomly placed and oriented on a conveyor belt and refining the position and orientation of sample trays.

Finder Templates and Training Mode

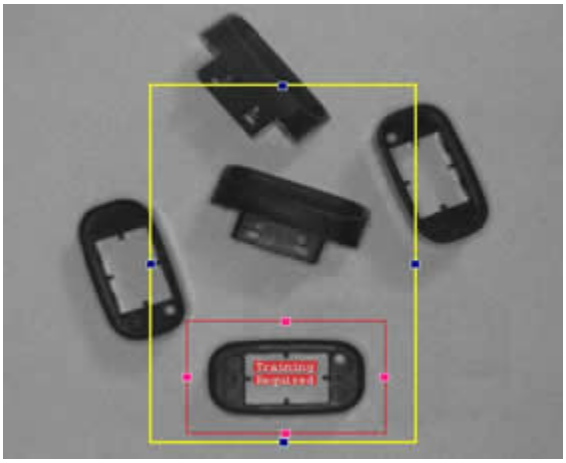
The **Finder** operates in one of two modes: "Training" or "Finding". Training mode teaches the system the visual features of a part and stores the information in a "Template" object model. Finding mode searches the camera image for regions of the image that are a suitable match for the Template.

When a **Finder** tool is first created, it is automatically placed into Training mode and the "**Template Settings**" Property List is presented. This list displays the size and the position of the Template and other properties that are specific to the type of part being taught, e.g. parameters for part symmetries, expected scale variations, etc. Presented above this property list are Special Feature buttons that aid in teaching and adjusting the template.

To create a Template, a single image of the part must be trained. The training process consists of bounding the part plus some of the adjacent background with a red "Template rectangle". This red rectangle defines the size and contents of the Template. When PreciseVision subsequently searches an image and reports the position and orientation of a matched part, it is actually stating the position of the center of the Template when the match was found and the amount by which the Template was rotated.

During the training process, a yellow "Training rectangle" must also be positioned around the Template. This yellow rectangle is used to teach the system how to distinguish the Template from other objects or background clutter that might be encountered at runtime. If different types of parts or clutter are expected to be present, the system will work best if one and only one full copy of the desired target part plus any different objects and clutter are enclosed by the yellow rectangle. In general, the larger the yellow rectangle, the better the Finder will operate since this allows the Finder to try more positions and orientations for the template as it optimizes its search parameters. Please note that since PreciseVision is a 2D vision system, different poses (stable states) of the same part that produce different 2D images are considered different parts.

The following picture illustrates how the training rectangles should be positioned to teach the **Finder** how to locate one orientation of a part while ignoring a second stable state.



If two or more types of parts are to be identified and located in the same image, multiple **Finders** and their Templates should be trained and included in the Vision Process.

After the Template and Training rectangles have been properly positioned, the **Finder** can be taught by pressing the **"Training Required"** special feature button located above the Template Properties list. To simplify the training and setup process, when you create a new **Finder**, a wizard is automatically displayed that walks you through the training steps. Using this wizard, performing the training and setting up the key **Finder** properties is extremely simple.

If any of the Template properties are subsequently modified that invalidate the Template, the system will automatically indicate that the Template must be trained again. To simplify retraining, the system always stores the image that was utilized during the prior training. This previous image can be easily recalled using one of the Special Feature buttons.

The Finder Search Algorithm

After a Template has been trained, the tool can be executed in Find mode and the alternate "Runtime Properties" list can be displayed using the Special Feature buttons. The Runtime Properties list presents the size and position of a green "Search rectangle". This rectangle defines the camera image area that the **Finder** will search for matches to the Template. The Runtime property list also contains parameters that control the search, e.g. the number of parts to find, the amount of clipping permitted, etc. In the Camera Display Window, the green Search rectangle and the blue "taught Template rectangle" will be displayed.

The heart of the **Finder Tool** is a patented algorithm that combines an intelligent search technique with the robustness of normalized correlation to match the Template with regions in the search region. This intelligent search technique avoids the computational requirements of applying correlation to a multi-dimensional search that accommodates concurrent changes in position, orientation and scale. Also, this technique does not introduce the problems encountered with correlation optimization methods such as

skipping pixels, while still retaining the important benefits of a normalized correlation match.

Like most search methods, this tool employs reduced resolution images for the initial search steps in order to quickly identify candidate positions. This method is often called a pyramid search since each successive search level has half as many pixels in each row and column. During the training process, this tool automatically computes the recommended number of levels that can be utilized while still retaining sufficient distinctive features to differentiate the target object from the background and other objects.

At each position where the template (or a reduced resolution template) is compared to the image (or a reduced resolution image), a normalized comparison operation produces a "recognition match score". This score ranges from 0, which indicates no match, to 1, which indicates a perfect match. At the reduced resolutions, all of the locations whose score exceeds a minimum value are retained and ordered according to their score value. The tool then pursues these candidate locations and validates whether the locations also satisfies a minimum match score when tested at the full camera resolution. When the system finds the requested number of successful results or all of the candidate locations have been exhausted, execution is terminated.

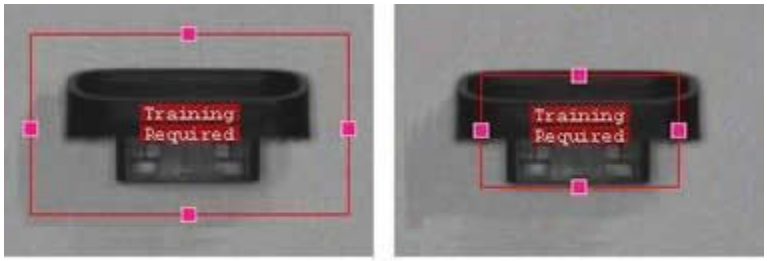
As a final step, if the Inspection Setting has been defined, each of the successful results will be tested against the Inspection criteria and will be tagged as to whether it "Passes" or "Fails".

Optimizing The Finder And Templates

The following are a number of guidelines for optimizing the performance of the **Finder** and **Finder Templates**.

Template (Red) Rectangle Guidelines

- The Template should fully enclose the target object and should contain no other objects or parts of objects.
- In general, the Template rectangle should be larger than the target object. It is best to have a boundary of at least 16 camera pixels all around the target object (more pixels are better than fewer pixels). For a 640x480 camera, 16 pixels is 1/30th of the full height of the camera image. This permits the system to capture the edges between the object and the background. This is especially important if the object does not have many internal features.

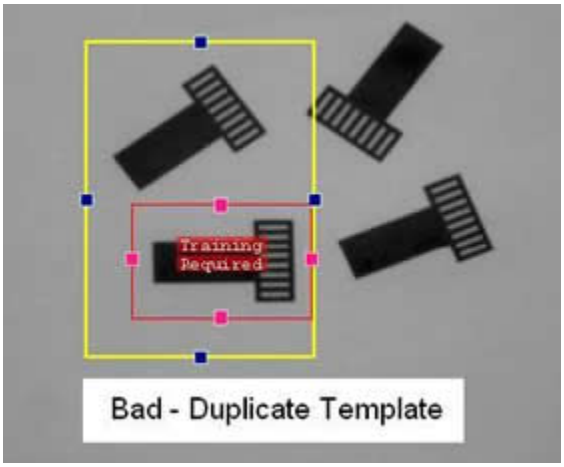


Good - partial border Bad - no border

- If the background varies significantly (e.g., there is dirt and paint blotches on a conveyor belt), then the part should be placed on a uniform, neutral background to teach the Template.
- The target object should be oriented in the camera image in what you want to define as 'Zero' degrees. That is, when the Finder later locates a part is in the trained orientation, it will return an orientation angle of 0 degrees.

Training (Yellow) Rectangle Guidelines

- The Training rectangle must be larger than the Template and must fully enclose the Template.
- The Training rectangle should include one and only one full image of the target object. Since the Finder uses the contents of the Training rectangle to determine distinguishing features of the Template, a second complete copy of the Template will confuse the system and will generate an error. It is acceptable to have additional partial images of the target object in the Training rectangle so long as each is less than 50% of the target object (see picture above in the Template discussion for an example).



- This rectangle should include any other objects or clutter that are expected to be encountered during runtime to aid the system in learning to distinguish between the target object and its surroundings.

- This rectangle should be as large as possible without violating the rules above. A larger region permits the system to test more positions and orientations for the Template during the training phase and assists in optimizing the runtime search strategy.

Finder Properties

- If your part is symmetric, e.g. a circle, a square, a rectangle, set the **MinMaxAngle** property to limit the rotation of the results.
- If the target object cannot be rotated into any arbitrary orientation, reducing the **MinMaxAngle** search range to less than the default value of +/-180 degrees will significantly reduce the execution time of this tool.
- If any portion of the part might extend beyond the search window, set the **PrecentClipped** property appropriately.

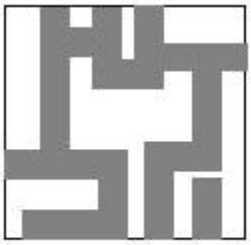
System and Camera Setup

- Prior to training or executing this tool, calibrate the camera's pixel to millimeter relationship. Failure to do so may result in parts not being located. This is due to the fact that some cameras do not have square pixels, i.e. their X and Y scaling factors are different. If this difference in scaling is not accounted for properly, as parts are rotated, their shape will be distorted and their size will be incorrect.
- Larger target objects (and therefore larger templates) generally perform better than small templates for various reasons: finding a very small template in a large search space is comparable to finding a needle in a haystack; larger templates usually have more image features; larger templates usually can be found with higher accuracy particularly in orientation; and larger templates containing some big features can be searched using taller search pyramids thereby accelerating the search.

Optimal Target Objects

- Target objects that contain corners that form a 90-degree angle result in more reliable recognitions and higher accuracy.
- Sharp features in focus provide more accurate results than smooth or blurred features.
- If the template has many edges in one direction but only a few in the orthogonal direction, matched locations will have greater accuracy in the direction perpendicular to the majority of the edges. In the figures below, templates "A" and "B" contain many vertical edges so both can be located with high accuracy in the horizontal direction. But template "A" has very few horizontal edges compared to "B", so along the vertical direction, "B" will be found with higher accuracy than template "A".


Template A (poor)


Template B (good)

- A wide dynamic range of brightness in the template provides more accurate results than an image with a narrow range. However, avoid saturation. Image information is lost when pixels are clipped at the two ends of the brightness range.
- Fine details in the image enhance the overall accuracy of the position calculations, but large features enhance the speed of search. Large features enable use of taller search pyramids.

Template Mask

- If the part contains features that are known to be variable from one image to the next, the unwanted features can be ignored by masking them out of the template. The masking operation is similar to using a simple painting tool to indicate regions to be ignored in the template.
- To define a Template mask, force the system into Training mode, right click in the camera display window or in the Vision Tool Definition window and select Finder > Edit Template.

Special Feature Buttons (located above the property editor)

Execute

Executes the Finder search algorithm once in the current camera image buffer.

Load Original Image

Each time a Template is trained, the image used for the training is automatically stored by the system. This button reloads the previous training image into the camera button. This greatly simplifies retraining the template if you wish to see the affect of altering various Template properties.

Reset Template Defaults / Undo Changes

Either resets the Template Property values to their default values, which were established when the Template was trained, or undoes any recently changes to the property values.

Show Template

Pops up a window that displays the Template that has been trained. The pixels in the Template that represent key features to be located are highlighted in yellow. If the Template includes a mask for ignoring unwanted areas, the mask is displayed in red.

Switch to Template Settings / Switch to Runtime Settings

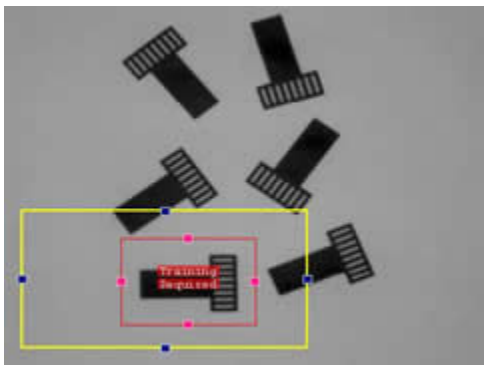
Switches between the Template Settings and the Runtime Settings Property lists.

Training Required

Indicates that the Template for the Finder is not valid. Pressing this button will train the Template.

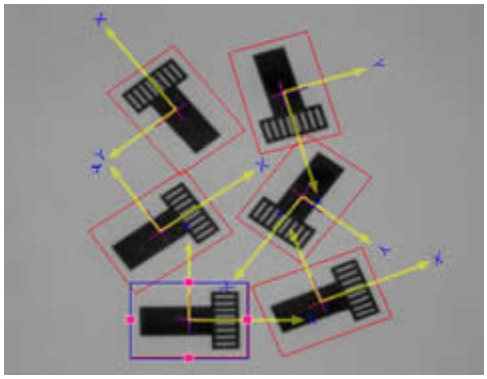
Examples

In the following example, a **Finder Template** is trained on one of six identical parts. Since the parts are all the same, the yellow Training rectangle is only placed around one complete part although other partial parts (each less than 50% of the full object) are included in order to enlarge the Training rectangle. Also, for a reasonable template boundary and for an intuitive orientation angle, the horizontal part is trained rather than the parts at a skewed angle. After the Training and the Template rectangles are positioned, the Train icon can be pressed.



For this simple part, training completes in approximately one second after which the tool is automatically placed into Find mode and the search operation is executed once. At the conclusion of the search, all six parts will be identified and each of the part's position and

orientation will be indicated by yellow coordinate axes as illustrated in the following picture.

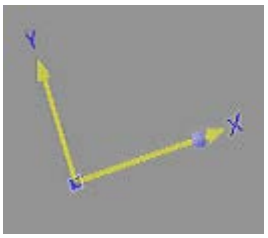


See Also

Vision Toolkit | **Connectivity**

Fixed Frame Tool

Computational tool that places a reference frame at a fixed image coordinate or at a constant offset relative to another vision object. Can optionally index the frame in an X and/or Y grid pattern to repeatedly execute any linked tools.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the origin of the reference frame and the Angle specifies its rotation.
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
3A. Advanced Operation			
FramePatternXIndex	**Integer**	1 - n	If the X and/or Y index values are greater than 1, the position of the Fixed Frame is indexed in a grid pattern the number of X and/or Y steps by the appropriate Pitch values. This permits any linked tools to be automatically and repeatedly executed in a grid pattern.
FramePatternYIndex	**Integer**	1 - n	
FramePatternXPitch	**Single**	mm	
FramePatternYPitch	**Single**	mm	
TargetDisplaySize	**Single**	0 - n	

TargetDisplayType	**List**	NONE / SQUARE / CIRCLE	Optionally draws an overlaid graphics shape relative to the position and orientation of the Fixed Frame. This is useful for highlighting a specific region in the image. The "size" defines the diameter or edge length of the shape in calibrated units.
5. Results Settings			
ShowResults	**List**	NONE / FRAME / LINE / POINT	The frame can be graphically represented as a coordinate frame, line or a point.
6. Results			
ResultAngle	**Single**	-360 to 360	Standard Results properties
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

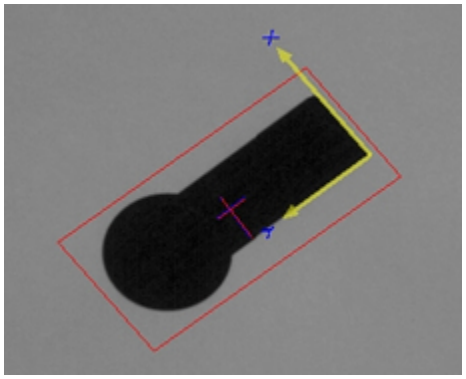
A Fixed Frame can be used in several different ways. It can be defined relative to another vision object with a fixed position and angular offset. This is an alternative to directly setting an offset in the Results Settings of most tools and has the advantage of allowing the offset to be graphically specified by dragging the Fixed Frame in the vision window.

Alternately, a Fixed Frame can be placed in an image at a specified position and orientation and can then serve as a datum from which other measurements are made.

Lastly, if the Pattern properties are utilized, a Fixed Frame can be automatically indexed about an X and/or Y grid pattern. This has the benefit of repeatedly executing any tools linked to (i.e. defined Relative-to) the Fixed Frame in a grid pattern.

Examples

Below is an example of a Finder that returns a frame that is at the center of the part. A Fixed Frame is placed relative to the finder and used to effectively re-define the part's frame of reference. Additional vision tools can now be placed relative to the Fixed Frame. The advantage of doing this is that if the Finder template is re-trained it may not be placed in exactly the same position and orientation relative to the part. The Fixed Frame can then be adjusted to account for any difference in the Finder position and orientation. By using this relative reference frame, we can easily ensure that all subsequent tools will maintain their positional relationship with respect to key features of the part.

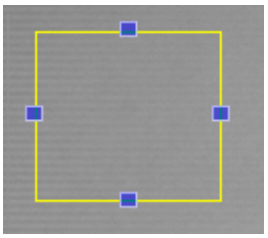


See Also

[Vision Toolkit](#) | [Line-Line Frame Tool](#) | [Point-Line Frame Tool](#) | [Point-Point Line Tool](#)

Image Process Tool

Vision tool that performs image filtering, edge extraction, thresholding and morphological operations on a specified area in the camera image.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	0 (*non-rotating*)	Standard Placement / Size properties. The X and Y values define the center of the region to be processed. The Height and Width define the dimensions of the region.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	0	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
InvertBinary	**Boolean**	TRUE / FALSE	(BINARY) For BINARY_* **ProcessModes**, if this parameter is True, after the binary pixels values are extracted, their values will be inverted before the remainder of the processing is performed.

Iteration	**Integer**	1 - n	Specifies the number of times that the **ProcessMode** should be applied to the area of interest. This is often used to repeatedly execute the morphological operations.
ProcessMode	**LIST**	CONV_* / BINARY_*	Specifies the image processing operation to be performed on the selected area-of-interest. These are destructive operations that alter the contents of the camera buffer so that all subsequent tools operate on the modified data. See the Remarks for an illustration of each of the supported operations.
Threshold	**Integer**	0 - 254	(BINARY_*) This is the value used to convert gray scale values into black and white pixels. (CONV_*) This value is not utilized. (EDGE_*) Specifies the threshold below which weak (low contrast) gradient edges are ignored. The lower this property is set, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
3A. Advanced Operation			
ConvolutionSize	**List**	SIZE_3X3 / SIZE_5X5	Specifies whether the convolution kernel is a 3x3 or a 5x5 matrix. The size of the kernel determines how many neighboring pixels are consider in the computation of each new pixel value.
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	

ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	0 (non-rotating)	
ResultXPos	**Single**	mm	X and Y position for the centroid of the processed area-of-interest.
ResultYPos	**Single**	mm	

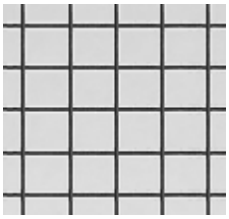
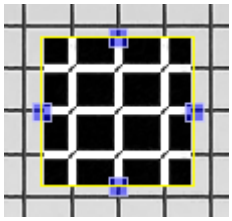
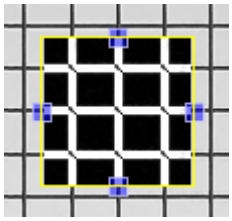
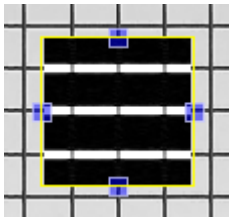
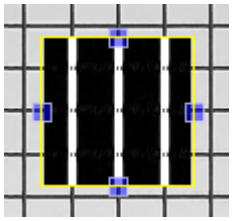
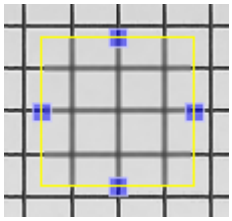
Remarks

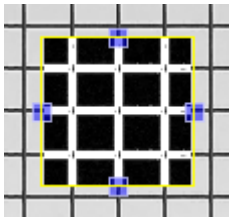
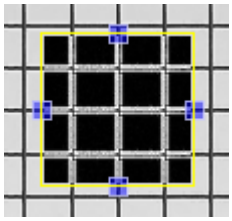
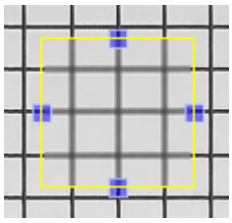
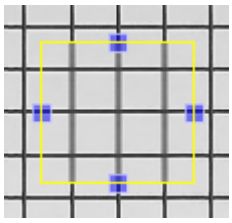
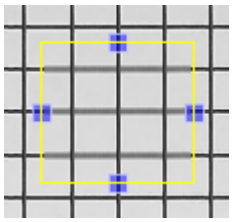
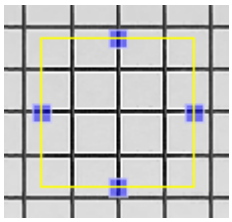
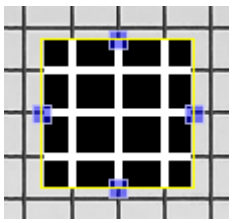
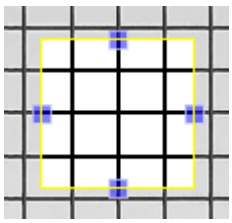
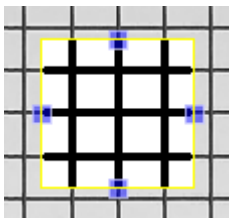
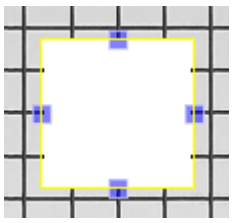
This tool performs an image filtering, edge extraction, thresholding or morphological operation on a specified area in the camera image. For example, an operator can be applied to smooth a region to eliminate noise due to poor lighting. Conversely, the image can be sharpened to highlight features.

While most vision tools analyze the data in the camera image and leave the data unmodified, these operators all alter the raw data in the image buffer. All subsequent vision tools will operate on the modified image data and not the raw information captured by the camera.

If you wish to revert back to the original raw captured image data, special properties must be set in the **Acquisition Tool** at the time that the picture is first taken.

The following table illustrates each of the image processing operations.

	Raw, unprocessed camera image.		CONV_FILTER_EDGE_DL Extracts edges with a bias for edges that slope down from the left.
	CONV_FILTER_EDGE_DR Extracts edges with a bias for edges that slope down from the right.		CONV_FILTER_EDGE_H Extracts edges with a bias for horizontal edges.
	CONV_FILTER_EDGE_V Extracts edges with a bias for vertical edges.		CONV_FILTER_GAUSSIAN Reduces image noise by blurring the region. This is a smoothing operator.

	CONV_FILTER_ LAPLACIAN Extracts edges using a Laplacian operator.		CONV_FILTER_ LAPLACIAN_ GAUSSIAN Extracts edges after smoothing with a Gaussian to reduce noise.
	CONV_FILTER_ LOWPASS Reduces image noise by eliminating high frequency changes. This is a smoothing operator.		CONV_FILTER_ LOWPASS_H Reduces image noise by eliminating high frequency changes but only in the horizontal direction. This is a 1D smoothing operator.
	CONV_FILTER_ LOWPASS_V Reduces image noise by eliminating high frequency changes but only in the vertical direction. This is a 1D smoothing operator.		CONV_FILTER_ SHARP Accentuates or enhances edges in the image.
	EDGE_THRESH Computes each pixel's edge strength and sets all strong edges to white and all weak edges to black.		BINARY_THRESH Converts all pixels to white or black.
	BINARY_ERODE Binary thresholds the region and shrinks white groups of pixels.		BINARY_DILATE Binary thresholds the region and grows white groups of pixels.

Special Feature Buttons (located above the property editor)

Adjust Threshold

Pressing this button displays a pop-up window that permits the value of the **Threshold** property to be easily adjusted while dynamically viewing the effect of the new setting in the Camera Display. In a BINARY_* mode, the displayed image shows all pixels converted to black or white. In an EDGE_* mode, the edge pixels are extracted and displayed in white.

See Also

Vision Toolkit | **Acquisition Tool**

Inspect Frame Tool

Inspection tool that passes through the XPos, YPos and Angle Results of a referenced tool and optionally rotates and shifts the Results based upon an inspection. The inspection can consist of the arithmetic combination of the Results of two other tools. So, this tool also provides a means for performing simple mathematical operations on the results of two tools.

Prerequisites

FrameSource must be defined for this tool to execute.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
ApplyFrameOffset	**List**	NONE / OFFSET_ON_PASS / OFFSET_ON_FAIL / OFFSET_ALWAYS	The nominal Results of this tool are equal to the Results of the specified **FrameSource**. This property specifies when the **OffsetX**, **OffsetY**, and **OffsetAngle** are added to the nominal Results. For example, this tool can rotate the orientation of the output of the **FrameSource** by 180 degrees based upon the pass/fail outcome of a specified inspection operation.
FrameSource	**List**	n/a	Name of the *required* tool whose **ResultsXPos**, **YPos** and **Angle** are taken as the nominal Results of this tool.
InspectTool1	**List**	n/a	If the **InspectType** specifies an arithmetic
InspectTool2	**List**	n/a	

			computation that operates on the Results of two tools, these two optional properties specify the tools that are the sources for the Results.
3A. Advanced Operation			
SortResults	**List**	NONE / SORTFROMLEFT / SORTFROMRIGHT / SORTFROMTOP / SORTFROMBOTTOM	If the **FrameSource** generates multiple Results, this property automatically sorts the Results as specified. This is useful for items on a conveyor belt that may need to be picked in order from the front to back.
ZeroAngle	**Boolean**	TRUE / FALSE	If True, will force the **ResultAngle** to ZERO. This is useful for randomly oriented parts that require the angle to be removed for picking. This operation does NOT change the sorting operation.
4. Inspection Settings			
InspectType	**List**	NONE / INSP1_MINUS_INSP2 / INSP1_PLUS_INSP2 / INSPECTIONS_PASS / INSPECTIONS_FAIL / INSPECT_XY_DISTANCE / INSPECT_X1_MINUS_X2 / INSPECT_Y1_MINUS_Y2 / INSPECT_X1_PLUS_X2 / INSPECT_Y1_PLUS_Y2 / INSPECT_X1_MULT_X2 / INSPECT_Y2_MULT_Y2 / INSPECT_X1_DIV_X2 / INSPECT_Y1_DIV_Y2 / INSPECT_ANG1_MINUS_ANG2 / INSPECT_ANG1_PLUS_ANG2 / INSPECT_ANG1_DIV_ANG2 / INSPECT_ANG1_MULT_ANG2	Standard Inspection Settings properties. The **InspectType** can specify a simple arithmetic operation that is to be performed on the Results obtained from **InspectTool1** and **InspectTool2**. The **InspectActual** returns the result of the arithmetic operation and the **InspectPassed** is set according to whether the result falls within the specified Min and Max values.
InspectMax	**Single**		
InspectMin	**Single**		

InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
InspectEval	**String**		Displays the arithmetic expression that is evaluated to compute the **InspectActual** results. This expression is a function of the **InspectType**.
InspectString	**String**	n/a	Optional label that is displayed in the camera window. This provides a convenient way to display part ID or other information.
5. Results Settings			
MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties. The Offsets are conditionally added to the nominal **ResultsXPos**, **YPos** and **Angle** depending upon the setting of **ApplyFrameOffset** and the pass / fail status of the inspection operation.
OffsetAngle	**Single**	degrees	
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ResultSelect	**Integer**	1 - n	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties. The **ResultsXPos**, **Ypos** and **Angle** are equal to the Results of the **FrameSource**, optionally adjusted by the **OffsetX, OffsetY, OffsetAngle**.
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

This tool passes through the **ResultXPos**, **ResultYPos**, and **ResultAngle** from a referenced **FrameSource** tool and optionally shifts and rotates the Results based upon a pass/fail test. This function can be used to correct the position and/or the orientation of an object based upon an additional inspection operation. If the referenced tool produces multiple Results, the test and correction will be applied to each generated Result.

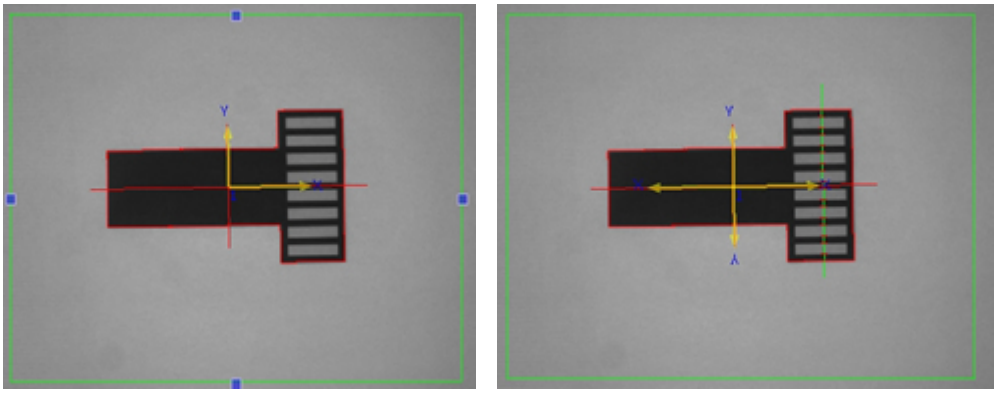
This tool can also be used to perform measurements between the Results of two vision tools. For example, the **InspectType** property can specify that the distance between the XY positions of **InspectTool1** and **InspectTool2** is to be computed and compared to

specified min/max values. The outcome of this inspection determines the pass/fail status of this tool. This pass/fail value can trigger the offsetting of the Results or can be used as input to other tools.

For example, when a **Connectivity Tool** is executed, the orientation of its Result is always ambiguous since this algorithm will return the same orientation and position even if an object is rotated by 180 degrees. If the **Connectivity Tool** is specified as the **FrameSource** of an **Inspect Frame Tool**, a measurement of a specific feature of the object can be performed. Based upon the measurement, the frame of reference that is generated by the **Connectivity Tool** can be conditionally rotated to correct its orientation.

Examples

Below is an example of a **Inspect Frame Tool** that alters the orientation of a **Connectivity Tool** output based on the inspection of an **Edge Locator Tool.** The **Edge Locator Tool** measures the width of the part (along the green line shown below in the picture on the right) by determining the distance between the first and last located edge. Depending upon whether the wide or narrow portion of the part is detected, the **Inspect Frame Tool** conditionally applies an angular offset to rotate the returned frame's orientation angle.



See Also

Vision Toolkit | **Inspect List**

Inspect List Tool

Inspection tool that passes through the XPos, YPos and Angle Results of a referenced tool and tests up to 12 different Results from multiple tools to determine if the object is good or bad. This tool provides a convenient way to perform complex testing of a located object and generating an equivalent location with a single Pass/Fail indicator.

Prerequisites

FrameSource must be defined for this tool to execute.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
FrameSource	**List**	n/a	Name of the *required* tool whose Results XPos, YPos and Angle are passed through as the position and orientation Results of the Inspect List tool.
InspectTool01-12	**List**	Note: Accessed using "Setup Inspection List" button.	List of vision tools whose Results are tested to determine if the part passes or fails. Many vision tools (i.e. connectivity) have multiple Results. Such tools can be referenced multiple times to test different Results property values.
3A. Advanced Operation			
ShowInspectTools	**Boolean**	TRUE / FALSE	If True will display the tools defined in the Inspect List during execution, otherwise only the resulting output frame is displayed.
4. Inspection Settings			
InspectPassed	**List**	Pass / Fail	Standard Inspection Settings properties.
InspectString	**String**	n/a	Optional label that is displayed in the vision window relative to the position and orientation of this tool. This provides a convenient way to display part ID or other information.
5. Results Settings			

MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties. For this tool, the ***ResultOnNotFound*** *should normally be enabled* on the Preferences > Tool Operation page. This property forces all tools to generate results even if their inspections fail. This ensures that the **Inspect List** has a consistent set of complete results from all referenced tools. If this property is not enabled, it is possible for referenced tools to past back a varying number of results depending which inspections pass and fail, with the result that the **Inspect List** will be confused and will not return an accurate good/bad indicator.
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ResultSelect	**Integer**	1 - n	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties. The XPos, YPos and Angle Results are equal to those of the **FrameSource**. As a special case for this tool, *GPL: VisResult.Info(0-11)* are equal to the Pass(-1) or Fail(0) results of **InspectTool01-12**. *GPL: VisResult.Info(12-23)* are equal to the actual values used for the pass/fail tests for **InspectTool01-12**.
ResultErrorCode	**Integer**		
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

Many vision applications only required one or two tests to determine if a part is acceptable. In these cases, the vision inspection tools can be executed directly in a Vision Process. However, if many tests must be performed, the **Inspect List** provides a very convenient method for executing up to 12 tests on the Results from multiple tools and generating a single Pass/Fail result that indicates if all of the tests are satisfied. This tool can also simplify and improve the efficiency of a GPL program that accesses vision results. If the GPL program only needs to know that all tests passed, the **Inspect List** tool can return a single value instead of requiring the GPL program to read the results from multiple tools.

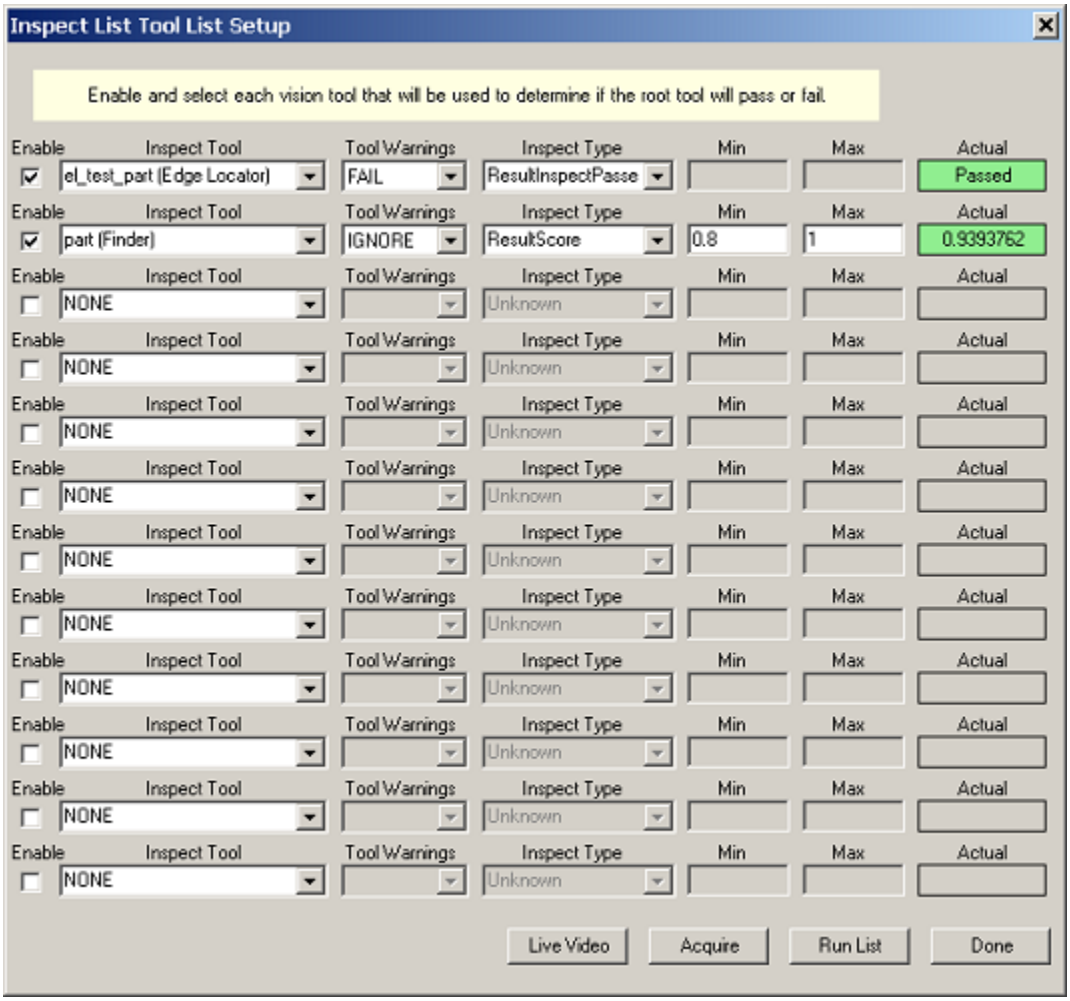
As a further convenience, the **ResultXPos**, **ResultYPos** and **ResultAngle** of this tool are set equal to the Results of the specified **FrameSource**. So, if **FrameSource** is a tool that locates the position of an object, the output of the **Inspect List** will provide the location of the object together with a single Pass/Fail property that indicates if up to 12 qualifying inspections were successful.

The 12 Results tested by this tool can come from a single vision tool or multiple tools. This permits the Results from multiple measurement or inspection tools to be combined. Alternately, for tools such as Connectivity that return many independent properties (e.g. **ResultBlobArea, ResultMajorAxisLength**, **ResultMinorAxisLength**, **ResultMaxRadiusAngle**, etc.), multiple properties from the same tool can be tested to verify that the part is acceptable.

Special Feature Buttons (located above the property editor)

Setup Inspection List

Since this tool has the ability to reference the Results properties from up to 12 other tools and each referenced tool can have properties that are specific to its tool type, a special pop-up dialog is provided to specify the 12 possible tests. The following is an example of this dialog page that is accessed by pressing the "Setup Inspect List" button above the tool's property list.



Each tool to be accessed is selected using the pull-down combo box "Inspect Tool". Once a tool is specified, the Results property to test is selected using the "Inspect Type". All tools support the same base set of inspection types: **ResultInspectPassed** / **ResultInspectFailed** / **ResultXPos** / **ResultYPos** / **ResultAngle**. Some tools also have special Results that can be tested. Please see the "Info()" array of the selected vision tool for details as to its tool specific Results properties.

In addition to returning a Pass/Fail inspection result, many tools can return various "warnings". In most cases, warnings can be ignored and the tool's inspection result can be interpreted as "passed". For example, the Line Fitter Tool returns a warning if some edge samples are missing even if the line fit is very accurate. In other cases, you might wish to treat warnings as "fail" conditions. For example, if you are using a Pixel Window Tool for inspecting a part and the window is clipped, it might be important for you to consider this as a failure. To control the treatment of

warnings, for each Inspect Tool, there is a "Tool Warnings" property that can be set to FAIL or IGNORE.

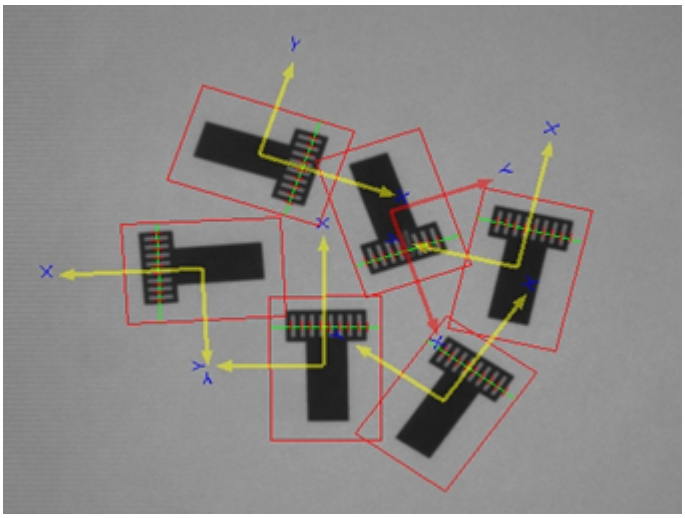
Once the Inspect Type is selected, the Min/Max inspection limits can be filled in. The retrieved property values are displayed in the Actual fields and the values are color coded to indicate if they fall within the Min/Max bounds.

At the bottom of this dialog panel, there are buttons to assist in setting up and testing the tool. When the "Run List" button is pressed, all of the referenced tools are executed and their properties are evaluated.

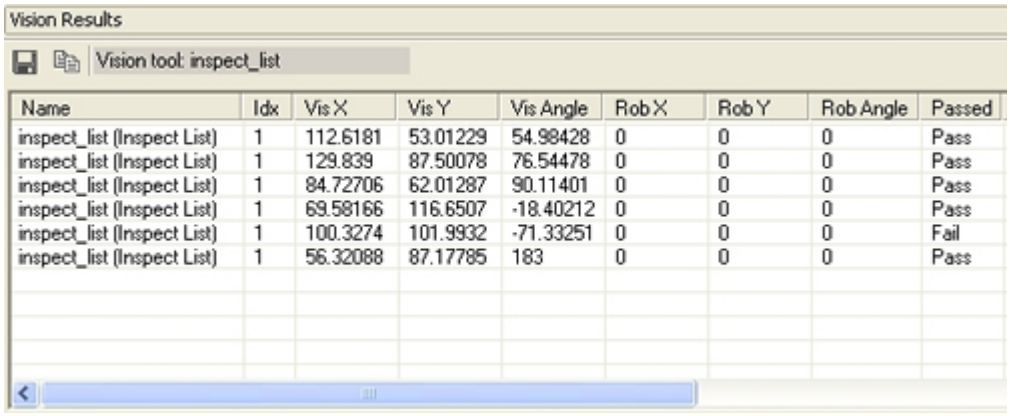
Examples

Below is an example of a **Inspect List** tool that is used to locate and qualify multiple parts in the camera's field-of-view. The **FrameSource** is a **Finder** tool. There is a **Edge Fitter** tool that is placed relative to each located object returned by the **Finder**. The **Edge Fitter** counts the transitions on each part. If the count is wrong, the part will fail and will not be picked.

In this example, a part near the center failed since it is missing one of its slots.



Even though multiple tools were utilized to locate and inspect each part, only a single tool result is passed to the connected robot controller for each part. This simplification is especially useful for more complex inspection requirements and when the number of located parts is large.



Name	Idx	Vis X	Vis Y	Vis Angle	Rob X	Rob Y	Rob Angle	Passed
inspect_list (Inspect List)	1	112.6181	53.01229	54.98428	0	0	0	Pass
inspect_list (Inspect List)	1	129.839	87.50078	76.54478	0	0	0	Pass
inspect_list (Inspect List)	1	84.72706	62.01287	90.11401	0	0	0	Pass
inspect_list (Inspect List)	1	69.58166	116.6507	-18.40212	0	0	0	Pass
inspect_list (Inspect List)	1	100.3274	101.9932	-71.33251	0	0	0	Fail
inspect_list (Inspect List)	1	56.32088	87.17785	183	0	0	0	Pass

See Also

Vision Toolkit | **Inspect Frame**

Inspect Region Tool

Inspection tool that inspects the perimeter and the interior of an opaque object to determine if there are any flaws in the object's outline or holes in the interior. This tool was designed to inspect wafers, where slight changes in the overall size and shape of the wafer are permitted, but chips in the perimeter or interior holes or cracks are unacceptable.

Prerequisites

FrameSource must be defined for this tool to execute. Multiple source tools must be created that define the perimeter of the object to be inspected.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
FrameSource	**List**	n/a	Name of the *required* tool whose **ResultsXPos**, **YPos** and **Angle** are taken as the nominal Results of this tool.
PerimeterTool01-12	**List**	n/a	Series of up to 12 vision tools whose results define the outline of the region to be inspected. These tools are normally **Line Fitters**, **Arc Fitters** or computed line tools. These tools must be listed in sequence as the perimeter is traversed in a counter-clockwise direction. The located lines and arcs plus the points of intersection that are automatically computed between adjacent tools define a mask that is applied to the image to detect flaws. The resulting

			mask must be a closed convex shape.
SetMinBlobArea	**Integer**	0 - n	Specifies the minimum number of connected binary or edge pixels that a flaw must contain in order for it to be considered a defect. This test is applied both to perimeter and interior flaws. Any flaw that consists of a smaller number of pixels will be ignored.
SetNumOfErosions	**Integer**	0 - n	After the mask is dynamically generated from the results of the **PerimeterTools**, the size of the mask can be reduced by a few pixels via an iterative erosion operation. If the outline of the region to be tested in the image is somewhat rough, this permits small errors in the outline to be ignored.
SetThreshold	**String**		Threshold that is used when the **ImageMode** is Binary to convert gray scale values into black and white pixels. For EdgeLevel mode, this specifies the threshold below which weak (low contrast) gradient edges are ignored. The lower this property is set, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
3A. Advanced Operation			
ImageMode	**List**	EdgeLevel / Binary	Specifies if the region of interest is to be analyzed as a binary or an edge image.

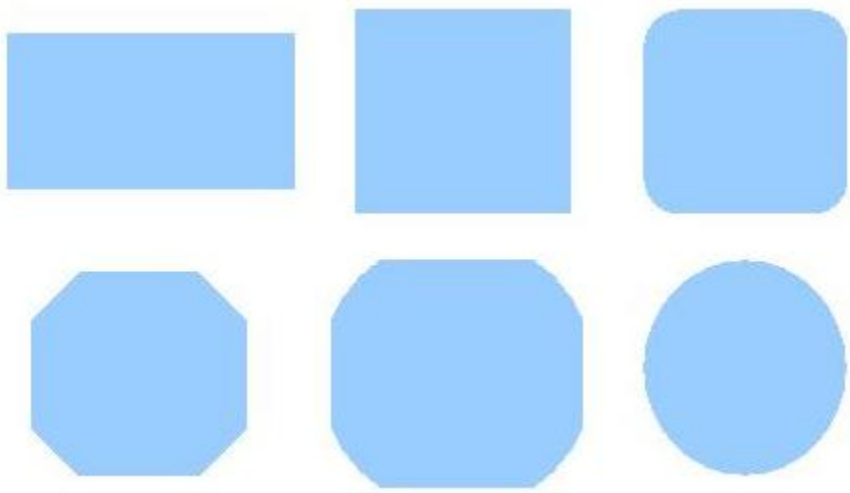
ProcessMode	**Boolean**	NORMAL / LINE_ARC_BOUNDARY / FLOODED_REGION / ERODED_IMAGE / EDGE_IMAGE / THRESHOLDED_IMAGE / ANDED_IMAGE / ANDED_IMAGE_WITH_BLOB	This is a setup / debugging aid. It permits the tool to be stopped at the completion of the specified processing step so that the results of the step can be viewed in the Camera Window. If this property is not set to Normal, the results of this tool will not be valid since the tool will not complete its full analysis.
ShowPerimeterTool	**Boolean**	TRUE / FALSE	If TRUE, the Results of each of the **PerimeterTools** will be displayed. These results can sometimes be useful to see, but at other times, can clutter the Camera Window.
4. Inspection Settings			
InspectActual	**Single**		Standard Inspection Settings properties. The **InspectActual** returns the number of defects that are detected. If no defects are found, the InspectPassed is set to Pass.
InspectPassed	**List**	Pass / Fail	
InspectString	**String**	n/a	Optional label that is displayed in the Camera Window. This provides a convenient way to display part ID or other information.
5. Results Settings			
MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties.
OffsetAngle	**Single**	degrees	
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultSelect	**Integer**	1 - n	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties.
ResultErrorCode	**Integer**		
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	

ResultYPos	**Single**	mm	

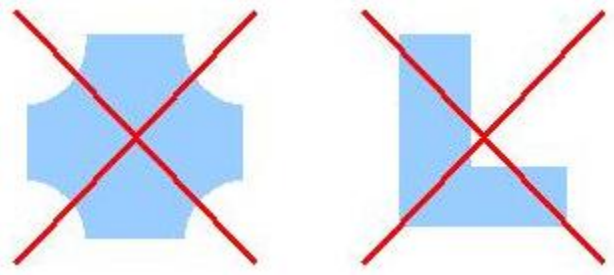
Remarks

This is a specialized inspection tool that is designed to locate chips in the perimeter of a smooth convex outline and interior holes and cracks. This was developed to inspect for defects in wafers, whose size and outline might vary slightly from one wafer to the next. Since acceptable wafers can vary in their exact size and shape, a fixed mask cannot be applied to inspect the perimeter. Instead, a mask is dynamically generated for each image using the results of **Line Fitter**, **Arc Fitter** and computed line tools. These tools can locate the actual perimeter to be inspected and can take into account expected variations in the outline of the wafer to be inspected.

This tool can operate on any convex shape whose outline can be defined by 12 lines and arc. An example of acceptable shapes are as follows:



While this tool could be extended to handle outlines with concave features, the following shapes are currently not supported.



This tool dynamically generates a mask to compare against the image using the results of up to 12 lines and arcs. The intersection of the lines and arc are automatically computed to bound the mask. Therefore, the lines and arc must be specified in the **PerimeterTools** list in the order in which they appear as you trace around the perimeter in a counter-clockwise direction. After the mask has been created, it can be shrunken if desired by a series of erosion operations. If the object to be tested has slightly rough edges, this will eliminate a few pixels along the perimeter from consideration.

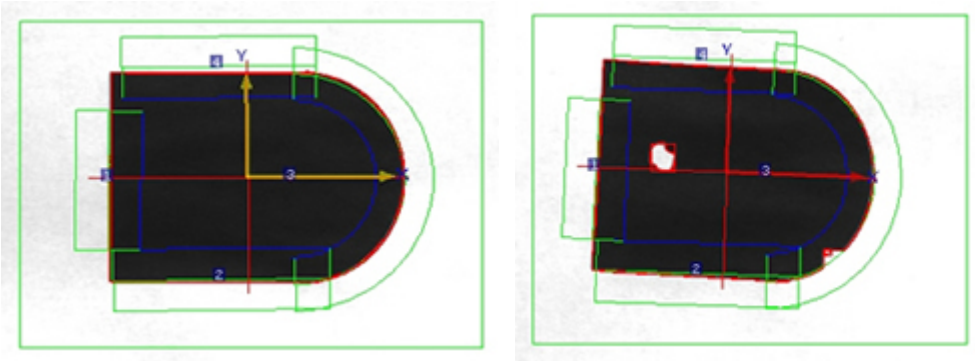
The region to be tested is then converted to either a binary or an edge image and the mask is AND'ed with the converted region. Any flaws along the perimeter or in the interior of the region will be left as non-zero pixels. A connectivity algorithm is next applied to identify any collection of non-zero pixels that are large enough to constitute an important defect.

The number of detected defects are counted and if any defects exist, this tool returns with a failure indication.

The sensitivity of the tool can be adjusted using the **SetMinBlobArea** and **SetNumOfErosions** properties. These parameters control the minimize size of flaws that are considered defects and how much the mask is shrunk before it is applied.

Examples

Below is an example of a **Region Inspect** tool that detects missing regions along the perimeter of a part. On the left is a good part that passes. On the right is a part that fails due to a missing region along an "arc" and an interior defect. Any detected defects are automatically highlighted by a bounding red rectangle.

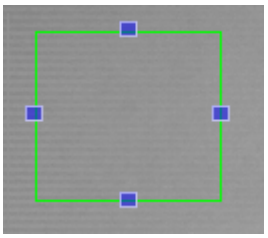


See Also

Vision Toolkit | **Line Fitter** | **Arc Fitter**

Light Recorder Tool

Specialized vision tool that acquires a sequence of camera images for a specified period of time, determines the brightness of each image, and returns the brightness readings in an array for subsequent analysis.



Prerequisites

Utilizes the camera properties (for example the **ExposureTime**, **VideoGain**, **VideoOffset**, etc.) defined in the previously executed Acquisition Tool for capturing the images required by this tool.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Height	**Single**	0 - AOI (mm)	Standard Placement / Size properties. The X and Y values define the center of the tool. The Height and Width specify the size of the image region being tested. If the referenced camera supports hardware AOI to permit faster frames per second rates and PreciseVision implements AOI for the camera, this tool will automatically utilize the camera's AOI capability to increase the maximum frame rate. Note, some camera's maximum frame rate occurs when the AOI is larger than a few pixels in each direction, so some experimentation with the AOI size might be require if the fastest frame rate is desired.
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
3. Operation			

Camera	**Integer**	1 - 6	Standard Operation property
GrayMode	**List**	MEAN / MAX / MIN	Specifies the type of result to be computed and returned for each frame.
DesiredSampleRate	**Single**	0 - n (msec)	Specifies the desired amount of time (in msec) between each acquired image. The actual time will vary depending upon the performance of the camera and the PC on which PreciseVision is executed. If this property is set to 0, images will be acquired as quickly as possible.
RecordTime	**Single**	0 - n (seconds)	Defines the period of time during which images are acquired, in seconds. The actual results returned are limited by how many values can be transmitted to the attached motion controller. Currently, at most 3000 sample results can be returned.
5. Results Settings			
ShowResults	**List**	NONE / FRAME / LINE / POINT	Standard Results Settings property
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultActualTime	**Single**	sec	Actual time in seconds for the operation to complete.

Remarks

This vision tool captures a sequence of images at a specified target rate (in msec), computes the mean, maximum or minimum brightness of each image, and returns the results in an array of values. This tool was originally designed to analyze the blinking pattern of an LED. However, it could also be used as a time based proximity sensor or light level detector.

While this function could be implemented with a combination of other vision tools, this tool is able to achieve higher image acquisition rates due to reduced overhead between image captures. The image capture rate can be especially high if this tool is applied to a camera that includes a hardware area of interest (AOI) capability that is supported by PreciseVision.

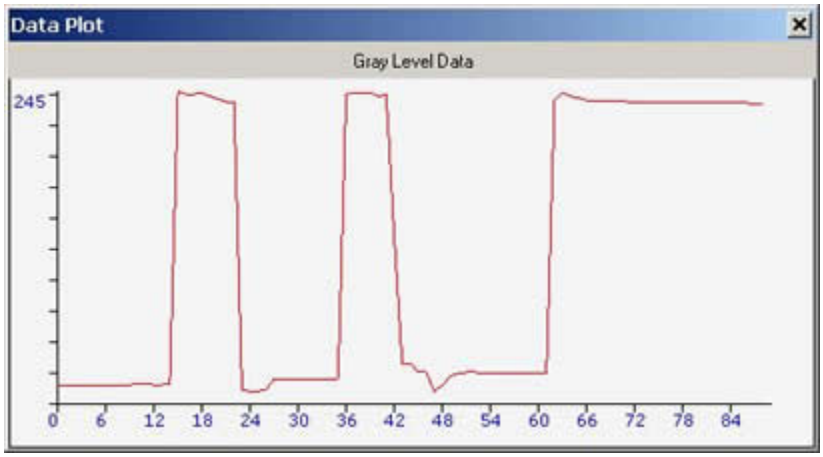
To permit a detailed analysis of the results of this tool, two integer values are returned for each image:

- The mean, maximum or minimum gray-scale value of all pixels within the tool's region of interest (0-255).
- The number of msec that elapsed from the start of the image capture until the start of the next image capture. This time will differ from the target time due to variability in the camera acquisition and delays due to the PC operating system. If pictures cannot be acquired at the specified rate, they will be captured as quickly as possible. The time for the first acquisition will be somewhat longer than subsequent times due to acquisition setup operations.

Special Feature Buttons (located above the property editor)

Execute and Graph

Pressing this button executes the tool a single time and displays the collected mean, maximum or minimum gray-scale value for each of the captured images. This provides a quick graphical presentation of the data collected by this tool.



Sample GPL Program

The following is a sample GPL program that executes a vision process ("main") that includes a **Light Recorder** tool. The output of this program for a sample image are shown following the program.

```
Public Sub MAIN

    Dim vis As New Vision
    Dim visRes As New VisResult

    vis.IPAddress = "192.168.0.112"

    vis.Process("main")
```

```
    If vis.ErrorCode = 0 Then
        visRes = vis.Result("light_recorder",1)
        Dim ii As Integer
        For ii = 0 To visRes.InfoCount-1 Step 2
            Console.WriteLine("Gray:" & CStr(visRes.Info(ii)) & _
                              " mSec:" & CStr(visRes.Info(ii+1)))
        Next
    End If
End Sub
```

Output:

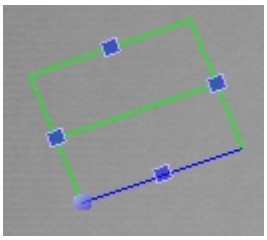
```
Gray:20 mSec:193
Gray:17 mSec:187
Gray:28 mSec:187
Gray:34 mSec:191
Gray:38 mSec:187
Gray:46 mSec:186
Gray:57 mSec:189
Gray:65 mSec:186
Gray:73 mSec:187
Gray:79 mSec:187
Gray:49 mSec:189
Gray:29 mSec:189
Gray:26 mSec:187
Gray:32 mSec:187
Gray:84 mSec:192
Gray:241 mSec:186
```

See Also

Vision Toolkit | Acquisition Tool | Pixel Window Tool

Line Fitter Tool

Vision tool that searches a specified region for edge points and returns the line that best fits the edges.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the tool and the center of the search region. The Height and Width specify the size of the search region.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
EdgeMode	**List**	..._NEAREST_DARK / ..._NEAREST_NOMINAL / ..._NEAREST_LIGHT / ..._MAX_GRADIENTS / ..._MAX_CONCENTRATION	If multiple lines exist within the search region, this parameter helps to distinguish which line should be returned. In addition to selecting the line closest to the dark side, light side, nominal (center line), or greatest concentration of edges, you can also select the

			line with the highest contrast. The concentration mode is particularly useful if there are two or more distinct non-colinear groups of edges and the returned fit line is attempting to span multiple groups. If concentration mode is specified, this tool will select the largest cohesive group of edges as the basis for the line fitting.
EdgeThreshold	**Integer**	0 - 254	Specifies the threshold below which weak (low contrast) gradients edges are ignored. The smaller the number, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
3A. Advanced Operation			
AllowedAngular-Range	**Single**	0.01 - 89 (deg)	(MAX_CONCENTRATION mode only) This limits the orientation of the line that will be returned. It is specified in degrees and is a plus or minus rotation relative to the nominal angle of the tool. In special cases when there are multiple uneven edges, this parameter can be utilized in combination with the MAX_ CONCENTRATION mode to locate the dominant line with a selected orientation.
BiasFilter	**List**	..._BIAS_NONE / ..._BIAS_TO_LIGHT / ..._BIAS_TO_DARK	For lines with an irregular or jagged edge, this parameter can be set to provide extra weight (importance) to edges that are closest to the light or dark side of the search region. For example, for a saw toothed pattern, this property could be set to fit the

			line to the tips of the high or low points rather than along the center line.
DrawEdgePoints	**List**	..._NO_EDGE_POINTS / ..._ALL_EDGE_POINTS / ..._USED_EDGE_POINTS	If set, draws yellow dots to indicate ALL edge points that are selected after the **EdgeMode** is applied or only those edges that are USED to fit lines after the **EdgeMode** is applied and all outliers are eliminated. This is very helpful for understanding the effect of changing various property settings. This value is NO_EDGE_POINTS by default.
MaxEdgePoints	**Integer**	2 - n	Maximum number of edge point searches to perform. Higher numbers produce greater accuracy at the expense of execution speed. The minimum value of this parameter is 2. This value is automatically limited to the number of pixels across the width of the tool. The default value is 30.
MaxIterations	**Integer**	0 - n	Maximum number of iterations for filtering. The filtering algorithm repeatedly removes outlier points, refitting the line each time, until no more points need to be removed or the maximum number of iterations have occurred. A value of zero disables filtering. The default value is 5.
MinFilterDistance	**Single**	0 - n.nn	Absolute minimum filter distance in pixels. No edge points closer to the fitted line than this are discarded during the iterative process. A minimum distance is needed because the standard deviation of distances to the fitted line can be less than a pixel with a good image of a

			clean edge. The default value is 1.
MinPercentEdges	**Integer**	1 - 100	(MAX_CONCENTRATION mode with **BiasFilter** only) Minimum width (as a percentage of the tool's **Width**) that a candidate edge must span in order to be an acceptable result. For special cases, this can be a very powerful feature to filter out short undesirable lines.
SigmaFilter	**Single**	0 - n.nn	Filter width in units of standard deviations. This value is multiplied by the standard deviation of the edge points' distances to the fitted line to compute the distance threshold beyond which edge points are removed during the iterative fitting process. The default value is 1.5.
4. Inspection Settings			
InspectType	**List**	NONE / ..._RMS / ..._ANGLE / ..._NUM_EDGES_FOUND / ..._NUM_EDGES_USED	Standard Inspection Settings properties
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			

ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultNum-EdgesFound	**Integer**		Number of edge points found. *[GPL: VisResult.Info(1)]*
ResultNum-EdgesUsed	**Integer**		Number of edge points remaining after the filtering process. *[GPL: VisResult.Info(2)]*
ResultRMS	**Single**		Root mean square of the line fit. *[GPL: VisResult.Info(0)]*

Remarks

The Line Fitter searches the region defined by its green and blue outlined rectangle for edge points and fits a line to the edge points using a least squares technique. By computing the edge positions to a sub-pixel accuracy and by employing multiple such edges in its computation, this tool is able to generate very accurate sub-pixel results.

This tool is very useful for accurately locating straight edges of objects. Once found, these fit lines can be used to compute distances between important features or for generating reference frames that can accurately locate an object or features in an object.

In order to make this tool more discriminating, the Line Fitter only utilizes edges whose dark and light sides have roughly the same orientation as the dark (blue) and light (green) sides of the search area. So, when positioning this tool, you should place the center (nominal) line of the search region at approximately the position and orientation that you expect to find the line with the blue side on the anticipated dark side of the object.

The operation of this tool is generally performed in the following three steps.

1. All of the edges in the rectangular region are extracted by performing **MaxEdgePoints** number of linear edge searches using the **Edge Threshold**.
2. For each edge search, the single edge point that best fits the **EdgeMode** criteria (nearest dark, nearest light, etc.) is retained and the other edge points are discarded.
3. A straight line is fit to the selected edge points.

Typically, the Line Fitter will perform the fitting process iteratively to increase the robustness of the results. During each line fitting pass, edge points that are too distant from the line (outliers) are discarded. This filters out edges that are not a part of the line and whose inclusion would incorrectly offset the result. **MaxIterations** specifies the maximum number of fits to perform. After each line fit, edges that are beyond **SigmaFilter** standard deviations are rejected, unless the edge is within **MinFilterDistance** pixels of the fit line. The iterative process stops if (i) **MaxIterations** are performed, (ii) no further edges are discarded, or (iii) only two edge points remain.

Of the various operations, the tool spends most of its time in the first step, extracting all of the edges. So, the speed and the accuracy of the Line Fitter can be traded-off by adjusting **MaxEdgePoints**. This property determines how densely the tool scans to detect edges. This parameter must be set to at least 2 in order to detect the minimum number of edges to define a line. At most, one edge search is performed for each pixel along the width of the Fitter. Execution time increases roughly in proportion to the value of **MaxEdgePoints**. On the other hand, the accuracy of the tools increases approximately as the square root of **MaxEdgePoints**. That is, if you increase **MaxEdgePoints** by a factor of 4, the tool execution time will quadruple but the accuracy will only double.

In order to address some unusual cases, the Line Fitter provides special modes of operation. For lines with an irregular or jagged edge, the **BiasFilter** property can provide extra weight (importance) during the line fitting operation to edges that are closest to the light or dark side of the search region. This has the effect of causing edge points that are furthest from the biased edges to be discarded.

As another special case, if the search area has two or more distinct groups of edges and it is not possible to utilize the **EdgeMode** nearest dark, light or nominal methods to distinguish the desired line, the "MAX_CONCENTRATION" mode can be utilized to locate the most prominent line. During Step 2, this method analyzes all of the edge points and computes a best guess line within the specified range of orientation that has the highest concentration of strong edges. Then, for each edge search, the point that is closest to this best guess line is retained and all other edges points are discarded. Step 3 progresses as usual to compute the best fit line to the selected edge points.

At the conclusion of the Line Fitter, the center position and orientation of the best fit line are returned. In addition, statistical data on the number of edges finally used and the root mean square of the line fit are returned. This statistical data indicates the quality of the final fit and can be tested via the "Inspect Settings" to yield a pass/fail indication for the operation.

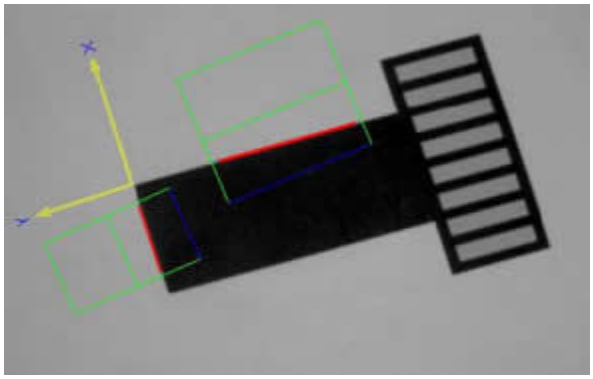
Special Feature Buttons (located above the property editor)

Adjust Threshold

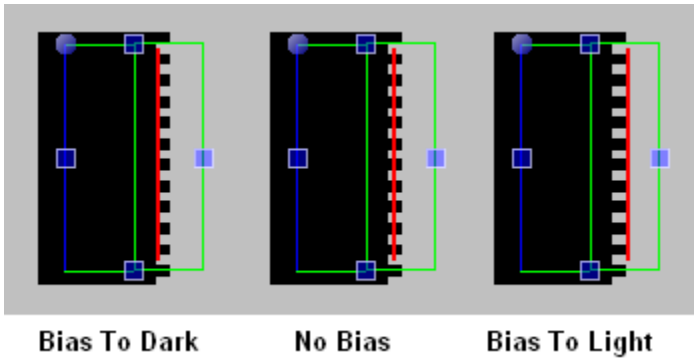
Pressing this button displays a pop-up window that permits the value of the **Edge Threshold** property to be manually adjusted while dynamically viewing the effect of the new setting in the Camera Display. The edges that satisfy the threshold criterion are displayed in white.

Examples

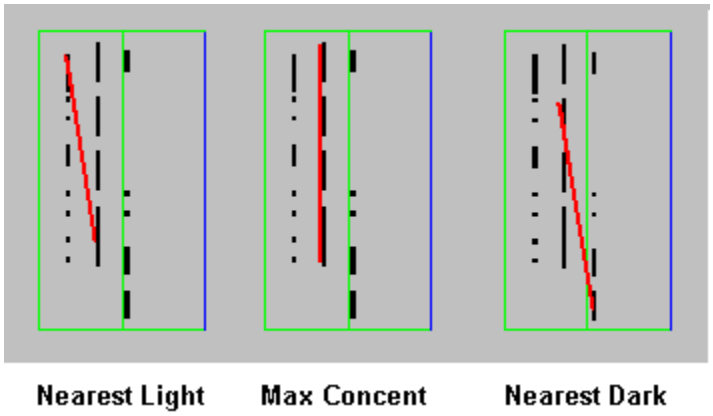
In the following example, two Line Fitters are utilized to accurately determine the positions and orientations of the straight sides of an object. A Computed Line-Line Intersection tool then takes the output of the two Fitters and generates a reference frame whose origin is at the intersection of the two fit lines. Since the two Line Fitters compute their results based upon multiple pixel values, their output values are very accurate.



The following example demonstrates the special case where the outer or inner edge of a series of fingers is to be located. If the standard Line Fitter is applied with FITLINE_BIAS_NONE, the fit line is located somewhere along the center of the fingers. To properly locate the inner or outer edge of the fingers, the **BiasFilter** should be set to FITLINE_BIAS_TO_DARK or FITLINE_BIAS_TO_LIGHT.



The following example demonstrates the special case where there are a series of closely spaced discontinuous parallel edges and you wish to locate the strongest edge. If you are not able to accurately position the finder and use **EdgeMode** to distinguish the desired line by specifying the nearest light, dark or nominal, FITLINE_MAX_CONCENTRATION may be of use. This method locates the line with the greatest concentration of strong edges and uses this to select the edge points that are considered during the iterative line fitting.

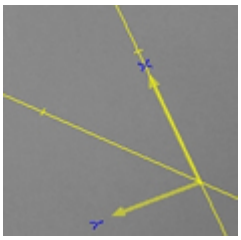


See Also

Vision Toolkit | Arc Fitter | Edge Locator | Find Centerline Tool | Point-Point Line Tool

Line-Line Frame Tool

Computational tool that determines the intersection of two lines defined by vision objects and returns a reference frame. Alternately, this tool can compute and return the midpoint of two reference frames defined by two vision objects.



Prerequisites

Requires two vision objects that each supply either a line or a frame.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
SourceLine1Name	**List**	n/a	Name of the vision object that defines the line that determines the X-axis of the computed reference frame.
SourceLine2Name	**List**	n/a	Name of the vision object that defines the second line that is intersected with the first line.
4. Inspection Settings			
InspectType	**List**	NONE / RESULT_ANGLE	Standard Inspection Settings properties. Can be used to validate the **ResultAngle**.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	

OffsetX	**Single**	mm	Standard Results Settings properties
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
ResultMode	**List**	LINE_TO_LINE_INTERS / MID_POINT_ANGLE_AVER / MID_POINT_ANGLE_SRC1 / MID_POINT_ANGLE_SRC2 / MID_POINT_ANGLE_ZERO	Specifies the result to be computed by this tool. LINE_TO_LINE... - Tool returns a reference frame that represents the intersection of two lines. MID_POINT_ANGLE... - Tool returns a reference frame whose origin is the midpoint between the origin of **SourceLine1Name** and **SourceLine2Name**. The returned orientation can be the average of the orientations of the two sources, the orientation of either source, or zero.
6. Results			
ResultAngle	**Single**	-360 to 360	Standard Results properties
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

This computational tool combines the results of two vision tools to compute a new reference frame.

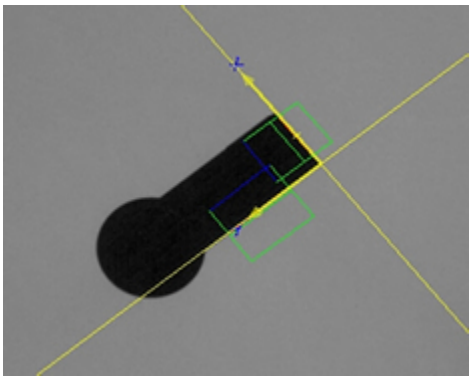
If the **ResultMode** is set to LINE_TO_LINE_INTERSECT, the inputs must be two vision objects that define two intersecting, non-parallel lines. The X-axis of the new reference frame will be collinear with the first input line. The origin of the new frame will be at the point where the two lines intersect.

This method is often used to uniquely define the position and orientation of a simple object or a section of an object. The resulting reference frame can then be used to position additional vision tools. As another example, this method can be utilized to compute the position of a corner of an object as determined by its two adjacent straight sides.

If the **ResultMode** is set to one of the MID_POINT_ANGLE_* methods, the origin of the returned reference frame will represent the midpoint between the origin of the two source vision objects. The orientation of the returned reference frame can be selected as: the average of the orientations of the two source vision objects; the orientation of one of the two source vision objects; or zero.

Examples

The following example illustrates how two Line Fitters can be applied to accurately identify the position and orientation of two adjacent sides of an object. The results of the two Fitters are then fed into a Line-Line Frame Tool with the **ResultMode** set to the LINE_TO_LINE_INTERSECT method to yield a reference frame that accurately represents the position and orientation of the enclosed corner.

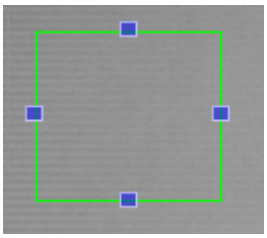


See Also

Vision Toolkit | **Fixed Frame Tool** | **Point-Line Frame Tool** | **Point-Point Line Tool**

Pixel Window Tool

Vision tool that counts pixels of a specific type or computes pixel statistics within a rotated rectangular or circular region. This tool is typically used to quickly determine if a feature is present or to verify that the average gray-level reading of a region is correct.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the center of the region to be processed. The Height and Width define the dimensions of the region.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
PixelMode	**List**	BINARY_STATS / GRAY_STATS / EDGE_LEVEL	When this tool executes, it operates in one of three different modes: "BINARY_STATS" counts white/black pixels after thresholding. "GRAY_STATS" returns gray-level statistics (i.e. mean gray-

			levels). "EDGE_LEVEL" counts edge pixels after a gradient operator is applied. Where a property is only relevant for a specific mode, the property is labeled as BINARY, GRAY or EDGE.
CountWhitePixels	**Boolean**	TRUE / FALSE	(BINARY) If TRUE indicates that the white pixels are to be counted. Otherwise, black pixels are counted.
Threshold	**Integer**	0 - 254	(BINARY) This is the value used to convert gray scale values into black and white pixels. (GRAY) This value is ignored. (EDGE) Specifies the threshold below which weak (low contrast) gradient edges are ignored. The lower this property is set, the more sensitive the system is in locating edges. Please see the description of the Edge Locator for more information on edge gradients. The default value is 30.
ToolShapeType	**List**	RECTANGULAR / ROUND / SQUARE	Shape of the window in which the pixels are analyzed. A square window has its aspect ratio forced to 1:1.
4. Inspection Settings			
InspectType	**List**	NONE / MEAN / NUM_PIXELS / STANDARD_DEV / MINPIXELVAL / MAXPIXELVAL / PIXELSUM / PIXELSUMSQUARE	Standard Inspection Settings properties
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		

InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	0 (non-rotating)	(BINARY,EDGE) X and Y position represent the centroid of the counted binary or edge pixels.
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultPixelCount	**Integer**		(BINARY,EDGE) Total number of counted binary or edge pixels in the rectangular region. *[GPL: VisResult.Info(0)]*
ResultMean	**Single**		(GRAY) Mean gray-scale value of all pixels in the region. *[GPL: VisResult.Info(1)]*
ResultMaxPixelValue	**Integer**		(GRAY) Maximum gray-scale value of all pixels in the region. *[GPL: VisResult.Info(4)]*
ResultMinPixelValue	**Integer**		(GRAY) Minimum gray-scale value of all pixels in the region. *[GPL: VisResult.Info(3)]*
ResultPixelSum	**Integer**		(GRAY) Sum of gray-scale values of all pixels in the region. *[GPL: VisResult.Info(5)]*
ResultPixelSumSqr	**Integer**		(GRAY) Sum of the squares of the gray-scale values for all pixels in the region. *[GPL: VisResult.Info(6)]*
ResultStandardDev	**Single**		(GRAY) Standard Deviation of the gray-scale values for all pixels in the region. *[GPL: VisResult.Info(2)]*

Remarks

This tool very quickly processes a rectangular or circular region of a gray-scale image and either counts black or white pixels after thresholding, counts edge pixels after applying a gradient operator or collects image statistics, such as mean, maximum, minimum and standard deviation of the pixel gray-scale values.

This tool provides a very efficient means for quickly extracting gross statistical information about a rectangular or circular region. This tool is often used to quickly verify the presence of a feature or obstacle or to collect gray-scale statistics. For example, this tool can be employed to inspect a part to ensure that expected holes are present or that tabs have been removed. The BINARY mode can be applied to detecting defects in a uniform area such as inspecting for scratches or pits in a lens. The GRAY mode can be used as a simple light meter to adjust the threshold for other operations or can determine the gray-scale uniformity of the region.

When this tool is operated in the BINARY mode, the specified region is converted to white and black pixels based upon the **Threshold** property. Then, either the white or the black pixels are counted based upon the **CountWhitePixels** property. The number of counted pixels and the centroid of all of the positions of the counted pixels are returned in the results.

If this tool is operated in the EDGE mode, the specified region is converted to edge pixels using a gradient operator and the setting of the **Threshold** property to select only strong edges. Then, the edge pixels are counted. The number of edge pixels and the centroid of all of their positions are returned in the results.

If this tool is operated in GRAY mode, statistics are collected on all of the pixel gray-scale values within the specified region. The returned results include the mean value, maximum value, minimum value, sum of the squares, and standard deviation.

Special Feature Buttons (located above the property editor)

Adjust Threshold

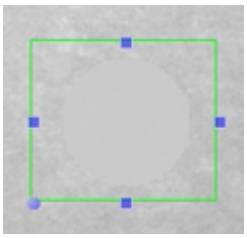
Pressing this button displays a pop-up window that permits the value of the **Threshold** property to be easily adjusted while dynamically viewing the effect of the new setting in the Camera Display. In BINARY mode, the displayed image shows all pixels converted to black or white, where the white pixels are the ones that are counted. In EDGE mode, the edge pixels that are extracted and will be counted are displayed in white.

See Also

Vision Toolkit | Pixel Color Window

Pixel Color Window Tool

Vision tool that tests if the color of a region matches a trained color. The color is considered acceptable if, for the region of interest, the match of the mean intensity of each of the RGB color layers and their trained values is above an acceptable threshold. In addition, this tool computes the mean HSV/HSI values for the region.



Prerequisites

Requires a color camera.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	0 (*non rotating*)	Standard Placement / Size properties. The X and Y values define the center of the region to be processed. The Height and Width define the dimensions of the region.
Height	**Single**	0 - AOI (mm)	
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	0	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation properties
Relative-ToolName	**List**	n/a	
ToleranceBlue	**Integer**	0-100%	Define the minimum match percentages that must be satisfied for this color comparison to return PASS. The color match is performed in the specified region between the mean intensity values for each RGB color layer and the
ToleranceGreen	**Integer**	0-100%	
ToleranceRed	**Integer**	0-100%	

			corresponding trained mean values. For example, if the **ToleranceBlue** is set to 95%, the computed match percentage between the mean of the pixel values in the specified region of the blue color layer and the **InspectNominalBlue** trained values must be 95% or greater for this tool to return PASS. Higher tolerance values require a better color match and lower tolerance values will accept a greater deviation in the color match.
ToolShapeType	**List**	RECTANGULAR / ROUND / SQUARE	Shape of the window in which the pixels are analyzed. A square window has its aspect ratio forced to 1:1.
3A. Advanced Operation			
HueOffset	**Single**	0-360	The computed hue value for the region (**ResultHSVI_Hue**) will range from 0 to 360 degrees. Green will nominally be at 120 degrees, blue at 240 degrees and red can be either 360 or 0 degrees. To simplify testing for red, the **HueOffset** is added to the returned hue value to rotate the color represented by 0 degrees. This can avoid a 0/360 discontinuity.
4. Inspection Settings			
InspectType	List	NONE / RESULT_NORMALIZED / RESULT_FIXED	Standard Inspection Settings properties. In the RESULT_FIXED mode, the color comparison is performed with the raw nominal and actual mean
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL /	

		VALUE_REAL / VALUE_INTEGER	intensity values. This can be used when the lighting is very controlled and the product color does not vary. In the RESULT_NORMALIZED mode, the nominal and actual mean values are adjusted each time the tool is executed to compensate for uniform variations in the color intensity values. This enables the color matching to be robust even when there is some variation in the lighting.
InspectMax	**Single**		
InspectActualBlue	**Integer**	0-255	
InspectActualGreen	**Integer**		
InspectActualRed	**Integer**		
InspectPassed	**List**	Pass / Fail	
InspectLabelDisplay	**List**	RGB_DISPLAY / HSVI_DISPLAY	If **InspectLabel** is set to VALUE_REAL or VALUE_INTEGER, PV will automatically display the computed mean color as either RGB or HSVI values next to the specified color region in the camera image.
InspectNominalBlue	**Integer**	0-255	These properties define the nominal mean values for the three color layers. These values are compared to the actual mean values for the color layers in the region of interest to determine if the color matches the expected value. These values are typically taught by pressing the Train Color button (see below).
InspectNominalGreen	**Integer**		
InspectNominalRed	**Integer**		
5. Results Settings			
OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	0	

ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultHSVI_Hue	**Single**	0-360	Mean Hue (color) of the pixels in the specified region. Ranges from 0 to 360 degrees as you go around the color wheel (red=0, green=120, blue=240). *[GPL: VisResult.Info(6)]*
ResultHSVI_Saturation	**Single**	0-100	Mean Saturation (intensity of the Hue) of the pixels in the specified region. 0% indicates a gray scale value. 100% indicates a saturated color. *[GPL: VisResult.Info(7)]*
ResultHSVI_Value	**Single**	0-255	Maximum lightness of the RGB colors in the specified region. This is the maximum of the three RGB values. *[GPL: VisResult.Info(8)]*
ResultHSVI_Intensity	**Single**	0-255	Mean lightness or darkness of the color in the specified region. This is the average of the mean RGB readings. *[GPL: VisResult.Info(9)]*
ResultMeanBlue	**Single**	0-255	Mean blue color value of pixels in specified region. *[GPL: VisResult.Info(2)]*
ResultMeanGreen	**Single**	0-255	Mean green color value of pixels in specified region. *[GPL: VisResult.Info(1)]*
ResultMeanRed	**Single**	0-255	Mean red color value of pixels in specified region. *[GPL: VisResult.Info(0)]*
ResultPercentBlueMatch	**Integer**	0-100	Match percentage for the blue color layer *[GPL: VisResult.Info(5)]*
ResultPrecentGreenMatch	**Integer**	0-100	Match percentage for the green color layer *[GPL: VisResult.Info(4)]*
ResultPrecentRedMatch	**Integer**	0-100	Match percentage for the red color layer *[GPL: VisResult.Info(3)]*

Remarks

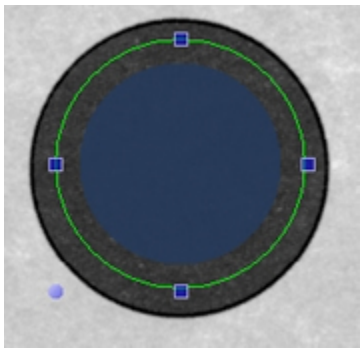
This tool compares the general color of a region of interest with a taught color and returns a Pass/Fail value that indicates if the colors matched as well as or better than specified minimum match criteria. The region can be defined as a rectangular, square or circular shape. The color of the region is determined by computing the mean intensity value for each of the three color layers (RGB). These actual mean values are compared against three taught mean RGB values. If all three pairs of numbers match as well as or better than three specified % criterion, the color of the region is considered to be a match and this tool returns a Pass indication.

To improve the robustness of this tool when the lighting conditions may vary, the **InspectionType** can be set to RESULT_NORMALIZED. In this mode, the taught nominal values and the actual computed mean intensity values are adjusted for the over-all brightness of the region of interest.

This tool always accesses all three RGB color layers of the captured color image independent of which layer the **Acquisition Tool** loaded into the current frame buffer.

To permit further analysis of the computed mean color by other tools, this tool also returns the color characterized by its HSV (hue, saturation, value) and HSI (hue, saturation, intensity) components. These representations map the RGB results into cylindrical color coordinate systems that are a bit more intuitive to interpret. Each circular plane of the cylinder corresponds to a "color wheel". The hue is the angle of the color about the circle and smoothly transitions between red, green and blue. The central vertical axis of the cylinder represents the gray colors ranging from black to white.

To assist in teaching this tool, the center of the tool is always displayed with a semi-transparent circular area that is tinted with the mean color for the region of interest. This is illustrated in the following picture where the center is tinted in blue.



In addition, the camera display window can always be switched into color display mode by pressing on the "Toggle Color Display Mode" button in the main Toolbar.

Special Feature Buttons (located above the property editor)

Train Color

Pressing this button assists in teaching the color for the region of interest. After confirmation, the values for the **InspectNominalRed, InspectNominalGreen, InspectNominalBlue** properties are set to the current mean values of the selected region of the currently displayed image.

Examples

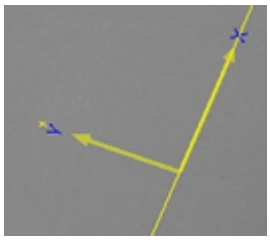
none

See Also

Vision Toolkit | **Pixel Window**

Point-Line Frame Tool

Computational tool that takes a point and a line from two vision objects and returns a reference frame.



Prerequisites

Requires two vision objects, one of which supplies a line or a frame and the other of which supplies a point.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
SourceLineName	**List**	n/a	Name of the vision object that defines the line.
SourcePointName	**List**	n/a	Name of the vision object that defines the point.
4. Inspection Settings			
InspectType	**List**	NONE / RESULT_ANGLE	Standard Inspection Settings properties. Can be used to validate the **ResultAngle**.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			
OffsetAngle	**Single**	degrees	

OffsetX	**Single**	mm	Standard Results Settings properties
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultAngle	**Single**	-360 to 360	Standard Results properties
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

Remarks

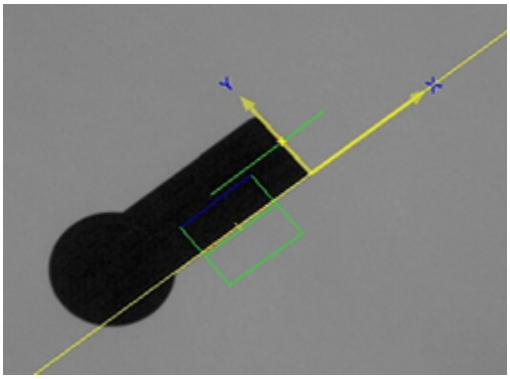
This computational tool combines the results of two vision tools to compute a new reference frame. The inputs to this method from the vision objects are a line and a point that is not on the line. The X-axis of the new reference frame will be collinear with the input line. The Y-axis of the new frame will pass through the specified point.

This method is often used to uniquely define the position and orientation of a simple object or a section of an object. The resulting reference frame can then be used to position additional vision tools.

Alternately, this method can be utilized to compute the distance between a point and a line since the origin of the generated reference frame defines the position on the line that is closest to the input point.

Examples

The following example illustrates how a Line Fitter and an Edge Locator can be applied to accurately identify the position and orientation of one side of an object and a point on a perpendicular side. The results of the Fitter and Locator are then fed into a Point-Line Frame Tool to yield a reference frame that accurately represents the position and orientation of this section of the object. The origin of this reference frame also specifies the point on the line that is closest to the point determined by the Edge Locator.

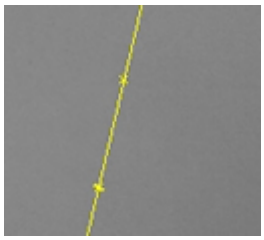


See Also

Vision Toolkit | **Fixed Frame Tool** | **Line-Line Frame Tool** | **Point-Point Line Tool**

Point-Point Line Tool

Computational tool that computes a line given two points from vision objects.



Prerequisites

Requires two vision objects that each supply either a point, a line or a frame.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
SourcePoint1Name	**List**	n/a	Name of the vision object that defines one of the two points on the line.
SourcePoint2Name	**List**	n/a	Name of the vision object that defines one of the two points on the line.
4. Inspection Settings			
InspectType	**List**	NONE / RESULT_ANGLE	Standard Inspection Settings properties. Can be used to validate the **ResultAngle**.
InspectLabel	**List**	NONE / PASS_AND_FAIL / PASS / FAIL / VALUE_REAL / VALUE_INTEGER	
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		
InspectPassed	**List**	Pass / Fail	
5. Results Settings			

OffsetAngle	**Single**	degrees	Standard Results Settings properties
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultOnNotFound	**Boolean**	TRUE / FALSE	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultAngle	**Single**	-360 to 360	Standard Results properties
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

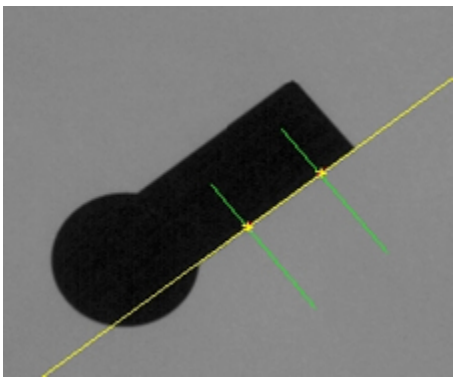
Remarks

This computational tool combines the results of two vision tools to compute a line. The inputs to this method from the vision objects are two points. The resulting line will intercept each of the two input lines.

Once computed, the resulting line can be combined with other vision results to establish reference frames or as a datum for measurements.

Examples

The following example illustrates how two Edge Locators can be applied to find two reference points on the edge of an object to sub-pixel accuracy. The results of the two Locators can then be fed into the Point-Point Line Tool to compute a line that represents the location of one side of the object. This line can then be combined with other tools to measure the dimensions of the object or to establish reference frames. Note that, in general, a more accurate line fit can often be obtained using the Line Fitter Tool. However, there are incidences where an edge is non-uniform and selecting two good points to construct a line is advantageous.

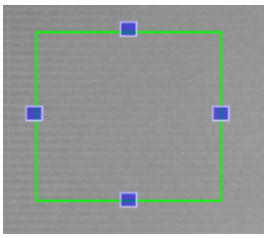


See Also

Vision Toolkit | Fixed Frame Tool | Line-Line Frame Tool | Point-Line Frame Tool | Line Fitter Tool

Sensor Window Tool

Vision tool that detects part movement or no movement within a specified region of the image. This tool takes multiple camera images in succession using a single camera and determines if there are any significant changes.



Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Height	**Single**	0 - AOI (mm)	Standard Placement / Size properties. The X and Y values define the center of the tool. The Height and Width specify the size of the image region being tested.
Width	**Single**	0 - AOI (mm)	
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
DelayAfterComplete	**Single**	0 - n	Specifies an amount of time to wait after the operation is completed (post-processing dwell time).
EdgeThreshold	**Integer**	0 - 254	Specifies the threshold below which weak (low contrast) gradients edges are ignored. The lower this property is set, the more sensitive the system is in locating edges. Please see the description in the Edge Locator Tool for more information on edge gradients. The default value is 30.
MaximumTimeout	**Single**	0-n (seconds)	Specifies a maximum time (in seconds) that the tool will execute.

			If this time is exceeded, the tool stops processing and returns an error.
PixelChangeAmount	**Integer**	0-n	If more than this specified number of edge pixels change between successive images, this tool declares that motion has been detected.
SenseMode	**List**	WAIT_MOTION_STOP / WAIT_MOTION_START	Specifies the operating mode of this tool. The default is to "wait for no motion".
3A. Advanced Operation			
ExposureTime	**Integer**		Same properties as in the Acquisition Tool. See that tool for details.
VideoGain	**Integer**		
VideoOffset	**Integer**		
5. Results Settings			
ShowResults	**List**	NONE / FRAME / LINE / POINT	Standard Results Settings property
6. Results			
ResultErrorCode	**Integer**		Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	
ResultActualTime	**Single**	sec	Actual time in seconds for operation to complete. *[GPL: VisResult.Info(1)]*
ResultPixelCount	**Integer**		Actual number of pixels detected during the operation. *[GPL: VisResult.Info(0)]*

Remarks

This motion sensing tool acquires a series of pictures sequentially and compares successive images to detect how many edges are changing. Based upon the number of changed edges, this tool decides upon whether any movement has been detected.

This tool determines movement by performing the following operations. First, it acquires a new image. Then the new image is subtracted from the previous image. A cross gradient edge extraction algorithm is performed across the resulting image and the edges are thresholded to produce a binary image. The binary image is eroded to eliminate insignificant changes. The resulting binary edges are counted and compared to the **PixelChangeAmount** to determine is sufficient pixels have changed to declare that movement has been detected. Based upon the setting of the **SenseMode** property, this tool either takes another picture and repeats the process or it terminates.

This tool requires successive images so a minimum of two pictures are always taken. During this operation, no graphics are displayed on the screen. A dialog is placed above the vision window to indicate that the operation is in progress and permits the operator to abort the tool.

This tool can be applied in applications where parts may be falling or moving (e.g. rolling parts). Instead of applying a worst case fixed delay time, the Sensor Window Tool can be used to ensure the parts are not moving before locating or executing the remaining processes.

Special Feature Buttons (located above the property list):

Show Motion Image

Displays the output of comparing a single pair of images. Any pixels that are detected as changed (moving) are drawn in white. If zero motion is taking place, the entire image will be black.

Test Motion (Continuous)

Places tool in a continuous execution mode that allows the user to see the results of the motion dynamically. A small dialog box will appear in the center of the vision window that displays the number of pixel counts sensed as being in motion. Click the 'abort' button on the dialog to stop the test operation.

See Also

Vision Toolkit | Acquisition Tool | Edge Locator Tool

Text Label Tool

Text labeling tool that displays string constants plus inspection and results data from a referenced tool. Text is shown in the Camera Display Window relative to the position of the referenced tool.

Prerequisites

None

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
2. Placement/Size			
Angle	**Single**	-360 to 360	Standard Placement / Size properties. The X and Y values define the starting point for the text output and the Angle defines the orientation of the text string.
XPos	**Single**	0 - AOI (mm)	
YPos	**Single**	0 - AOI (mm)	
RelAngle	**Single**	-360 to 360	
RelXPos	**Single**	0 - AOI (mm)	
RelYPos	**Single**	0 - AOI (mm)	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
SourceToolName	**List**	n/a	Name of the vision object whose results are displayed by this tool.
EnableResultName	**Boolean**	True / False	If True, the name of the results data specified by **ResultToolInfo** will automatically be included in the output text string.
ResultToolInfo	**List**	NONE / InspectPassed / InspectFailed / ResultXPos / ResultYPos / ResultAngle / ...	This is a list of Inspection and Results properties that can be displayed. The items in the list are a function of the type of the tool selected by **SourceToolName**. For most tools, at least the properties listed in the Range field are available.
TextColor	**Window Color Selector**	n/a	Permits the font color to be selected from the system color palette.

TextForceZeroAngle	**Boolean**	True / False	If True, will force the output text string to have a zero **Angle** even if the text is relative to a source tool whose result is rotated.
TextOutputType	**List**	FORMAT_INTEGER / FORMAT_REAL	Defines if numerical values should be output as integer or real numbers.
TextSize	**Integer**	1 - 50	Specifies the font size of the output string as a Windows Font Size (not in pixels).
TextTitle	**String**	n/a	Optional string that is placed at the start of the output text. This provides a way to customize the label for the results data. For multiple lines use the delimiter pipe character '|'. When this character is encountered, subsequent items will automatically be placed on the next line, including the results.
5. Results Settings			
ShowResults	**List**	NONE / FRAME / LINE / POINT	Standard Results Settings properties
6. Results			
ResultAngle	**Single**	-360 to 360	Standard Results properties
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

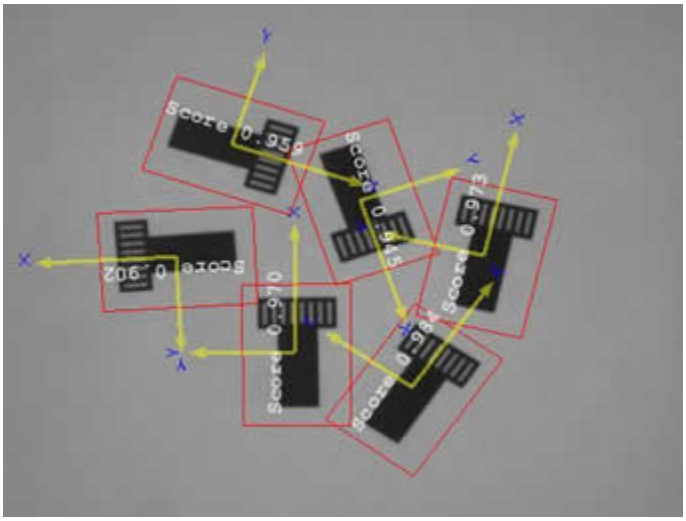
Remarks

This tool displays text labels either in fixed positions in the Camera Display Window or relative to the results of other tools. If the text tool is relative to a source tool, the text can automatically include the values of selected Inspection and Results properties of the source tool.

NOTE: This tool cannot be specified as one of the source tools in an **Inspect List Tool**.

Examples

Below is an example of the **Text Label Tool**. In this example, each of the objects that has been located by a **Finder Tool** is labeled with the Finder's recognition score.



See Also

- **Vision Toolkit**

Tool Filter Tool

Filter tool that takes the output from another vision object that generates multiple sets of results, and returns a subset of the results based upon specified criteria.

Prerequisites

Requires another vision object that generates multiple sets of results.

Properties

Property Name	Data Type	Range	Description
1. Identity			
Name	**String**	n/a	Standard Identity properties
Type	**String**	n/a	
3. Operation			
Camera	**Integer**	1 - 6	Standard Operation property
SourceToolName	**List**	n/a	Name of the vision object that outputs multiple sets of results. These sets of results serve as the input to this tool.
3A. Advanced Operation			
SortResults	**List**	NONE / SORTFROMLEFT / SORTFROMRIGHT / SORTFROMTOP / SORTFROMBOTTOM	Automatically sorts the results based on the selection. This is useful for items on a conveyor belt that may need to be picked in order from the front to back of the conveyor.
ZeroAngle	**Boolean**	True / False	If True, will force the **ResultAngle** to ZERO. This is useful for randomly oriented parts that require the angle to be removed for picking. This operation does NOT change the sorting operation.
4. Inspection Settings			
InspectType	**List**	NONE / RESULT_ID / RESULT_X_POS / RESULT_Y_POS / RESULT_ANGLE / RESULT_INSPECT_-STATUS	Standard Inspection Settings properties. All of the sets of results generated by the **SourceToolName** object are tested against this inspection criteria and those that pass become the output of the Tool Filter.
InspectMax	**Single**		
InspectMin	**Single**		
InspectActual	**Single**		

InspectPassed	**List**	Pass / Fail	
InspectString	**String**	n/a	Optional label that is displayed in the vision window on all input results that satisfy the inspection criteria. This provides a convenient way to display part ID or other information.
5. Results Settings			
MaxResults	**Integer**	-1 or 1 to n	Standard Results Settings properties
OffsetAngle	**Single**	degrees	
OffsetX	**Single**	mm	
OffsetY	**Single**	mm	
ResultSelect	**Integer**	1 - n	
ShowResults	**List**	NONE / FRAME / LINE / POINT	
6. Results			
ResultCount	**Integer**	0 - n	Standard Results properties
ResultAngle	**Single**	-360 to 360	
ResultXPos	**Single**	mm	
ResultYPos	**Single**	mm	

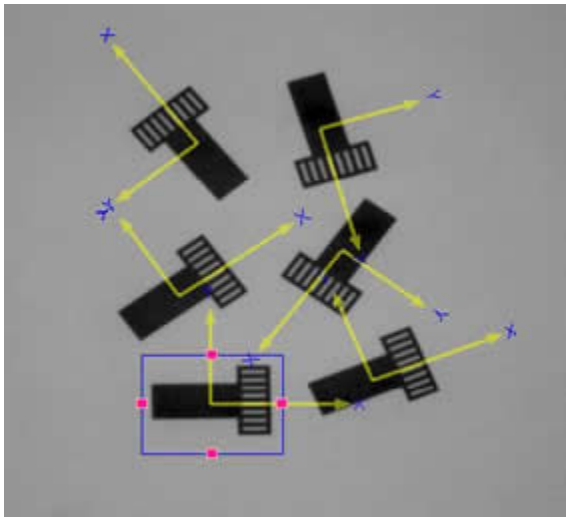
Remarks

This tool inputs the results from another vision object that generated multiple sets of results, and applies criteria specified by the "Inspection Settings" to filter the results. This allows a process to isolate results that require further special processing or to create queues of results based upon specified criteria. Multiple **Tool Filters** can be applied to the same vision object to allow all of the results of the input tool to be segmented and processed.

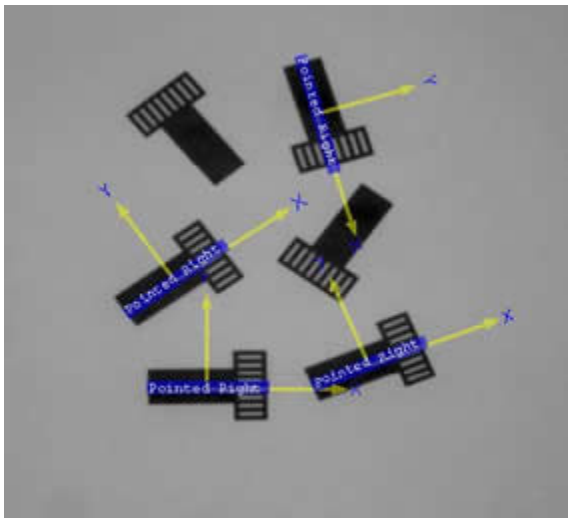
For example, if the input tool is a **Finder**, special processing can be applied to all objects located by the **Finder** that are in a specific region of the screen by defining a **Tool Filter** with the **InspectType** set to "RESULT_X_POS" or "RESULT_Y_POS".

Examples

Below is an example of a single **Finder** that has located 6 identical parts. You will note that the bottom left part was used as the tool template, so that all future part orientations will be with respect to this horizontal position with the top of the "T" pointed to the right.



If we now wish to segment these 6 parts by their orientation, we can apply a **Tool Filter** that references the output of the **Finder** by specifying the **Finder**'s name as the **Filter's SourceToolName**. By setting the **Tool Filter's InspectType** to "RESULT_ANGLE", we can select only those parts with a specific orientation. If we set the **InspectMin** and **InspectMax** to -90 degrees and 90 degrees respectively, only those objects that are approximately pointed to the right will be selected as the output of the Filter. We can also add a label to these parts by filling in the **InspectString**. The picture below shows the results of this filter where **InspectString** has been set to "Pointed Right". You will note that only four of the six parts located by the Finder will be output by the Tool Filter. If we wanted to further process these four parts, the Tool Filter output could be utilized as the input to yet another tool.



See Also

Vision Toolkit | Finder Tool

Camera Calibration

Camera Calibration Introduction

The PreciseVision camera calibration procedures generate the data necessary to convert camera pixel positions into real world units of measure, typically millimeters. For robot guidance, this calibration also translates camera positions into the robot's world coordinate system. This permits a robot to grasp a part that has been located by the vision system. Consequently, a camera must be calibrated with respect to a robot before a vision-guided application can be executed. Even for simple vision testing where PreciseVision is not communicating with a robot, calibration is still highly desirable (but not required), since the calibration process corrects for non-square pixels and cameras that are not perfectly perpendicular to the imaged plane, which causes perspective distortion. This correction is important as parts rotate to ensure that dimensions are always computed the same independent of the part's orientation.

So, when a camera is first put into use, it should be calibrated. In addition, calibration data should be updated if a camera with different optical characteristics is utilized, a camera is moved, a different camera lens is used or the length to the focal plane is altered. In general, it is always best to calibrate each camera that is placed into service and whenever any of the optical characteristics of a camera are modified.

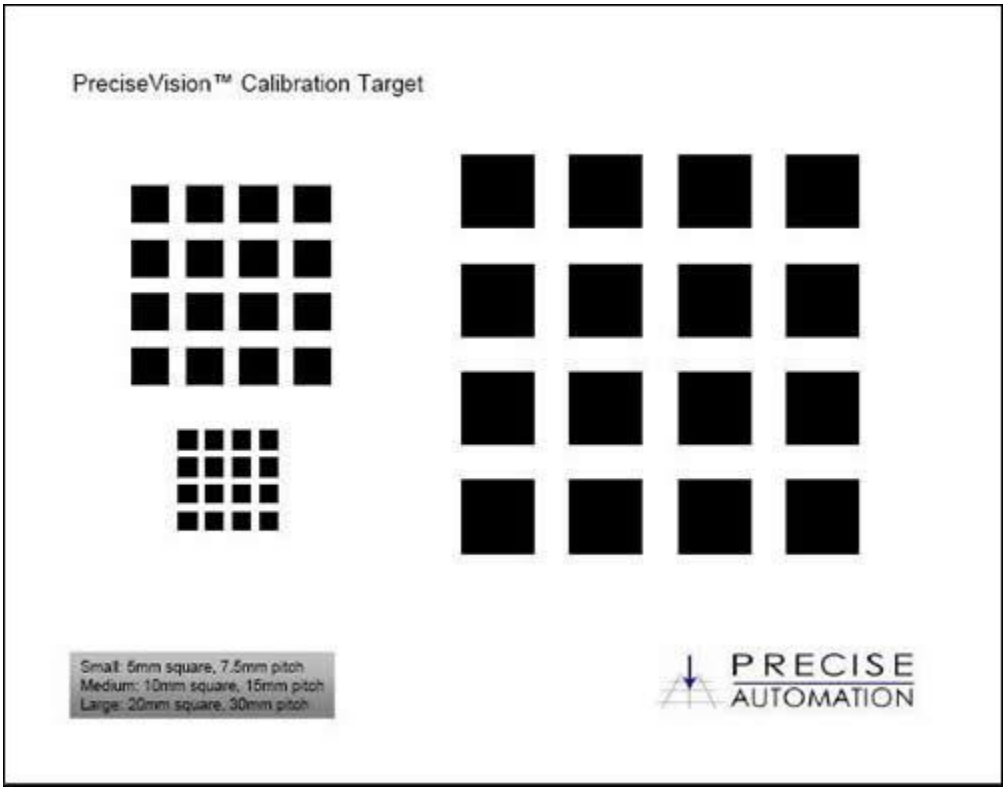
In the next section, general instructions and information that apply to all calibration methods are presented. This is followed by information concerning the various calibration methods.

General Calibration Instructions

This section includes descriptions of general procedures that apply to all calibration methods. This section should be read prior to executing any of the calibration methods.

Standard Calibration Target Sheet

All calibration methods are required to establish the conversion between camera pixel positions and millimeters. In most cases, PreciseVision determines this relationship based on viewing a Standard Calibration Target Sheet that contains patterns of squares that have a known size and pitch. Three different pattern sizes are provided to accommodate different camera fields of view. The target sheet can be used without a robot or any automatic motion. This sheet looks like the following:



This target sheet is provided as a .pdf file and can be found via the Windows Start menu:

Start > Programs > Precise Automation > PV xxx > Documentation > PreciseVisionCalibrationTarget.pdf

or by browsing to the “\resource\” folder created upon installation of PreciseVision.

\Program Files\Precise Automation\PreciseVision #.#\resource\PreciseVisionCalibrationTarget.pdf

The calibration target sheet should be printed out at the highest possible resolution and quality. When you are instructed to do so, this sheet should be placed in the focal plane of the camera. The focal plane of the camera should be as close as possible to the actual Z height of the primary part features to be located or inspected.

Camera Setup

Prior to executing a calibration process, the camera must be rigidly mounted, selected in software and carefully setup. The camera can be selected from the ***View > Select Camera*** top-level menu. The aperture and focus can then be manually set by viewing the camera image in "live video" mode. If the camera is not in this mode, it can be initiated by pressing an icon in the main toolbar. Finally, the camera

gain and offset that control the brightness and brightness range should be set by selecting ***Tool > Camera Video Properties*** .

Camera Calibration Wizard

When you select ***Tool > Calibrate Vision > Camera Calibration Wizard***, a wizard is initiated and the following pop-up is displayed.

This wizard takes you step-by-step through the process of calibrating a camera either for standalone vision or for a vision-guided robotic application with the camera mounted in a variety of configurations. The instructions are fairly self-explanatory.

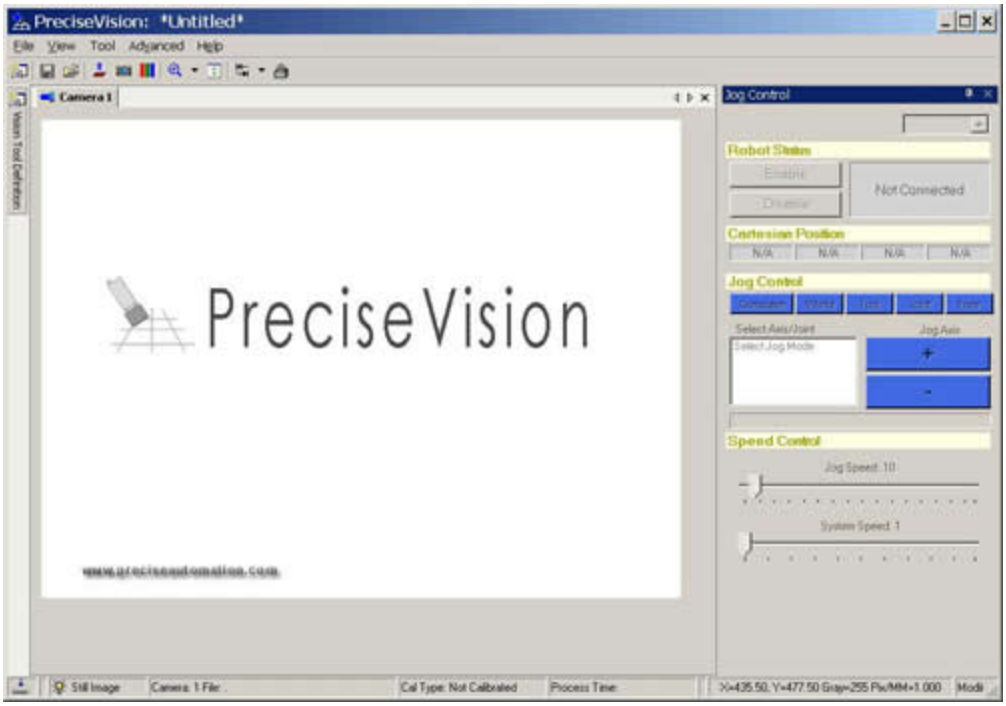
While each page of the wizard is different, the following toolbar buttons are available on every page to assist you with the calibration process.

Calibration Wizard Toolbar		
Icon	**Tool Tip Title**	**Description**
	Live Video	Continuously updates the displayed camera image to show a "live" image.
	Single Picture	Captures a single picture (snap shot) from the selected camera.
	Load Robot Calibration	Displays a pop-up file folder window that permits an existing vision calibration file to be selected and loaded. This allows an old calibration file to be loaded and tested.

	Save Robot Calibration	Displays a pop-up file folder window that permits the current calibration data to be saved to a disk file.
	Continuous Threshold	Updates the displayed camera image as a thresholded binary image instead of the standard gray scale.
	Open Threshold Tool	Displays the pop-up that permits changing the threshold for the binary image.
	Reset Calibration	Provides a way to reset the camera calibration to its default (unity scaling).

Calibration Layout

When the Camera Calibration Wizard is selected, the layout of the main PreciseVision window automatically changes to the following "Calibration Layout".



This arrangement has a large Camera Display Window on the left and a panel on the right that is provided as a convenience for moving a robot in manual control. The robot panel is utilized if a camera-to-robot calibration is performed. The Calibration Wizard includes a panel for connecting to a robot controller. When this connection is established, the manual control panel becomes active.

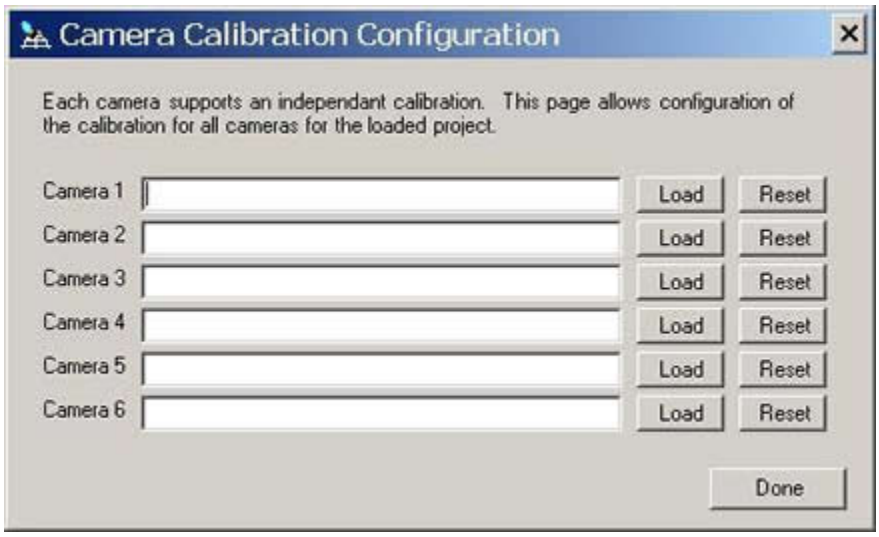
To switch back to the default arrangement for the main PreciseVision window, select ***View > Layouts > Load Default*** or ***View > Layouts > Load Edit Layout***.

Calibration Data Files

Calibration information is stored in disk files with a ".dat" extension. By default, these files are stored in the "ExampleProjects" folder in the PreciseVision directory. For example, for the default installation directory for PreciseVision version 1.2, the calibration files will be stored in the following directory:

C:\Program Files\Precise Automation\PreciseVision 3.0\ExampleProjects

After a calibration data disk file has been created, it has to be specified in each Vision Project in which it is to be used. When a Vision Project is opened, to associate a calibration file with a camera, open the Camera Calibration Configuration pop-up by selecting **Tool > Camera Calibration Configuration**. The pop-up panel should appear as follows:



Click the Load button to select a calibration file to be associated with a camera. A dialog window will be displayed that will allow you to browse to the directory location that contains the calibration (.dat) file. If you wish to delete an association between a camera and a calibration file, click on the Reset button.

Once a camera calibration file is specified in the Camera Calibration Configuration menu, the calibration will automatically be applied whenever a picture is acquired from the referenced camera. Also, whenever the Vision Project is opened, the specified calibration files will be automatically opened and reloaded.

Basic Camera Calibration

All of the calibration methods including standalone vision (Camera Only) and robot guidance begin by calibrating the camera's pixel coordinate system into real world units of measure, typically millimeters. In addition, this procedure adjusts for perspective distortion, which occurs when the camera is not perfectly perpendicular to the plane of the objects to be located. The origin of the camera's coordinate system is set to the bottom left-hand corner of the camera image.

After this basic calibration is completed, the vision system can be used for metrology operations or for other applications not needing a robot. However, since this step does not include a transformation between the camera frame of reference and the world coordinate system of a robot, this step is not sufficient for vision-guided robot applications.

This procedure relies upon the Standard Calibration Target Sheet and does not require a robot or any automated motion.

Prior to executing the Calibration Wizard, the following steps must first be performed:

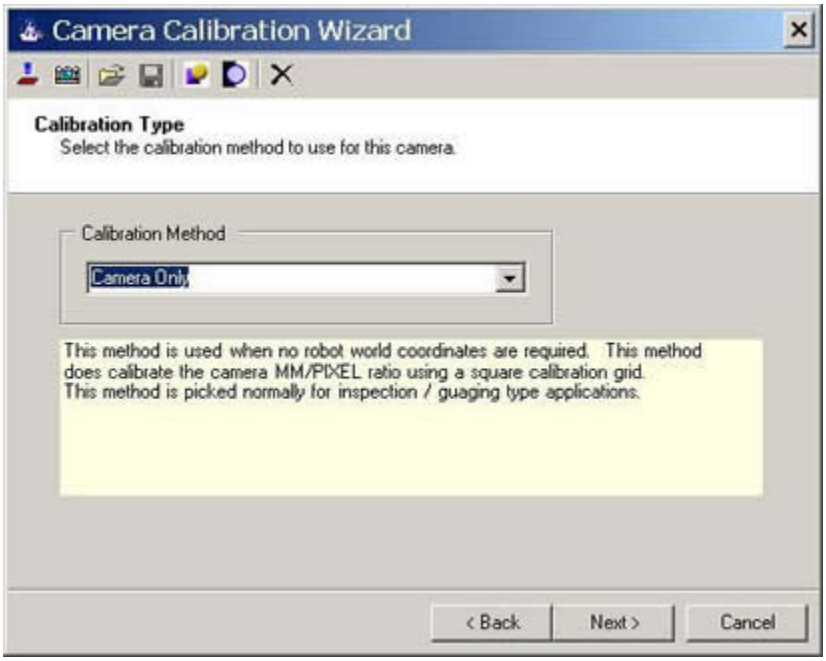
- The "General Calibration Instructions" section of this manual should be read.
- The Standard Calibration Target Sheet must be printed and available.
- The camera must be rigidly mounted, selected, and accurately setup in terms of its focus, aperture setting, and gain and offset.

The Calibration Wizard is fairly self-explanatory. The following provides some additional information for first time users.

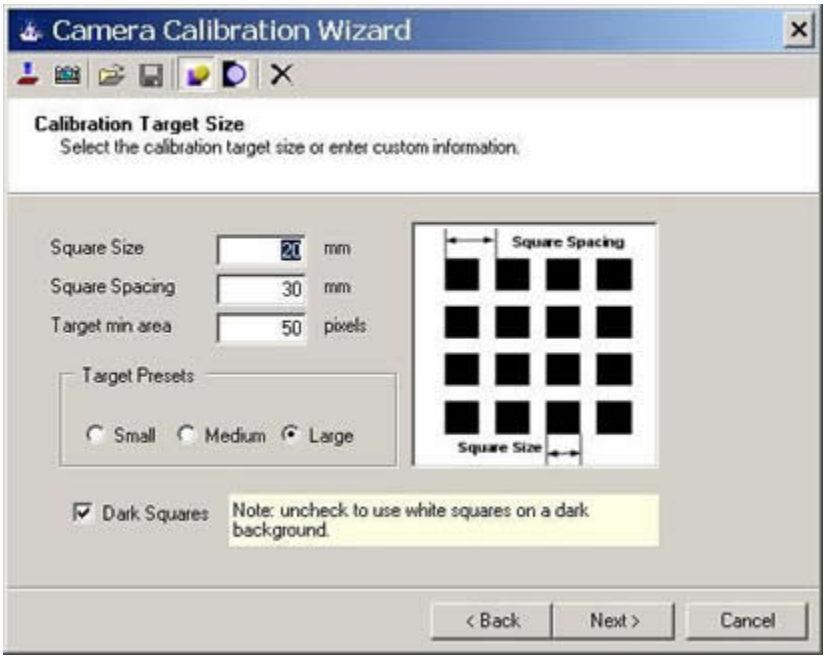
» To start this method, select ***Tool > Calibrate Vision > Camera Calibration Wizard.***

» Advance past the title page by pressing the ***Next*** key. You can stop the calibration process at anytime by pressing the ***Cancel*** key.

» Select the type of camera calibration you wish to perform. ***Camera Only*** will not perform the camera to robot calibration and is only appropriate for testing and standalone vision.



» Advance to the next page and specify the size of the calibration target you will be using.

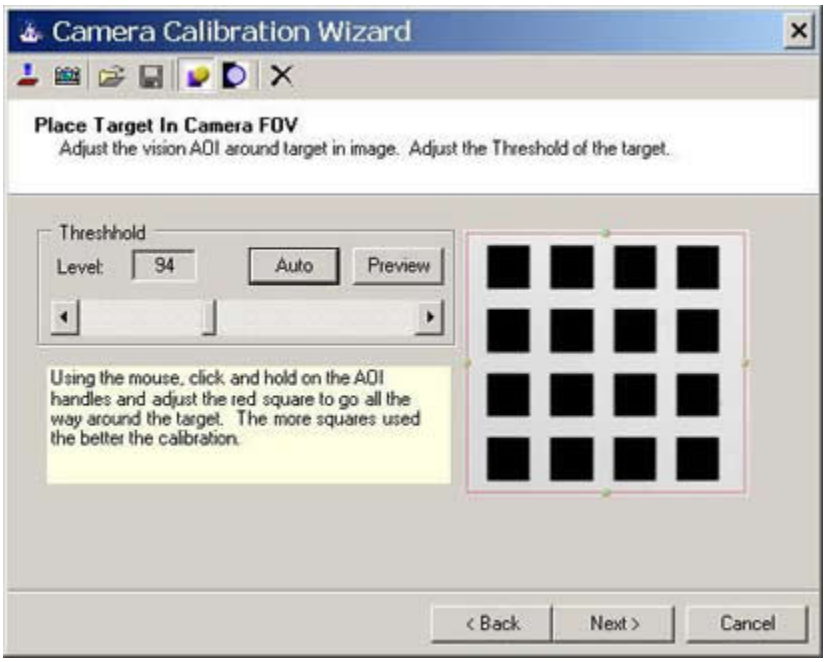


This page permits you to select from one of the standard sizes that are provided on the PreciseVision Calibration Target sheet or to manually enter the target dimension parameters if you have generated a custom sized target that is more appropriate for your application. You should select the largest pattern that is fully visible within the camera's field-of-view.

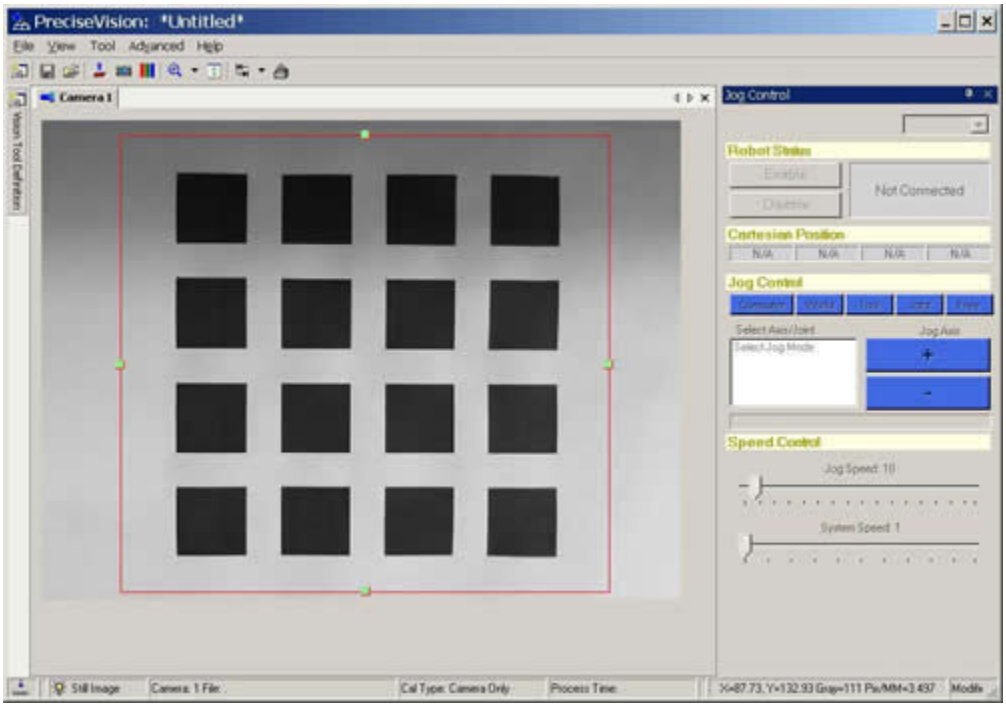
The standard target has black squares printed on a white background, so the ***Dark Squares*** should normally always be checked.

» Place the Calibration Target Sheet in the same focal plane that you expect the primary part features to be located. Position the sheet such that the pattern you have selected is fully visible in the field-of-view.

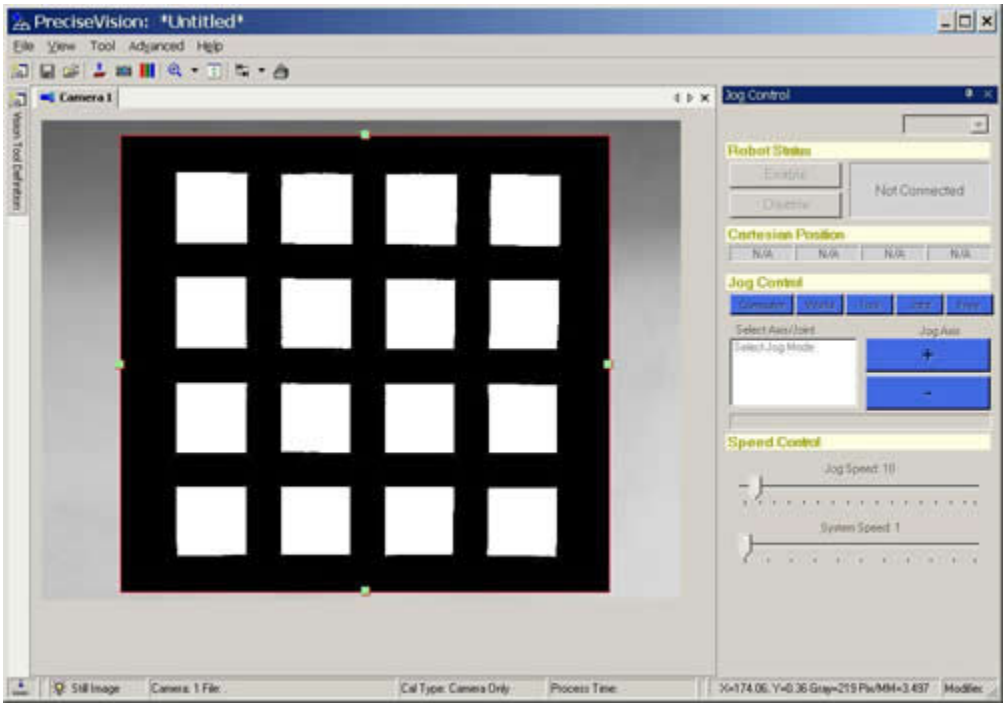
» Advance to the next page of the Wizard to adjust the binary threshold of the image and to set the size and position of the AOI that contains the target.



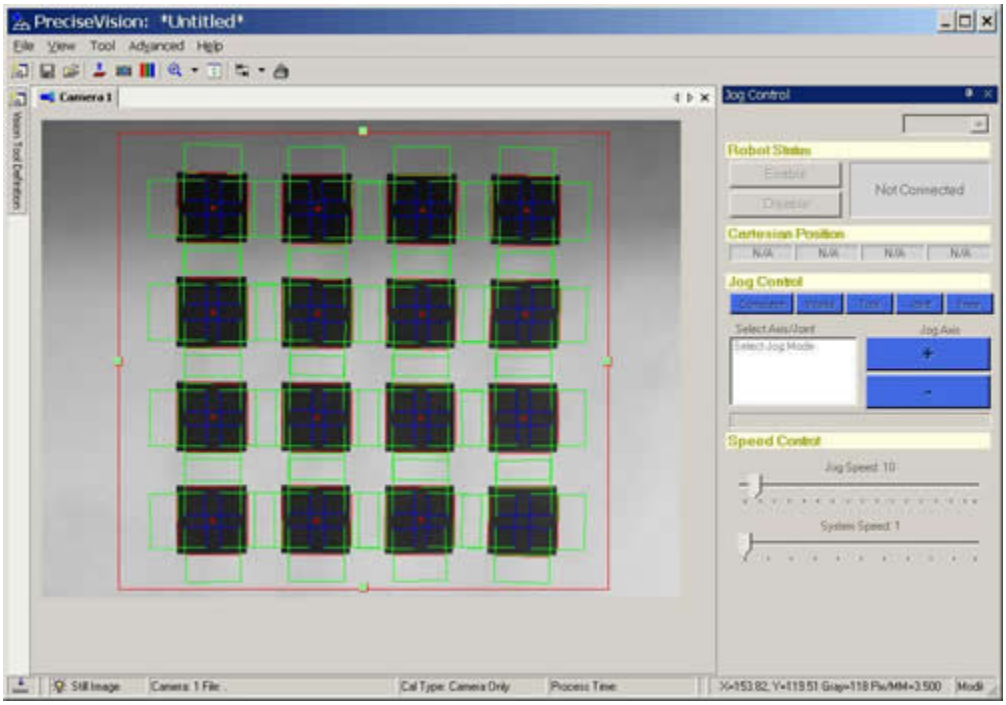
As instructed in the Wizard, you should view the Camera Window and adjust the size and position of the red square outline using the green handles so that the red outline completely surrounds the calibration squares.



When the image is thresholded, it will appear as follows:



» Follow the Wizard's instructions to compute the camera calibration using the target image.



At the completion of the process, the Camera Window will appear as shown above, which indicates how vision tools were used to locate the calibration squares. Each of the four sides of each square should have been automatically and accurately located with a Line Fitter tool. If a message indicates that an error has occurred, most likely, the ***Threshold*** value or the ***Dark Target*** check box were not set correctly.

This completes the Basic Camera Calibration procedure.

Robot Vision Camera Calibration

After the Basic Camera Calibration is completed, if vision is being used with a robot, the transformation from the camera's frame of reference to the robot's world coordinate system must be computed. This information permits PreciseVision to send object X/Y/Z/Theta values to a motion controller in a form that can easily be used to define a pick-up location. PreciseVision includes several different methods for performing this calibration because cameras can be mounted in a number of different positions (e.g. fixed in a stationary position, mounted on a link of the robot, etc.).

Since the instructions for executing the Calibration Wizard are fairly self-explanatory, this section includes just an overview of the methods that are common to all of the procedures.

In general, these methods compute the calibration data by having the operator move a customer supplied calibration disk to several positions in the camera's field of view. At each position, a gripper or pointer attached to the robot must be moved to the same position to establish a correspondence between vision positions and robot positions. In order to execute this procedure, the robot's controller must be powered up and properly setup so that it can be safely operated.

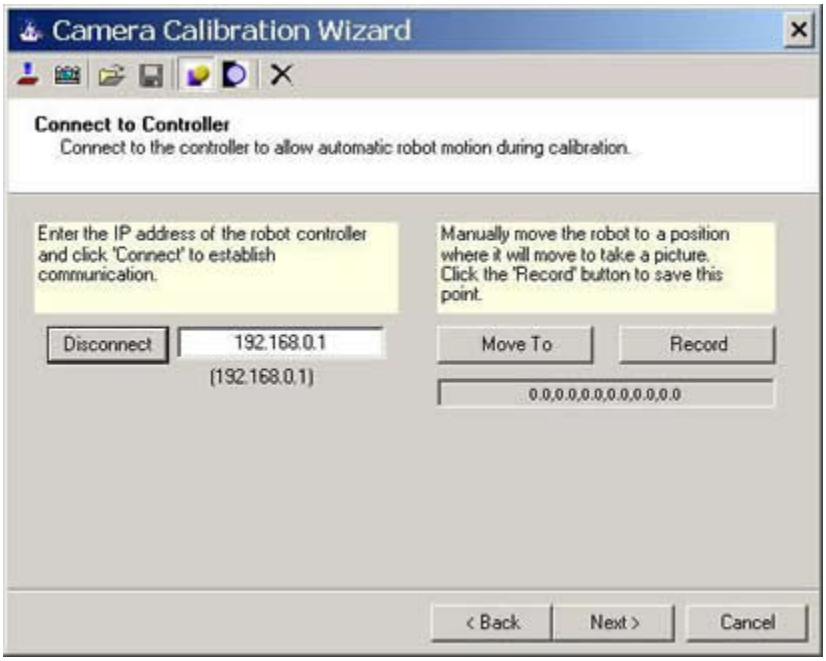
Prior to executing this calibration method, the following steps must first be performed:

- The "General Calibration Instructions" section of this manual should be read.
- The Standard Calibration Target Sheet must be printed and available.
- The camera must be rigidly mounted, selected, and accurately setup in terms of its focus, aperture setting, and gain and offset.
- A circular, relatively flat calibration disk and a pointer or gripper mounted to the tool flange of the robot must be obtained. The calibration disk's diameter is recommended to be roughly 10-15% of the width of the camera's field of view. The calibration methods typically rely upon locating the center of the disk at several different positions. Consequently, if the disk is too large, its size would limit the maximum distance between center points. The color of this disk should be selected to maximize its contrast with the background. The disk should also have a feature that aids the user in aligning the center of the pointer with the center of the disk.
- The Guidance Controller for the robot must be powered on and interfaced via Ethernet to the PC that is executing PreciseVision.
- The robot must be homed and capable of being moved. Some procedures do not require the robot to be moved automatically or under power, while other do.

DANGER: Before proceeding with this procedure, please ensure that the robot has been properly mounted, all required safety interlocks have been installed and tested, and power has been connected. All safe guards required to permit the robot to be manually or automatically repositioned must be in place. For the Precise robots, this information is provided in the robot's *Hardware Introduction and Reference Manual.*

The following general instructions and pictures illustrate the concepts at are common to all of the robot vision calibration procedures.

» When prompted by the Calibration Wizard, specify the IP address of the robot controller and press ***Connect*** to establish an Ethernet connection between the PC and the controller.

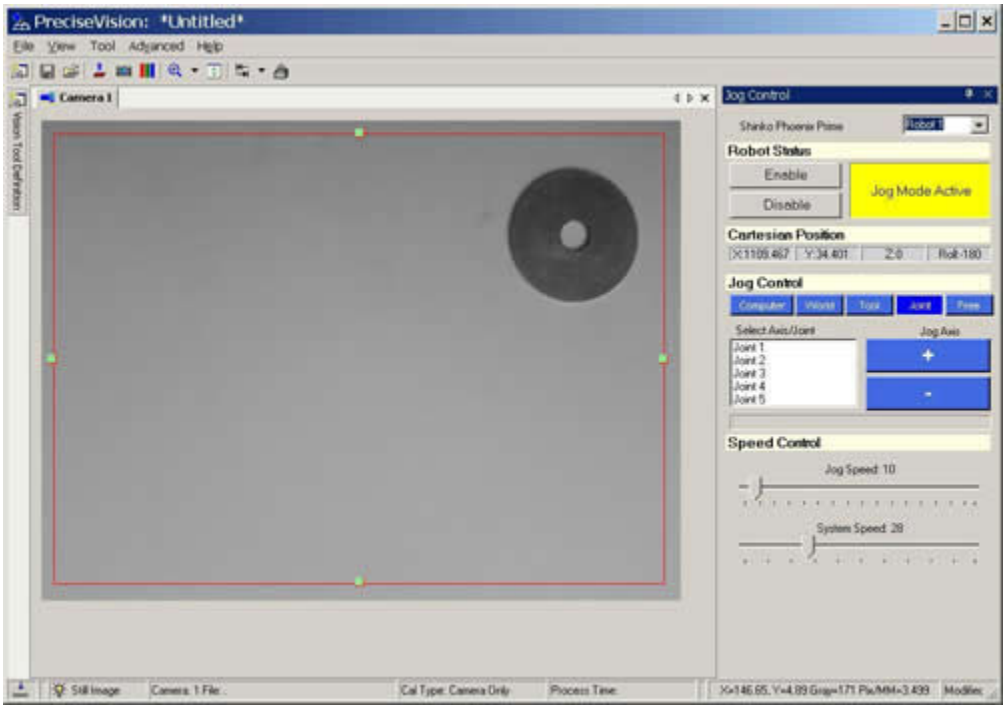


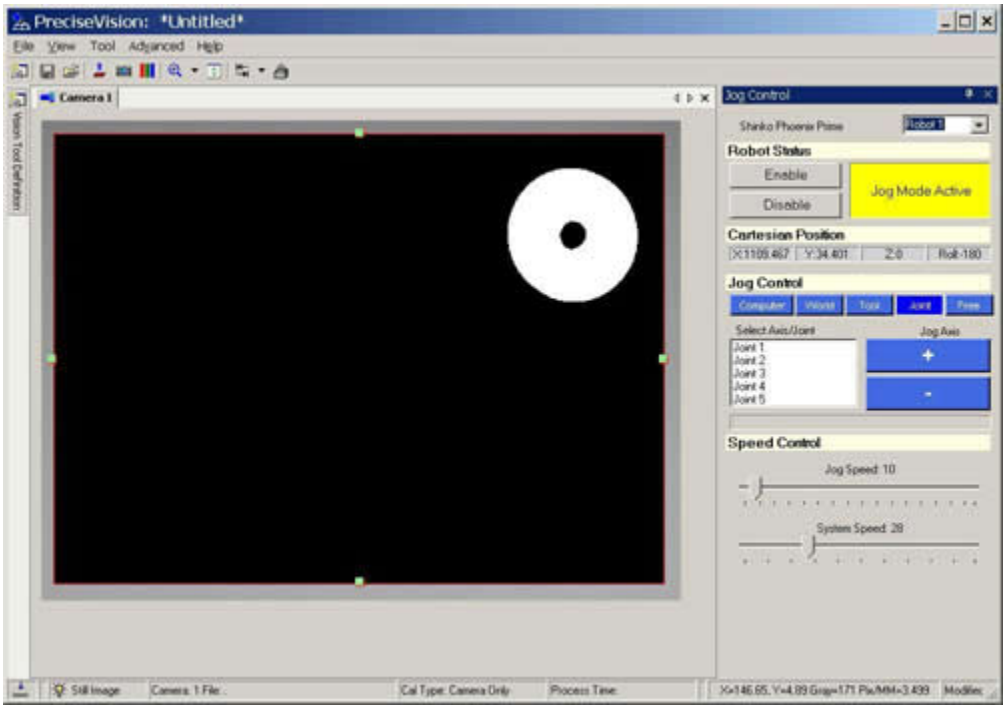
This connection is required in order for PreciseVision to automatically read the position of the robot and to permit manual and automatic control of the robot. After this connection is made, the manual control panel in the main PreciseVision window will become active, although a hardware MCP or the standard web based Virtual MCP can be used as well.

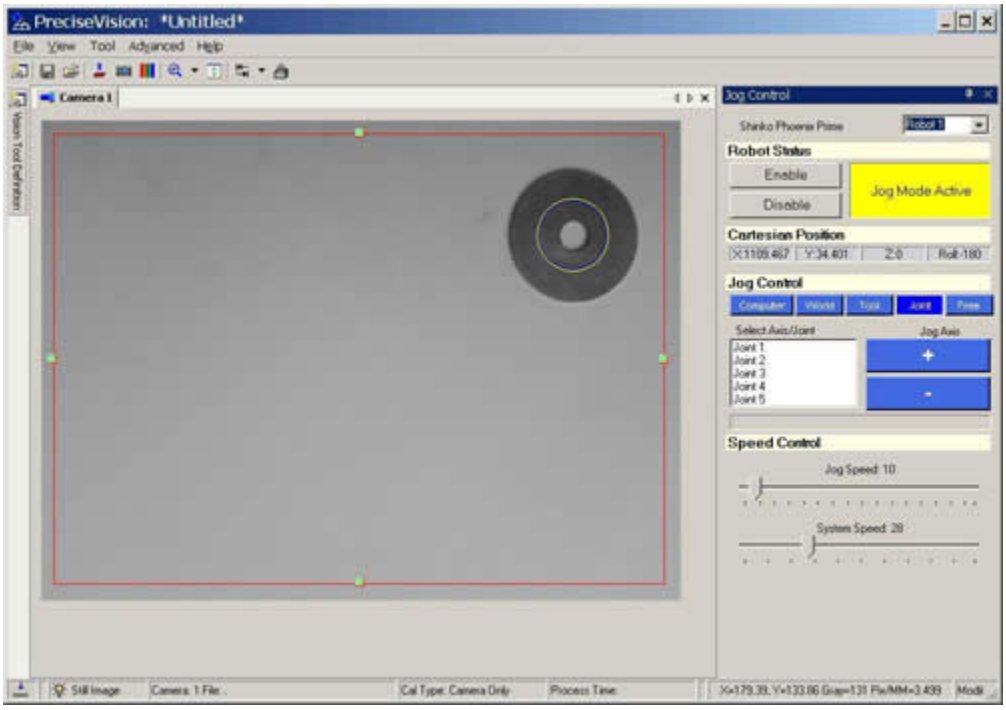
As you execute the Wizard, you will be presented with various options for teaching locations, selecting the teaching method, and moving the robot automatically to return to picture locations or to clear the picture area. These options are dependent upon the specific camera calibration method being performed.

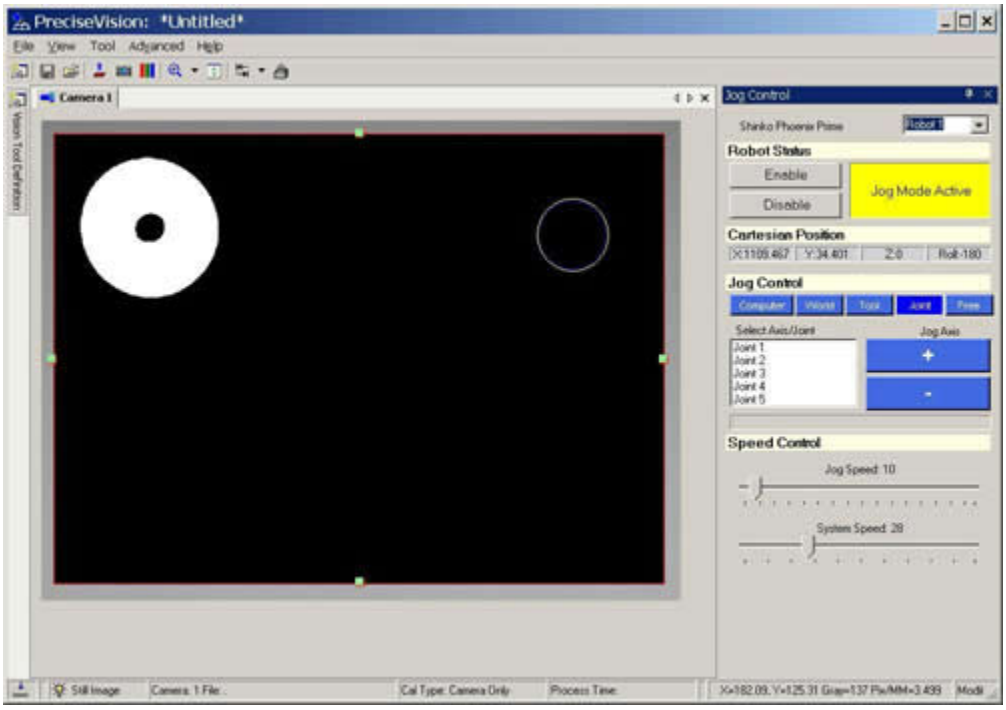
Independent of the specifics of the calibration procedure, fundamentally, all of these methods provide a means for taking multiple pictures, visually locating a calibration disk in different positions within the camera's field of view, and recording the robot's corresponding positions.

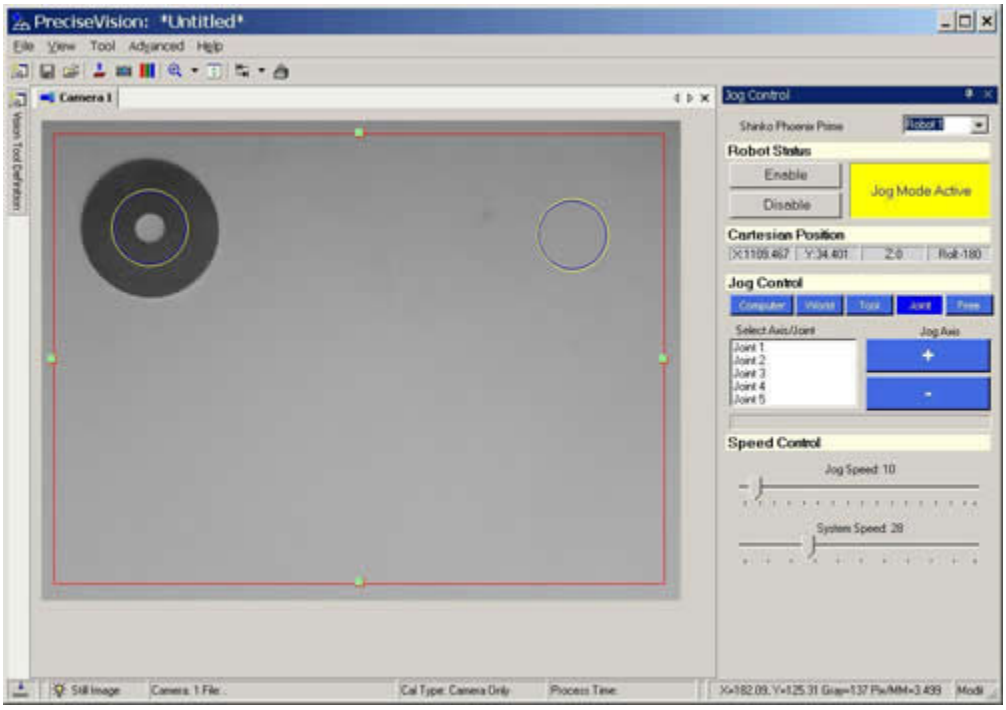
The following series of pictures illustrates how a disk is moved to various quadrants of the camera's field of view and PreciseVision locates disk and marks the spot with a circular indicator. At each position, the robot's location also needs to be recorded.

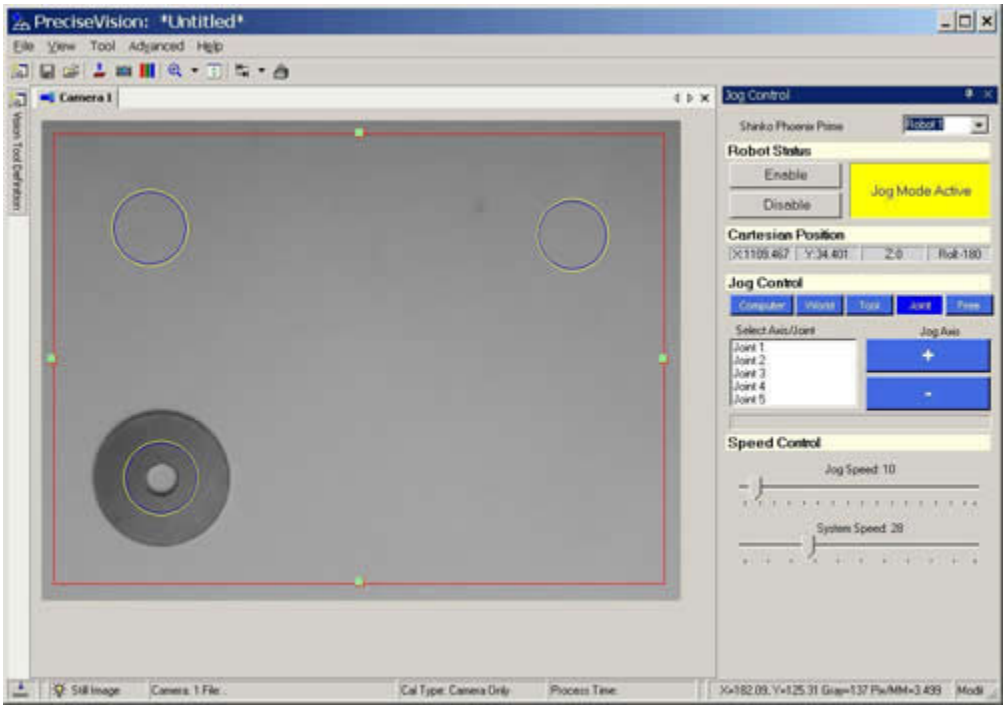


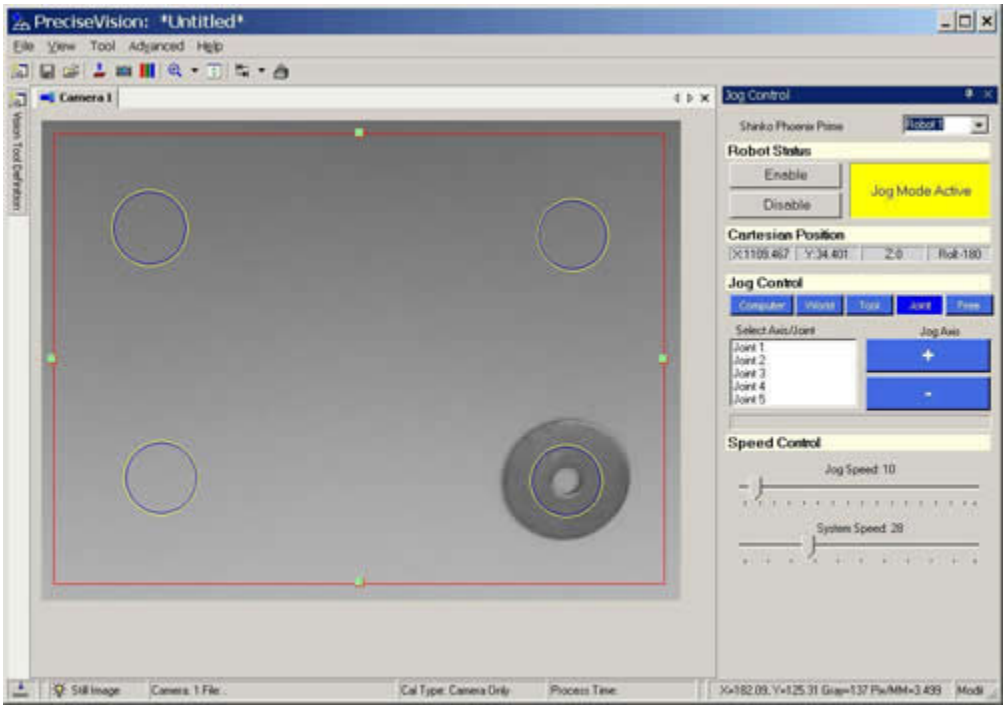












After this or similar data is collected, PreciseVision will automatically compute the required transformation between the camera and the robot. This information is then stored in a camera calibration file along with the camera calibration data for subsequent use.

This completes the overview of the Robot Vision Area Camera Calibration procedures.

Appendix A: FAQ

Frequently Asked Questions

This section contains a compilation of frequently asked questions related to PreciseVision.

A. General Questions

B. IDS uEye USB Cameras

General Questions

What do you do if PV was working but will no longer properly start up?

Sometimes the preferences information for PreciseVision can become corrupted or can reference a PV project that is corrupted. When this occurs, PV will fail to launch. To work around this problem, do the following.

1. Locate the folder where PreciseVision is installed. Typically, this will be "C:\Program Files\Precise Automation\PreciseVision V__\".
2. Rename your preferences file "Preferences.vpr" to "Preferences.bad".
3. Restart PreciseVision.

If PreciseVision starts properly, please email your bad Preferences file and your PreciseVision projects to support@preciseautomation.com with a short explanation so we can diagnosis the problem.

How do I transfer my PV license to another host computer?

The PreciseVision software is only licensed to execute on a single PC. In general, if you need to execute PV on additional PC's, additional licenses must be purchased.

However, if the PC on which PV is licensed to execute is being obsoleted, you can transfer a purchased PV license to a new PC.

Note, as soon as you execute the following procedure, PV will no longer be permitted to execute on your old PC.

To transfer a purchased PV license to another PC, perform the following procedure.

1. Follow the standard procedure for installing PV on the **new** PC.
2. On the **new** PC, execute the procedure for licensing PV to obtain the PC's ID information. This is needed to issue you a new license for this PC.
3. Execute PV 2.0 or later on the **old** PC. If you are executing an older version of PV, obtain a new version from the Precise Support website and install it on the **old** PC.
4. Select *"Help > Product Activation*".
5. Click the *"Uninstall"* button.
6. Follow the instruction to execute the uninstall procedure. *After you execute this procedure, you will no longer be able to execute PV on the **old** PC.*
7. Email the following information to sales@preciseautomation.com and request that the license be transferred.
    - Your company name
    - The name of the individual that the license was issued to, if you know
    - The original license code displayed by the uninstall procedure
    - The uninstall code displayed by the uninstall procedure
    - The new PC's ID displayed by the licensing procedure
8. Once you obtain your new license key, execute the standard activation process on the **new** PC.

This completes the license transfer process.

What PC TCP/IP ports are used by PreciseVision?

PreciseVision listens on port 1410 for connections from a Precise Controller. PreciseVision Remote PC control listens on port 1450 for connections from a command client. If you are using the Multi-Instance PV option, the ports numbers are incremented by 1 for each additional instance, as shown below.

PV Instance	GPL to PV Port	Remote PC Control Port
1	1410	1450
2	1411	1451
3	1412	1452
4	1413	1453

5	1414	1454
6	1415	1455
7	1416	1456
8	1417	1457

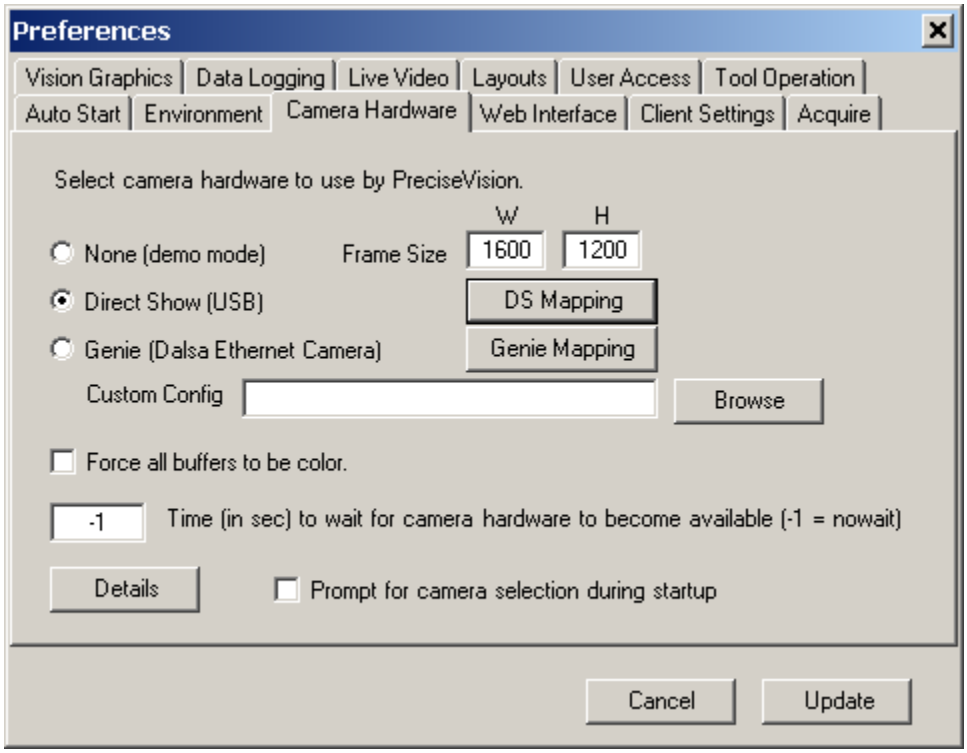
IDS uEye USB Cameras

What do you do if you are not able to get an image with an IDS or USB camera?

If you are not able to get a image from an IDS or other USB camera from within PreciseVision, there are several possible sources for this problem. Most of the problems and remedies described below apply to systems that are being executed for the first time, but some are appropriate for systems that have already operated properly.

A. IDS uEye/USB cameras may not be configured as the hardware to use with PreciseVision. PreciseVision can operate with a variety of USB and Ethernet cameras and can even operate in a demonstration mode without a camera. However, at any given time, only one camera type is permitted. The following procedure verifies that PreciseVision is configured to capture images from IDS uEye or other USB cameras.

1. Launch PreciseVision, typically by selecting *Start* > *Programs* > *Precise Automation* > *PV_* > *PreciseVision.*
2. Open the Preferences pop-up by selecting the following within PV: *File* > *Preferences*.
3. Select the "Camera Hardware" tab in the Preferences pop-up window.
4. Ensure that the "Direct Show (USB)" option is selected.
5. Restart PV to put this change into effect.
6. Return to this panel and click on the "DS Mapping" button to select the USB cameras to utilize and define their operational parameters: image resolution, color or grayscale mode, and whether the image is to be reversed.



B. <u>The required software packages may not be installed.</u> To utilize the IDS uEye cameras with PV the following software must be installed on the PC: PreciseVision, DirectShow (DirectX) and the IDS camera driver. DirectX is shipped as part of the Windows operating system and should therefore not be a problem. However, if the other software packages are missing, you will not be able to capture camera images.

1. To verify that all of the required software has been installed, bring up the Windows Control Panel typically by selecting *Start > Settings > Control Panel.*
2. Select *"Add or Remove Programs".*
3. Verify that the following packages are installed: (i) IDS uEye, and (ii) PreciseVision.
4. If either of these packages are missing, refer to the PV installation instructions for installing the missing application packages.

C. <u>Cameras may not have proper unique or registered ID's.</u> Each IDS camera must have a unique ID stored into its NVRAM and these ID's must be registered with DirectShow in order for PreciseVision communicate with the camera. If an IDS camera is being used and it has the default ID of "1", it should be possible to communicate with the camera. However, if multiple IDS cameras are being used or the default camera ID is not set to "1", one or more cameras may not be operating properly.

Please refer to the PreciseVision IDS Camera Interface Installation section for instructions on setting the camera ID's and registering the ID's with DirectShow.

D. <u>The camera resolution or display mode may be incorrect or incompatible.</u> USB cameras typically have several different resolutions that they will operate at and several different formats that can be used to fetch their image data. The formats that are supported by PreciseVision and its DirectShow interface, which are the most common formats, are as follows:

RGB8 (8-Bit Grayscale)
RGB24 (24-bit color with 8-bit R,G,B components)
YUY2 (16-bit color)

All IDS cameras have resolutions and formats that are compatible with PreciseVision. However, if you are not able to view the image, please refer to the IDS Camera Interface Installation section for a description of how to select the resolution and display format.

If you are using a third party camera, there is a possibility that your camera driver does not support DirectShow or that it does not have an image format that is compatible with PreciseVision. If this is the case, you will not be able to use the camera with this software. However, please refer to the IDS Camera Interface Installation section for a description of how to select the resolution and display format to see if these setting can resolve your problem. ***Please keep in mind that Precise cannot provide assistance with cameras that are not tested and fully supported by Precise.***

What IDS uEye camera trigger and strobe parameters are available?

Most IDS uEye cameras have a hardware input signal for triggering the start of a camera acquisition and a hardware strobe output signal for triggering a strobe to freeze moving parts. In their native operating mode, IDS cameras provide a number of parameters for controlling the operation of these two hardware signals. Unfortunately, IDS does not currently provide the full set of parameters when a camera is operated via DirectShow, as is the case for PreciseVision.

The following table describes the parameters that can currently be modified within PreciseVision and the fixed values for those parameters that cannot currently be altered.

IDS Camera Parameter	Setting Within PreciseVision
Trigger Mode	This specifies if the trigger occurs on the rising or falling edge of the external hardware input trigger signal. This parameter can be specified within PreciseVision by setting the **TriggerActive** property of the **Acquisition Tool** that defines the picture to be taken.
Trigger Delay	This defines the delay between when an external hardware trigger signal is received and the start of the camera acquisition. This is currently fixed at 0 microseconds.
Flash Delay	This is the delay between when an external hardware trigger or internal software trigger (from PreciseVision) arrives and the strobe output is asserted. This is currently fixed at 0 microseconds.
Flash Duration	This is the time that the hardware output strobe signal is asserted high when the strobe is triggered. This is currently fixed at 5000 microseconds.

Appendix B: Remote PC Control

PreciseVision Remote PC Control

Even though PreciseVision executes as a standalone application within a PC environment, its user interface and vision processing capability can be integrated with the user interface of another PC application. PreciseVision provides the ability for any PC application to send commands to PreciseVision using TCP/IP. These commands can specify the format of the PV window and its size and position on the screen. This set of commands also includes methods for triggering the execution of vision processes and for reading and writing data within PreciseVision. So, these commands allow any PC application environment to manipulate PreciseVision to look and feel like it is part of the environment without the need for complicated installation of COM objects or distribution issues.

The communication interface uses simple string messages that are passed through an open TCP socket connection. When PreciseVision begins execution, it automatically runs a remote TCP server in a background thread. This thread listens on port (1450) and is dedicated to servicing commands from external applications. This interface can be tested and debugged using any Telnet session connected via the communication port.

Whenever the communication connection is lost, the PV user interface is automatically restored to its normally stand alone mode, i.e. the form title, status, menu and toolbars are all displayed.

Command Definition

Each transmitted command must conform to the following string format:

command [key] arg1 arg2 arg3 … arg20 {LF}

While in Telnet mode, the command response may be as simple as a {LF} or it may be a string that returns requested information. Please see the command description table below for details.

In general, the format of the response returned for each command is formatted as follows:

errorcode arg1 ... argN {LF}

If no error occurs, the returned "errorcode" will have a value of "0" to indicate success. If an error occurs, a negative error code is returned and "arg1" contains the string representation of the error. For some commands, "arg1 ... argN" contains information returned by PreciseVision.

Supported Commands

The following table describes the legal commands. Please note that the **CMD** name is case sensitive.

CMD	KEY	Argument List	Description & Responses
getnumresults		*<tool>*	Retrieves the number of results that exist for the specified tool. The *<tool>* must be of the form *processname.toolname.* The information returned by this command is as follows: *<err> <num_results>* where *<err>*: Standard error code (can be positive or negative) or 0 for success. *<num_results>*: The number of results available for this tool.
getresults		*<tool> <index>*	Retrieves the numeric results for a specified tool that was executed as part of a specified process. The *<tool>* must be of the form *processname.toolname.* The information returned by this command is as follows: *<err> <x> <y> <z> <yaw> <pitch> <roll> <objectID> <passed> <value> <info_array>* *where* *<err>:* Standard error code (can be positive or negative) or 0 for success. *<x> <y> <z> <yaw> <pitch> <roll>:* Cartesian coordinates for the located object. *<objectID>*: Optional index that some tools return to identify the specific instance of an object that is located. *<passed>*: 0 if tool's inspection was successful or -1 if the inspection failed. <value>: Value of the tool property that was tested by the vision inspection process. *<info_array>*: Please see the information on the specified tool for a description of any additional result values that are returned.
getstringresults		*<tool> <index>*	Retrieves the results for tools that return a string value instead of numerical results. For example, this command should be used for the Barcode Reader tool. The *<tool>* must be of the form *processname.toolname.* The information returned by this command is as follows: *<err> <string>*

			where *<err>:* Standard error code (can be positive or negative) or 0 for success. <*string*>: String value that is returned by the tool. This is identical to the output that is sent to a Guidance Controller. Please see the information on the specified tool for a description of any string result values that are returned.
load	image	<*file_name*>	Loads an image from a specified file into the camera display buffer. <*file_name*>: The full path and name of a image file to be loaded.
load	importproject	<*file_name*>	Loads a PreciseVision project from a specified file and mergers its contents with the currently loaded project. If the new project contains any processes or tools whose names conflict with items that are already loaded, "_r" is appended to the name of the new item. Also, any camera calibration information that is contained in the new project is ignored. <*file_name*>: The full path and name of the PreciseVision project file to be loaded and merged.
load	layout	<*file_name*>	Loads a window layout given a specified path and file name. <*file_name*>: The full path and name of a compatible PV layout file. This type of file is created when a layout is manually saved.
load	project	<*file_name*>	Loads a PreciseVision project file from the disk. <*file_name*>: The full path and name of the PreciseVision project file to be loaded. This allows a PC based application to change the project or override the default project loaded when PV starts.
property	set/get	<*property*> <*value*>	Reads or writes the value of a specified property of a specified tool. The <*property*> must be of the form *toolname.propertyname.* For write operations, <*value*> defines the new property value that is set. Not all properties are available for reading and/or writing. Please see the vision tool definitions for details on each tool type's

			properties. As a means for transmitting system information between PreciseVision and Precise Guidance Controllers, a special built-in *"System"* tool always exists. For the most part, this tool duplicates functionality available via explicit Remote PC Control calls. However, some functionality is only available via the System tool. If required, the System tool can be accessed by specifying it as the *toolname* in this command. For a explanation of the available *propertyname*'s, please consult the *Guidance Programming Language Dictionary Pages*.
run		<*process*>	Initiates execution of the specified vision process.
set	displaymode	<*mode*>	Dictates the contents and visibility of the PV window. The permitted values for <*mode*> are as follows. 0: Resets display mode to normal display. PV returns to stand alone mode with all normal windows, toolbars, menus etc. available for use. 1: Displays only the camera window and the current Tool's window. The PV form border, title, and status bars are all hidden. 2: Displays only the camera window. The PV form border, title, status bars and all dockable controls are all hidden. 3: Minimizes the PV window.
set	layout	<*layout_type*>	Selects one of the predefined panel layouts to display within the PV window based upon the value of <*layout_type*>: 0: Edit mode 1: Runtime mode 2: Calibration mode
set	position	<*xpos*> <*ypos*> <*width*> <*height*>	Positions and sizes the PV window relative to the screen. <*xpos*>: X position relative to top left of screen <*ypos*>: Y position relative to top left of the screen <*width*>: width to set window <*height*>: height to set window All units are in pixels.

			Note: <*width*> and <*height*> are optional. If not specified, only the X,Y position of the window will be altered.
set	topmost	<*mode*>	Specifies whether the PV window is on top or other windows. The value of <*mode*> is interpreted as follows: 0: Normal (resets topmost property) 1: Keeps window on top. This allows PV to stay on top of another application while the user clicks or drags windows.
set	zoom	<*scale_factor*>	Selects the Zoom scale factor for the camera window. The <*scale_factor*> ranges from 0 to 5 in steps of 0.1. A value of 0 will automatically set the zoom so that the entire frame buffer will be displayed within the camera display window.

Appendix C: Multi-Instance Execution

PreciseVision Multi-Instance

For most applications, a single copy of PreciseVision is connected to a single robot. If an application requires PreciseVision to communicate with multiple robots or if one or more robots need to overlap the processing of several cameras, the Multi-Instance version of PreciseVision should be utilized. This version of the software permits several instances of PreciseVision to be executed concurrently on the same PC.

Each instance of PreciseVision executes completely independently of other instances. In particular, each instance:

1. Connects to its own set of cameras with their own camera calibration data. Individual cameras cannot be shared by different PV instances. Different instances can simultaneously capture images from separate cameras subject to the bandwidth limitations of the PC. In the standard version of PreciseVision, if two or more cameras are connected to a PC, pictures must be sequentially captured and analyzed. In a PC executing multiple instances of PreciseVision, each instance can take a picture at any time without regard to any acquisitions being performed by another instance, subject to the bandwidth limitations of the PC's hardware interfaces.
2. Can only connect to a single camera interface type (i.e. Ethernet GigE or USB), but different instances can be connected to different camera interface types.
3. Connects to a single robot or a single motion controller or nothing. This allows multiple robots to share a single PC and permits a robot or motion controller to perform simultaneous frame grabs and vision analysis by referencing different instances of PV on the same PC. In the standard version of PreciseVision, only a single robot or motion controller can be connected to a single instance of PV running on the PC.
4. Loads it own Vision Project and executes its own Vision Processes. This enables each robot to execute either the same Vision Process or independent Vision Processes.
5. Operates with its own set of Preferences. This enables independent control of display options and other parameters.

Software Installation and Activation

The Multi-Instance capability is only supported in PreciseVision 3.0 and later versions. Both the standard and the Multi-Instance software are provided in the same software installation distribution. To load the software and camera drivers, follow the standard software installation instructions detailed in the *Supported Hardware and Software Installation Chapter* of this manual.

The only difference in the installation procedure is the specific software license key that is utilized to activate the software. The “PreciseVision Multi-Instance” key must be installed in place of the standard “PreciseVision” key.

> **NOTE**: When the PreciseVision Multi-Instance key is installed, it replaces any previously installed standard key. Unfortunately, this will de-activate any 2.x and older versions of PreciseVision that are still installed on the PC.

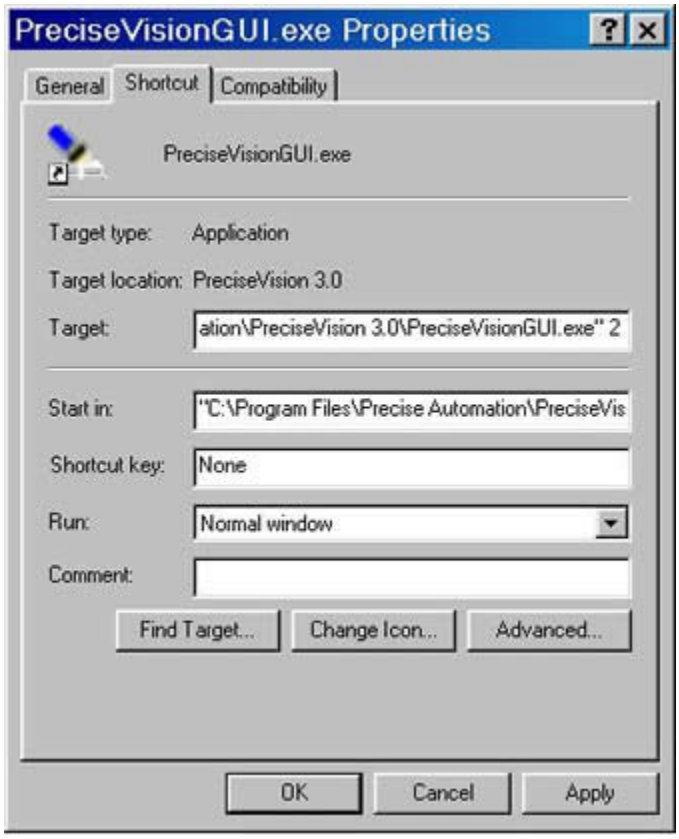
Executing Multiple Instances

Once PreciseVision 3.0 or later is loaded and the Multi-Instance key is installed, multiple instances can be executed. Each time the *Start* > *Programs* > *Precise Automation* > *PV 3.0* > *PreciseVision* menu item is selected; only the first instance of PreciseVision is initiated.

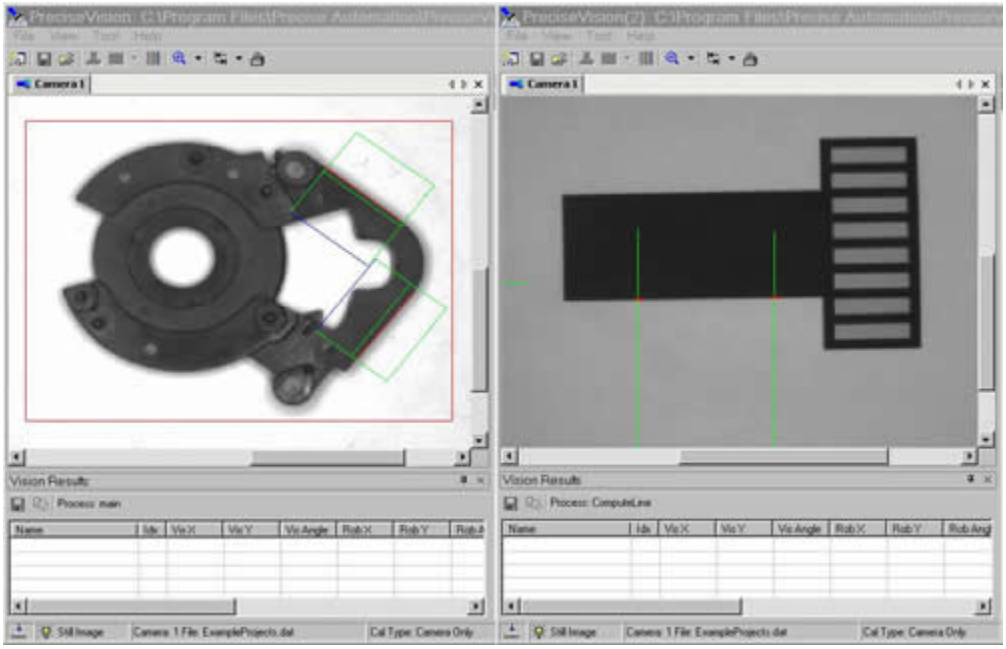
To execute instances 2 to N, a command line switch must be specified when starting PreciseVision. In Windows, the simplest means for accomplishing this is to create a “Shortcut” to PreciseVision.

1. To create the shortcut, locate the PreciseVisionGUI.exe file. This is normally stored in *C:\Program Files\Precise Automation\PreciseVision x.x*\.
2. Right click on this file and select *Create Shortcut*.
3. Right click on the new shortcut and select *Properties*.
4. On the Shortcut tab of the Properties page, edit the *Target* to add a number from 2 to N after the “.exe” to specify the instance of PreciseVision that is to be launched by the Shortcut.

At the conclusion of this process, a typical Shortcut Property page that would launch the second instance of PreciseVision will look like the following:



Below is a screen shot that illustrates two instances of PreciseVision Multi-Instance simultaneously executing. Pleases note that the second instance is recognizable by the "(2)" following "PreciseVision" name in the title bar.



Once multiple instances of PV have been launched, a robot controller can communicate with a specific instance of PV by executing the GPL *vision_object*.**IPAddress** and *vision_object*.**Instance** methods to specify the specific PC and instance of PV to be accessed.

Appendix D: Error Messages and Warnings

PreciseVision Error Messages and Warnings

PreciseVision displays error messages and warnings in the Vision Results window and the Tool Property Lists. Errors indicate more significant problems and may stop the execution of a Vision Process. Warnings are typically benign. Most of the error messages and warnings are self explanatory. The following table describes some of these messages.

Code	Text	Type	Description
	ACQUIRE_DIRECT_SHOW_BAD_SELECTED_RESOLUTION	Error	This is an internal check error that should not occur. It indicates that the index number for the resolution of a camera is out-of-range for the selected camera. If this error is generated, please report it to Precise together with the PV version, camera model number and camera driver version.
	ACQUIRE_DIRECT_SHOW_COPY_IMAGE_FAILED	Error	This error should rarely happen. It indicates there was a problem copying the acquired image from the DirectShow buffer to the camera image buffer. This could be caused by a faulty camera.
	ACQUIRE_DIRECT_SHOW_DEVICE_NOT_OPEN	Error	This error indicates that when a picture acquisition was performed, the camera was not opened as a DirectShow device. This is an unusual situation and may indicate that the camera cable has been disconnected or the camera signal has been lost for some reason.
	ACQUIRE_DIRECT_SHOW_ERROR	Error	This is a general error that is generated when any problem occurs with acquiring an image using a USB camera. In PV 4.0 and later, more specific errors are generated and this message should not be displayed. If you encounter this error, please report it to Precise together with the PV version, camera model number and camera driver version. In systems before PV 4.0, the most common problem indicated by this error is that the camera's pixel clock is set too fast and is exceeding the capabilities of the PC's USB hardware. Other problems include: incorrect exposure time,

			gain, offset or binning settings; a problem with the camera's trigger; excessive time waiting for the acquire to complete; or an error copying the DIB image to an image buffer.
	ACQUIRE_DIRECT_SHOW_ERROR_TYPE1	Error	This is a diagnostic error that should not occur. If this error is generated, please report it to Precise together with the PV version, camera model number and camera driver version.
	ACQUIRE_DIRECT_SHOW_FRAME_RESET_EVENT_FAILED	Error	This is a very unusual error that should not occur. If this error is generated, please report it to Precise together with the PV version, camera model number and camera driver version.
	ACQUIRE_DIRECT_SHOW_LIVE_ACQUIRE_STOP_TIMED_OUT	Error	Not currently used.
	ACQUIRE_DIRECT_SHOW_OPEN_DEVICE_FAILED	Error	This error is generated at a low-level between DirectShow and the camera, and then propagated up to PV. It is typically caused by the camera settings calling for more performance than the camera and computer can reliably provide or a hardware problem such as noise in the camera cable. The performance is a function of: the camera model; computer CPU chipset; computer interface port chipset (which can be different for different ports of the computer); camera driver and firmware; and the cables. To correct this problem, try lowering the properties that control performance, such as the pixel-clock rate and the frame rate; try an alternate USB port on the computer; verify you are using up-to-date camera drivers; and ensure the cables are shielded and in good order.
	ACQUIRE_DIRECT_SHOW_PROPERTY_ASSERTING_ACQUIRE_FAILED	Error	This error is generated when a picture acquire command fails after the steps that set DirectShow properties have succeeded. The likely problems are: (1) one or more of the properties is causing the acquire to fail or be unreliable or (2) the new properties are calling for more performance than the camera and computer can reliably provide or a hardware problem such as noise in the camera cable. The performance is a function of: the camera

			model; computer CPU chipset; computer interface port chipset (which can be different for different ports of the computer); camera driver and firmware; and the cables. To correct this problem, try reverting the properties to a known good set, try lowering the properties that control performance, such as the pixel-clock rate and the frame rate; try an alternate USB port on the computer; verify you are using up-to-date camera drivers; and ensure the cables are shielded and in good order.
	ACQUIRE_DIRECT_SHOW_RESOLUTION_NOT_USABLE	Error	This is an internal check error that should not occur. It indicates that the index number for an unable resolution of a camera has been permitted. If this error is generated, please report it to Precise together with the PV version, camera model number and camera driver version.
	ACQUIRE_DIRECT_SHOW_SET_DEVICE_PROPERTY_FAILED	Error	This error is generated at a low-level between DirectShow and the camera, and then propagated up to PV. It is typically caused by the camera settings calling for more performance than the camera and computer can reliably provide or a hardware problem such as noise in the camera cable. The performance is a function of: the camera model; computer CPU chipset; computer interface port chipset (which can be different for different ports of the computer); camera driver and firmware; and the cables. To correct this problem, try lowering the properties that control performance, such as the pixel-clock rate and the frame rate; try an alternate USB port on the computer; verify you are using up-to-date camera drivers; and ensure the cables are shielded and in good order.
	ACQUIRE_DIRECT_SHOW_SET_HW_TRIGGER_FAILED	Error	When attempting to setup a hardware or software triggered picture acquisition process, the messages for setting up the picture acquisition failed. These errors indicate that a DirectShow API published by the camera vendor failed. Please send a bug report for these problems to Precise including the camera model number, the camera driver version and (for hardware triggers) the selected hardware trigger polarity.
	ACQUIRE_DIRECT_SHOW_SET_SW_TRIGGER_FAILED		

	ACQUIRE_DIRECT_SHOW_SET_MODE_FAILED	Error	When attempting to setup a software or hardware triggered picture acquisition process, the "SetMode" DirectShow method failed. This error indicates that a core DirectShow API failed. Please send a bug report for this case to Precise including the camera model number and the camera driver version.
	ACQUIRE_DIRECT_SHOW_STREAM_DEVICE_ONESHOT_ENABLE_FAILED	Error	These errors indicate a problem within DirectShow for the camera stream device. Try reinitializing the DirectShow driver, disconnect and reconnect the camera, run standard DirectShow acquisition programs (if any) to verify the normal operation of the stream device, and then re-verify the operation of DirectShow by running PV. If the error persists, please report a bug to Precise including the camera model number and the camera driver version.
	ACQUIRE_DIRECT_SHOW_STREAM_DEVICE_ONESHOT_DISABLE_FAILED		
	ACQUIRE_DIRECT_SHOW_STREAM_DEVICE_RUN_FAILED		
	ACQUIRE_DIRECT_SHOW_STREAM_DEVICE_STOP_FAILED		
	ACQUIRE_DIRECT_SHOW_TRIGGERED_TIME_OUT	Error	This error typically indicates that the camera settings are calling for more performance than the camera and computer are able to reliably provide. The performance is a function of: the camera model; computer CPU chipset; computer interface port chipset (which can be different for different ports of the computer); camera driver and firmware; and the cables. To correct this problem, try lowering the properties that control performance, such as the pixel-clock rate and the frame rate; try an alternate USB port on the computer; verify you are using up-to-date camera drivers; and ensure the cables are shielded and in good order.
	FINDER_DIFFERENT_CALIBRATIONS	Error	If the camera calibration pixel-to-mm X or Y scale factors are changed after a **Finder Tool** template has been taught, the template must be retrained. While the **Finder** produces results in calibrated units, some of its calculations rely upon these scale factors.
64	FITLINE_ALLOC_FAILED	Error	Not enough memory available
256	FITLINE_INCONSISTANT_ARGS	Error	The specified properties are inconsistent

32	FITLINE_ INVALID_ ACQUISITION	Error	The previous camera acquisition failed
8	FITLINE_ INVALID_CAM_IMAGE	Error	The camera image data is invalid
128	FITLINE_ MAP_IMAGE_FAILED	Error	Not enough memory available
2	FITLINE_ NOMINAL_OUTSIDE_ IMAGE	Error	No part of the tool's nominal line is within the image
4	FITLINE_ TOO_FEW_POINTS_ FOR_FIT	Error	Two good edges could not be found
1	FITLINE_ UNREASONABLE_ COORD	Error	Coordinates values are very large or very small
16	FITLINE_ WIDTH_TOO_SMALL	Error	The MaxEdgePoints property value is < 2
-4024	VISION_TOOL_ NO_RESULTS	Warning	Indicates that a vision tool generated results data, but that the tool failed its inspection operation. Normally, this error is not seen because if a tool fails, it will not produce any results. However, if the **ResultOnNotFound** property is TRUE, vision tools generate results for successful and failed operations. This warning marks results that were produced by a failed tool.
-4025	VISION_TOOL_ RESULTS_MISMATCH	Error	This error is typically generated by an **Inspect List Tool** if its output is invalid because one or more of the referenced tools did not generate the same number of results. This can occur if the **Inspect List** operates on the output from a tool that generates multiple results, such as a **Finder**, and the **Inspect List** contains multiple inspection operations. If one or more of the inspection tools fail, they will not generate results so the **Inspect List** will be operating on arrays of results that are different in length. To resolve this problem, in the Preferences, turn on "Return results if not found". This will enable the **ResultOnNotFound** property and force tools to return results even if if the tool's inspection operation fails.