



User Guide



HexSight 4.4 User Guide

Copyright

Copyright (c) 1998-2014 by LMI Technologies

Disclaimer

LMI Technologies shall not be responsible for any errors that may appear in this document and shall not be liable for any damages arising out of or related to this document or the information contained in it, even if LMI Technologies has been advised of the possibility of such damages.

Trademark Information

HexSight is a trademark of LMI Technologies

The following are trademarks or registered trademarks of the associated company:

.NET, ActiveX, COM, C + +, C#, DirectShow, DirectX, MFC, Notepad, Visual Basic, Visual C + +, Visual Studio, Windows, Windows XP, Windows Vista, HTML Help: Microsoft Corporation

Acrobat Reader, PDF: Adobe Systems Incorporated

DCAM, IIDC: 1394 Trade Association, Institute of Electrical and Electronics Engineers

DepictMaster4X1: Opteon Corporation

DT-3152, DT-3155, DT-3157: Data Translation, Inc.

FireWire: Apple Computer, Inc.

GenICam: European Machine Vision Association

GigE Vision: Automated Imaging Association

GretagMacbeth, ColorChecker: X-Rite, Inc.

IC-ASYN, PC-DIG, PCVision, ITEX, IFC: Dalsa Corporation

Meteor-II/Digital, Meteor-II/Multi-Channel, Meteor-II/Standard, Meteor-II/CameraLink, Orion, MIL-Lite:

Matrox Electronic Systems Ltd.

Pentium, MMX: Intel Corporation

PX610A, PXC200: Imagenation Corporation

Sony Intelligent Camera, Sony ICam: Sony Corporation

Gocator: LMI Technologies

Table of Contents

Table of Contents	1
Conventions Used in this Document	5
Installing HexSight	7
HexSight 32-bit versus 64-bit	7
Contents of your HexSight Kit	8
HexSight Online Documentation	8
System Requirements	8
Installing HexSight Software	10
Installing Frame Grabbers and other Devices	12
Getting Started	15
What is HexSight?	16
Getting Help	27
Project Examples Provided with HexSight	28
Demonstration Mode	34
Color Support and Functionality in HexSight	37
Color Image Acquisition	37
Color Calibration	38
Color Locator	39
Configuring a Color Locator Tool	40
Color Processing with Inspection Tools	42
Color Finder Tool	44
VB .NET Tutorials	47
VB .NET Tutorial 1: Locator	48
VB .NET Tutorial 2: Display	69
VB .NET Tutorial 3: Caliper	73
VB .NET Tutorial 4: Blob Analyzer	82
VB .NET Tutorial 5: Pattern Locator	90
VB .NET Tutorial 6: Edge Locator	97
VB .NET Tutorial 7: Calibration	107
Visual C++ Tutorials	111
Visual C++ Tutorial 1: Locator	112

Visual C++ Tutorial 2: Display	140
Visual C++ Tutorial 3: Caliper	144
Visual C++ Tutorial 4: Blob Analyzer	153
Visual C++ Tutorial 5: Pattern Locator	161
Visual C++ Tutorial 6: Edge Locator	168
Visual C++ Tutorial 7: Calibration	178
C# Tutorials	181
C# Tutorial 1: Locator	182
C# Tutorial 2: Display	203
C# Tutorial 3: Caliper	207
C# Tutorial 4: Blob Analyzer	216
C# Tutorial 5: Pattern Locator	224
C# Tutorial 6: Edge Locator	231
C# Tutorial 7: Calibration	241
Using HexSight Displays	245
Introduction to HexSight Displays	246
Elements of HexSight Displays	247
Coordinate Systems	251
Right-Handed vs. Left-Handed Coordinate Systems	251
Relationship between Coordinate Systems	252
World Coordinate System	253
Image Coordinate System	254
Object Coordinate System	255
Tool Coordinate System	256
Unit and Angle Conversions	257
Application Control	259
Accessing the Application Control	259
Using the Application Control	261
Application Control Properties	273
Locator	277
Understanding Locator Concepts	278
Using the Locator	332
Acquisition Device	381
Understanding Acquisition Device Concepts	381
Using the Acquisition Device	390

Inspection Tools	481
List and Description of HexSight Tools	482
Inspection Tool Basics	484
Using Inspection Tools	500
Caliper	509
Understanding Caliper Concepts	510
Using the Caliper	518
Edge Locator	537
Understanding Edge Locator Concepts	538
Using the Edge Locator	545
Arc Caliper	563
Understanding Arc Caliper Concepts	564
Using the Arc Caliper	576
Arc Edge Locator	597
Understanding Arc Edge Locator Concepts	598
Using the Arc Edge Locator	609
Blob Analyzer	629
Understanding Blob Analyzer Concepts	630
Using the Blob Analyzer	645
Pattern Locator	673
Understanding Pattern Locator Concepts	674
Using the Pattern Locator	681
Color Finder	699
Understanding Color Finder Concepts	700
Using the Color Finder	701
Point Finder	719
Using the Point Finder	719
Line Finder	737
Using the Line Finder	738
Arc Finder	759
Using the Arc Finder	760

Builder Tool	781
Using the Builder Tool	782
Image Processing Tool	793
Understanding the Image Processing Tool	794
Using the Image Processing Tool	808
Image Histogram	817
Understanding Image Histogram Tool Concepts	818
Using the Image Histogram Tool	823
Image Sharpness Tool	837
Understanding Image Sharpness Concepts	837
Using the Image Sharpness Tool	840
Sampling Tool	855
Understanding Sampling Tool Concepts	856
Using the Sampling Tool	860
Evaluator Tool	873
Using the Evaluator Tool	874
Output Tool	879
Using the Output Tool	880
Barcode Reader	885
Understanding Barcode Reader Concepts	886
Using the Barcode Reader	893
Data Matrix Reader	915
Understanding Data Matrix Reader Concepts	916
Using the Data Matrix Reader	927
OCR Fixed Font	953
Understanding OCR Fixed Font Concepts	954
Using the OCR Fixed Font	962
Class Library	985
Relationship Between Database Objects	986
Visual Basic Example	987
Visual C++ Example	989

Conventions Used in this Document

Bold	User interface elements: E.g. command buttons Keywords such as tool names, process names Keys and keyboard combinations
Capitalized Terms	Specific terms as they apply to HexSight applications E.g. “View” as opposed to the common use sense of “view”
<i>Italics</i>	Typed user entries Emphasis and special notes to the reader
<i>Bold italics</i>	Designates a specific <i>Book, Chapter, Section...</i>
Courier typeface	Programming example code Enumeration of coded properties & methods
<u>One character underline</u>	Menu options, pull-down menus, menu paths E.g. <u>F</u> ormat> <u>E</u> nt means select Font from the Format menu

Installing HexSight

This section presents information required for installing HexSight.

HexSight 32-bit versus 64-bit

HexSight is available in 32-bit and 64-bit versions. The 32-bit version can be installed on both 32-bit and 64-bit Windows, whereas the 64-bit version can only be installed on 64-bit Windows. It is recommended to use the HexSight version that corresponds to the target platform of the final application. See the following table for a summary.

Table 1 Development system and target applications

Development system	Target applications
32-bit Windows / 32-bit HexSight	32-bit applications
64-bit Windows / 32-bit HexSight	32-bit applications
64-bit Windows / 64-bit HexSight	32-bit or 64-bit applications

For 64-bit application development, the 64-bit HexSight binaries must be used when specifically targeting x64 platforms, when using other 64-bit libraries from third parties, and when creating applications that require a large amount of memory. When running natively on x64 platforms, HexSight can use much more memory than its 32-bit counterpart, allowing the user to work with bigger images and more complex configurations that use more tools.

Note: Symbology tools are only available in 32-bit applications.

Contents of your HexSight Kit

Ensure that your HexSight kit is complete. It should contain the following items:

1. Installation CD
2. One hardware key (dongle)
3. The **HexSight Quick Start Guide**, printed version
4. One software license agreement

HexSight Online Documentation

Note: The HexSight documentation set is comprised of the Reference Manual and two User Guides: HexSight Basics and HexSight Tools. These manuals are in PDF format and can be installed from the HexSight installation CD. The manuals are also contained in the Online HexSight HTML Help. Online help is context sensitive from the interface of the Visual Basic or C++ code editor.

System Requirements

To run HexSight on your computer, the following hardware and software are required.

Hardware Requirements

Processor: Pentium II or higher

Hard Drive space: From 100 MB to 500 MB depending on selected components

Memory (RAM): 128 MB

PCI frame grabber board. HexSight 4.4 supports the following frame grabbers:

- Dalsa IC-ASYN
- Dalsa PC-DIG
- Dalsa PCVision
- Data Translation DT-3152
- Data Translation DT-3155
- Data Translation DT-3157
- Imagenation PX610A
- Imagenation PCX200
- Matrox Meteor-II/Standard
- Matrox Meteor-II/Multi-Channel
- Matrox Meteor-II/Digital
- Matrox Meteor-II/Camera Link
- Matrox Orion
- Opteon DepictMaster 4X1

IEEE 1394 and USB Cameras with a DirectShow driver are supported, IEEE 1394 devices using the DCAM standard, as well as all GigE Vision compliant cameras.

Display: Minimum resolution: 1024 x 768 pixels. Colors: 65,536

CCD Camera: Refer to the frame grabber documentation.

Software Requirements

Operating system: Windows XP with Service Pack 3, Windows Vista (x86 or x64) or Windows 7 (x86 or x64).

Visual Studio .NET (2003 or higher), or other ActiveX-compliant environments.

Adobe Acrobat Reader 5.0 or higher and Internet Explorer 5.0 or higher are recommended to respectively access pdf format documentation and online HexSight Help.

Installing HexSight Software

The following three sections describe pre-installation steps as well as the Install/Uninstall procedures.

Before Installing HexSight

1. Ensure that you have Administrator privileges on your PC. You must have permission to write to the necessary system registry sections.
2. To acquire live images in HexSight you will need a compatible camera and frame grabber or a compatible IEEE 1394 or USB camera.
3. If you are using a frame grabber, we strongly recommend that you install the original driver for the frame grabber before installing HexSight. See the specific section at the end of this chapter for the frame grabbers you intend to use: [Installing a Data Translation Frame Grabber](#), [Installing an Imagenation Frame Grabber](#), [Installing a Dalsa Frame Grabber](#), [Installing a Matrox Frame Grabber](#), [Installing an Opteon Controller Board](#). Please refer to the documentation provided with the frame grabber to ensure that you follow the appropriate procedure.
4. Install the hardware key or USB key (dongle) provided with HexSight. Although it is not required for the software installation, the key must be installed to develop and save HexSight applications.
5. If you are using HexSight in demonstration (Demo) mode and/or you do not have a dongle, you will only be able to run existing applications, such as the example applications and the tutorial example provided on the HexSight CD. Any changes made to parameters in an application will be lost once you close the application.

6. Visual Studio 2003 .NET or higher should be installed on your PC before installing HexSight.
7. Close any other open applications.

Installing HexSight

1. *If you have an older version of HexSight installed on your system, you must first uninstall this version before installing the new one. See the section [Uninstalling HexSight](#) for more details.*
2. Insert the HexSight CD in the CD-ROM drive. Autorun begins the installation process as soon as you insert the CD.
3. When the **Welcome** window is displayed, choose **Next**.
4. Read the licence agreement and then click **I Agree**.
5. Choose an installation folder and click **Next**.
6. Choose a start menu folder and click **Install**.
7. HexSight will now install. When the installation is complete, click **Close** to exit the installation program.

To verify if HexSight was successfully installed on your computer, we encourage you to open and try the first part of the Visual C++ Tutorial, found in the **Tutorial\VisualC++\HookInjection\Part1** folder on the selected installation path.

Uninstalling HexSight

1. In the Control Panel of your computer, double-click **Add/Remove Programs**.
2. Find **HexSight 4.4** in the list and click **Remove**.
3. Click **Uninstall** to confirm the removal.
4. When the removal is completed, close the Add/Remove Programs window.

Installing Frame Grabbers and other Devices

The following section provides installation information for frame grabbers that are supported by HexSight. Note that you must first install the original driver for the frame grabber then apply specific requirements, if any, as described in the following sections. Once properly installed, supported frame grabbers can be configured either programmatically or through the HexSight user interface.

The [Acquisition Device](#) chapter in both the *HexSight Basics User Guide* and *Reference Manual* provide information for configuring frame grabber parameters.

Installing a Data Translation Frame Grabber

The interface DLLs needed to connect to the frame grabber driver are installed with HexSight. No specific installation procedure is required.

Installing an Imagenation Frame Grabber

In order to connect to an Imagenation frame grabber driver, you will need to copy the following interface DLL files from their installation directory (**bin** subfolder of the installation path) to the Windows system path (the **\$(windows)\system32** subdirectory).

- PX610A: WPX5_NT.DLL
- PCX200: FRAME_32.DLL, PXC_NT.DLL, PXC2_NT.DLL

Installing a Dalsa Frame Grabber

To configure a Dalsa frame grabber, HexSight uses configuration files built with the appropriate Dalsa configuration application. These configuration applications include the device drivers for the frame grabber:

- IC-ASYNC: ITEX Camera Configurator, provided with the ITEX Library SDK (Version 4.1 or higher)
- PCVision: ITEX Camera Configurator, provided with the ITEX Library SDK (Version 4.1 or higher)

- PC-DIG: IFC Camera Configurator, provided with the IFC Library SDK (Version 5.0 or higher)

Itex and IFC software is not provided with the frame grabber and must be purchased separately from Dalsa. In accordance with the Dalsa license agreement, you can however redistribute the device drivers and configuration files as long as you include a frame grabber with your application, so only one copy of the software is required for the development system.

Installing a Matrox Frame Grabber

To configure the frame grabber, HexSight uses configuration files built with the Intellicam application provided with MIL-Lite (Version 6.1 or higher), which also includes the device drivers for the frame grabber.

MIL-Lite is not provided with the frame grabber and must be purchased separately from Matrox. In accordance with the Matrox license agreement, you can however redistribute the device drivers and configuration files as long as you include a frame grabber with your application, so only one copy of MIL-Lite is required for the development system.

Installing an Opteon Controller Board

The interface DLL to needed connect to the controller board driver is installed with HexSight. No specific installation procedure is required.

Installing a DirectShow Device

Any IEEE 1394 or USB camera with a DirectShow driver is supported. All DirectShow standard parameters are accessible programmatically. No specific installation procedure is required.

Installing a IEEE1394 DCAM Device

Any IEEE 1394 using the DCAM standard is supported. It allows for greater control over the camera. The CMU DCAM driver is required for the camera to be added within HexSight.

Installing a Gocator Sensor

Any Gocator sensor 2000-2300 series is supported. No specific installation procedure is required. To add the device, use one of the discovered device on the subnet or the LMI sensor entry in the drop down list of the Add Device dialog box. Enter the IP Address and match the current Gocator configuration in the HSAcquisitionDevice parameters; HexSight does not modify the Gocator's settings, it only start and stop the sensor.

Getting Started

This chapter introduces basic concepts and procedures you will need to use HexSight vision software.

Chapter Outline

[What is HexSight?](#)

[Basic HexSight Terminology](#)

[What is a HexSight Application?](#)

[Creating a HexSight Application](#)

[Adding a Component to the Toolbox](#)

[Referencing a Control](#)

[Graphical Interface Elements](#)

[Getting Help](#)

[Project Examples Provided with HexSight](#)

[Demonstration Mode](#)

What is HexSight?

HexSight is a comprehensive library of machine vision tools, available as ActiveX controls, C++ DLL libraries, or a .NET DLL assembly.

At the heart of HexSight is the unique and powerful Locator tool that provides accurate and robust object recognition through geometric object location. HexSight uses the geometry of parts to quickly locate them in a two-dimensional field of view. The Locator finds the contours of parts in the image, and quickly identifies each part's position and orientation. This information can be sent to a robot or machine controller to guide part handling.

In addition, HexSight provides integrated calibration and a complete set of inspection tools. HexSight tools can be model-based to leverage the Locator's results by automatically transposing tools on parts regardless of their scale or rotation.

Basic HexSight Terminology

Specific terms are used in the **User Guide** and the **Reference Manual** to describe the functioning of HexSight ActiveX controls. The most important terms, fundamental to HexSight applications, are defined below.

Configuration: A configuration is a set of HexSight tool instances and the configured properties of each tool. Configuration files are saved to the HexSight cfg format. In the Acquisition Device, a Configuration is a set of parameters for a given input source.

Database: A collection of Views, which can contain any number of database objects of the type [Scene](#), [Image](#) and/or [Entity](#). The runtime database acts as a database exchange container for the HexSight processes.

Disambiguate : As applied to object location, disambiguation means to resolve the ambiguity between many possible solutions by using one or more distinguishing characteristics or features of an object.

Entity : An object that is created and used by HexSight Finder tools. Basic entities are Points, Lines and Arcs. These entities can be saved to proprietary file formats and accessed through the Class Library. See [HexSight File Formats](#).

Image: Data, represented as matrix of pixels, that represents the 2D appearance of an object. Images are typically generated by a camera.

Input: Data used by the process algorithm to compute a result, which can be an image, a Scene, or a set of numeric values.

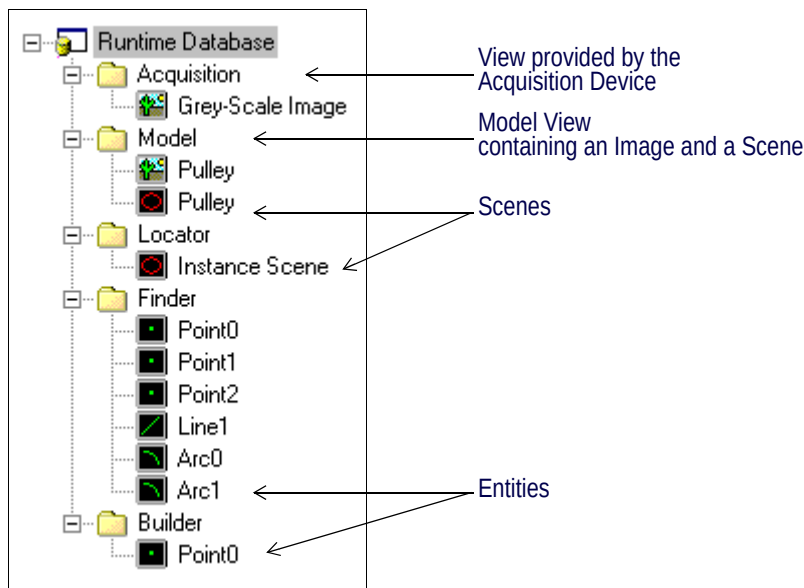


Figure 1 GUI Representation of a HexSight Runtime Database

Instance: A representation, such as a Model or a HexSight Process type.

Instantiate: To actualize an abstract representation by defining a particular configuration of the process, giving it a name, and including it in the process list. For example, if you add a Caliper tool to an application, you are instantiating a HexSight HSCaliper process.

Model: A Model is a representation of an object, based on the geometry of its contours. An object Model is the reference used to identify and locate instances of objects of the same type.

Output: Data generated by the process algorithm, e.g. an Image, a Scene, an Entity or a set of numeric values.

Properties: The characteristics of an object.

Scene: A vectorized description of an object's contours, as opposed to an Image, which is made up of pixels.

View: A collection of Scenes, Images and Entities. By analogy, one could say that a View is a folder that can contain Scenes, Images or Entities. displays an example of a HexSight Database. In this specific example the Views are named Acquisition, Model, Locator, Finder and Builder.

HexSight File Formats

The following file formats are used by HexSight:

- HexSight Views (.hvw) save all information in a View folder including Images, Scenes and Entities.
- Bitmap Images (.bmp)
- HexSight Images (.hig)
- HexSight Scenes (.hsc)
- HexSight Points (.hpt)
- HexSight Lines(.hln)
- HexSight Arcs (.hac)
- HexSight Configuration(.cfg)
- HexSight Calibration (.cal) files contain calibration information.
- HexSight Database (.hdb)
- HexSight Frame Grabber (.fgp) files contain frame grabber parameters.
- HexSight Color filter (.cff) save information for color filters that can be used by the Color Finder.

What is a HexSight Application?

A HexSight application is composed of a sequence of processes, which can also be visualized as modules, each having its own function or purpose. Each of the processes exchange data through the runtime database. A typical

HexSight application contains at least the following two processes: the **Acquisition Device tool** and the **Locator tool**.

These processes perform the following basic operations:

1. The **Acquisition Device** acquires an image of the work surface. Acquired images can be calibrated within the Acquisition Device interface.
2. The **Locator** tool generates a vectorized Contour Scene from the Images provided by the Acquisition Device. The Locator finds and locates, in the Contour Scene, all occurrences of objects of defined in the Model database. Models are created and edited in the Locator interface.

Inspection tools such as the Caliper can be added to these two basic processes depending on the task to be performed.

Creating a HexSight Application

Microsoft Visual Studio is required to create a HexSight application. As a shortcut, it is possible to compile any HexSight example provided with the development package. They are available in three languages: C++, C# or VBNET. See ***Project Examples Provided with HexSight***.

A typical HexSight application is created by placing a HexSight Application control on a blank application form, as shown in [Figure 2](#). The steps required to complete such a task are:

1. Click on the HexSight Application control icon in the Visual Studio toolbox and use the mouse to place this control on a blank form. If the icon is not available in the toolbox, see ***Adding a Component to the Toolbox*** for more details.
2. Drag a square with the mouse on the form and release the left mouse button to place the HexSight Application Control.
3. Right-click the HexSight Application Control in the form and select **Edit** to display the Process Manager as shown in [Figure 3](#). The Process Manager is used to instantiate and configure the sequence of processes.

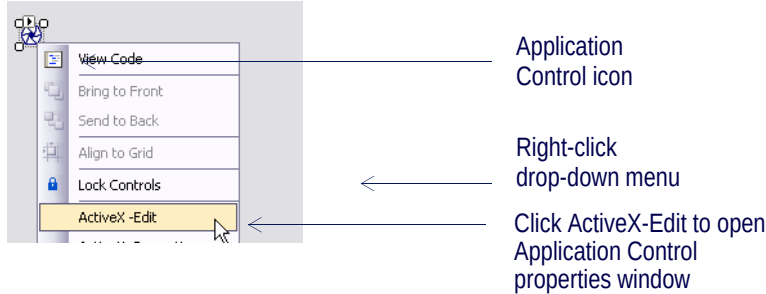


Figure 2 Application Control Placed in a Visual Basic Form

The Process Manager interface contains two lists:

- **Available Processes**, which lists all HexSight process types and
- **Instantiated Processes**, which contains the names of the processes used in the current application.

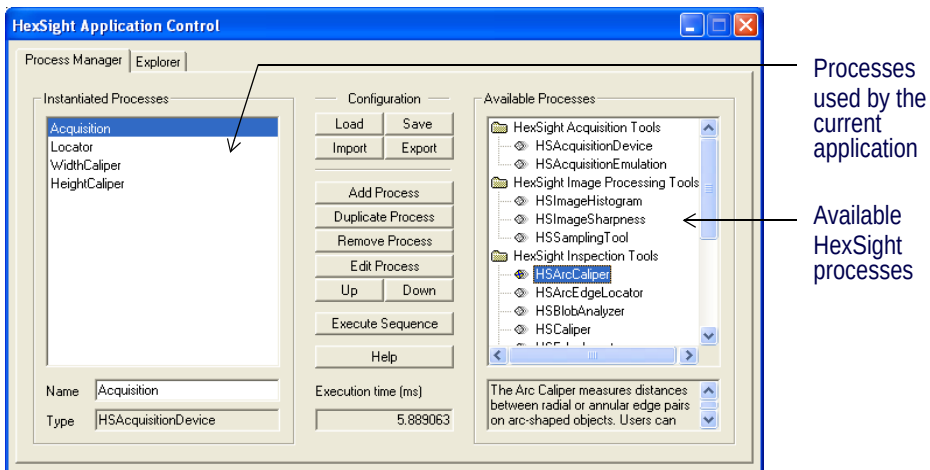


Figure 3 Process Manager

Command buttons enable you to save the configuration of the current application, modify the list of instantiated processes in the application, and execute the instantiated processes.

The following list provides a short description of the available processes, grouped by type.

Acquisition Tools

Acquisition Device: The Acquisition Device acquires images from a frame grabber or from a database of images.

Locator Tools

Locator: The Locator tool finds and locates objects that are defined by models. Models are created using the integrated Editor and saved to database files. The Locator provides the scale factor, orientation, and position for each located object.

Arc Finder: The Arc Finder finds and locates circular features on objects and returns the coordinates of the center of the arc, the start and end angles, and the radius.

Line Finder: The Line Finder finds and locates linear features on objects and returns the line angle and point coordinates.

Point Finder: The Point Finder finds and locates point features on objects and returns the angle as well as the coordinates of the found point.

Builder: The Builder creates geometric entities such as points, lines or arcs from user-defined parameters or by combining other geometric entities.

Image Processing Tools

Image Processing Tool: The Image Processing Tool processes images by applying arithmetic, assignment, logical, filtering, morphological or histogram operators. Users can define custom filtering operators.

Image Sharpness Tool: The Image Sharpness Tool computes the sharpness of preponderant edges in a user-defined region of interest.

Image Histogram Tool: The Image Histogram tool computes greylevel statistics within a user-defined region of interest.

Sampling Tool: The sampling tool is used to extract a section of an image and output it as a separate Image. The sampling tool can be configured to

operate as model-based. It can be used by a HexSight application to apply a custom processing on a region of interest on an object.

Inspection Tools

Edge Locator: The Edge Locator finds and locates an edge or a set of edges that meet user-defined criteria.

Arc Edge Locator: The Edge Locator finds and locates an edge or a set of edges in an arc- or circular-shaped area.

Caliper: The Caliper finds and locates one or more edge pairs and measures distances between the two edges within each pair.

Arc Caliper: The Arc Caliper finds and locates one or more edge pairs on an arc or circular shaped area and measures distances between the two edges within each pair.

Blob Analyzer: The Blob Analyzer finds and locates *blobs* within a defined area on an Image and returns various results for each blob.

Color Finder: The Color Finder searches and analyzes images to find areas of color that match user-defined filters.

Pattern Locator: The Pattern Locator finds and locates instances of a pattern occurring within an Image. Pattern images are provided by the HexSight Locator control.

Output Tools

Evaluator Tool: The Evaluator Tool converts results from other tools to a Pass/Fail decision by using threshold comparison and/or mathematical operations.

Output Tool: The Output Tool schedules an event on a digital IO of a LMI Gocator™ device, based on a Pass result from the Evaluator Tool.

Symbology Tools (optional)

Barcode Reader: The Barcode Reader reads and extracts information from 1D symbologies commonly known as bar codes.

Data Matrix Reader: The Data Matrix Reader reads and extracts information from 2D symbologies commonly known as data matrixes.

OCR Fixed Font: The OCR Fixed Font tool recognizes and reads OCR-A, OCR-B, or SEMI font character strings.

Adding a Component to the Toolbox

To modify the content of a HexSight application, you must add a HexSight Application control to a Windows form. This enables you to access the Process Manager and the Explorer interfaces in design mode or in runtime.

You should also add at least one HexSight Display control to the form. This will allow you to visualize results in runtime mode.

To add a control to the toolbox:

1. From the toolbox context menu, select: Choose Items.
2. Select the COM Components tab and scroll down to find HSApplication Control. Enable its check box and click OK. The HexSight Application Control appears as a lens shutter icon in the toolbox.

Note: A variable cannot be declared as a given HexSight control type if it is declared as a component. Nevertheless, any object created by pasting a control (selected as a component) on a Visual form can be accessed programmatically using the name of the object.

Note:

The following HexSight control types should be used as components:

- Application control
- Display control

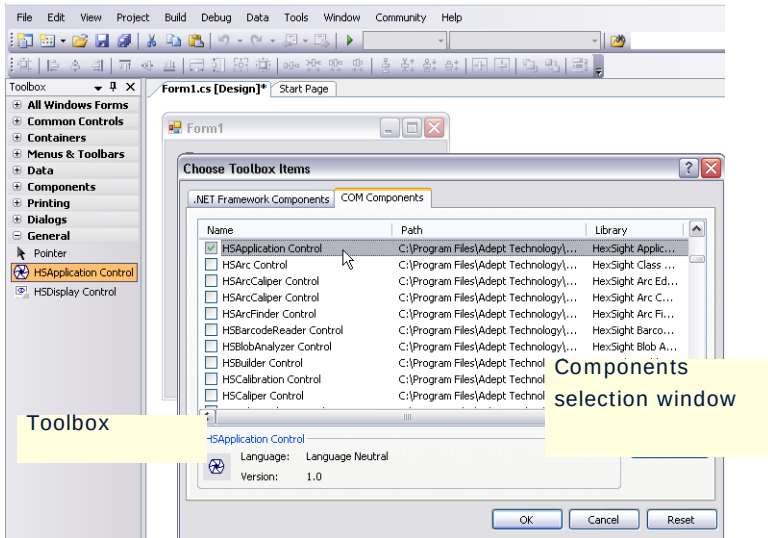


Figure 4 Addition of a Component to the Visual Basic Toolbox

Referencing a Control

To create HexSight objects programmatically, all HexSight control types used in the user's Visual Basic program must be referenced. This is accomplished the following way:

1. From the Microsoft Visual Studio menu bar, select Project > Add Reference...
2. Select the COM tab, and enable the required HexSight controls.
3. Click OK. The controls are added to project's references and can now be accessed programmatically.

Note: A HexSight control cannot be declared as a component and as a reference at the same time. be declared as a component and as a reference at the same time.

Command Buttons

All properties windows for HexSight processes contain the same set of command buttons. The purpose of these buttons is described below:

- **Apply**: Registers changes made to the parameters.
- **Reset**: Resets the parameters to the configuration obtained after the last **Apply**, **OK** or **Execute**.
- **Default**: Resets all parameters to their original configuration when properties window was first opened.
- **Execute**: Saves the latest changes of parameters and executes the process.
- **OK**: Saves the latest modifications made to the parameters of the process and closes the process interface window.
- **Cancel**: Cancels all modifications made to the parameters of the process since the last **Apply**, **OK** or **Execute** and closes the properties window.

Graphical Interface Elements

Most of graphical interface elements that are often referred to in the User Guide are illustrated in [Figure 5](#).

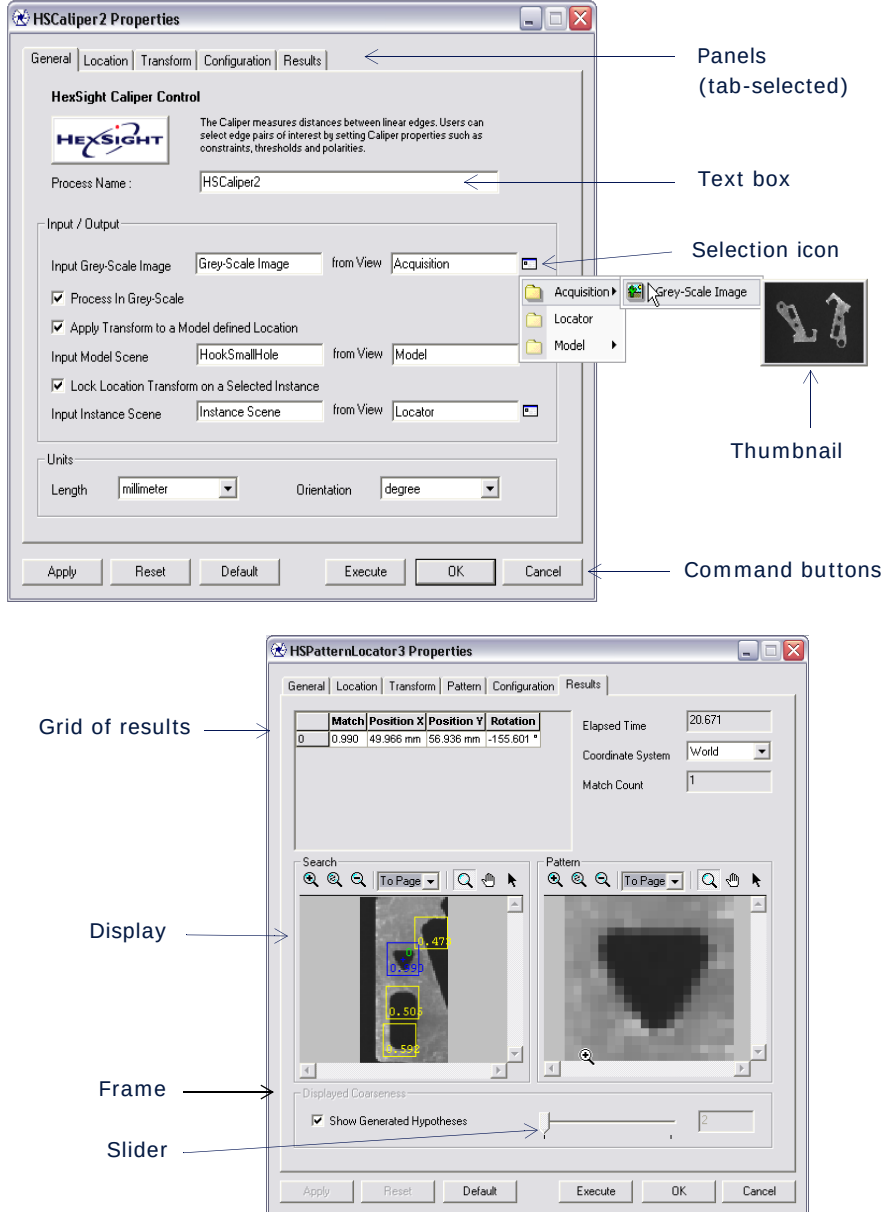


Figure 5 Common Element of HexSight Interfaces

Getting Help

This section explains how to access the HexSight online help as well as HexSight customer support service.

Online Help

The HexSight toolset is provided with online help. HexSight Help contains both the Reference Manual and the User Guide.

There are two ways to access this system:

- From the Application Control interface, click the **Help** command button. See the **Process Manager** section in the **Application Control Interface** chapter.
- In any property window of a HexSight tool, place the cursor on an object and press **F1**.

HexSight Technical Support

We provide technical support for our customers via our web page as well as by telephone.

E-mail Queries

Send your technical questions by e-mail, directly to our technical support team at techsupport@hexsight.com. Your e-mail queries will be directed to the appropriately qualified personnel and promptly answered.

Telephone

You can contact the HexSight technical support between 9:00 A.M. and 5:00 P.M. PST at +1 604 636 1011.

Project Examples Provided with HexSight

The HexSight package provides complete examples of machine vision applications. These examples show how to implement specific features such as automated calibration and model teaching.

Examples of application projects are provided for the Microsoft Visual Studio 2003 .NET (or higher) development environment. The target application platform can be set using the Configuration Manager dialog box. It can be set to x86 (Win32), x64, or Any CPU. If Any CPU is chosen, which can only be used with .NET languages, the same executable will run on either x86 or x64 platforms.

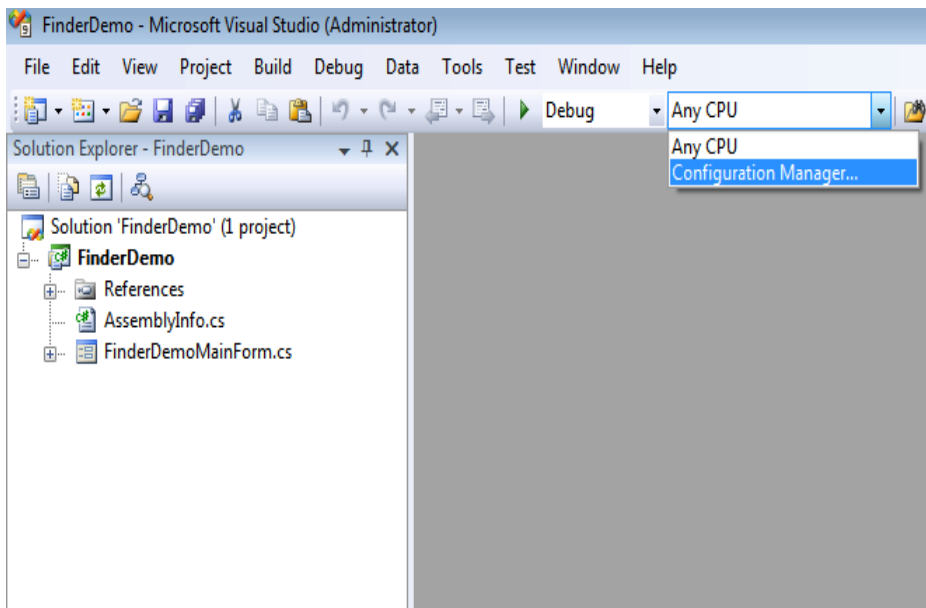


Figure 6 Selecting target platform

The target application platform for all of HexSight's C# examples is set to Any CPU by default. The C++ LocatorDemo example includes configurations that target both the x64 and x86 platforms. However, any example can be

compiled as an x64 version simply by adding a configuration that targets the x64 platform in the Configuration Manager dialog box, using a copy of the x86 configuration as the starting point (see figure).

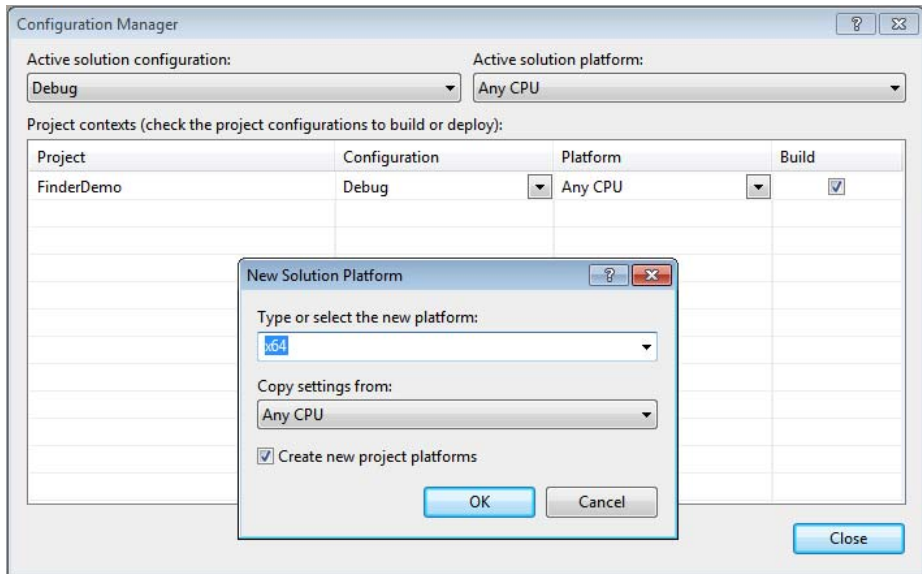


Figure 7 New configuration in Configuration Manager

Note: The Visual Studio x64 cross compiler must be installed in order to add an x64 configuration.

Board Location

This application identifies and locates a printed circuit board (PCB). The application demonstrates how to create a model in which one big feature is used to locate the object and a second smaller feature is required to correctly identify the orientation of the object. A **Sampling** tool samples the image from a model-based region on the located boards. This sampled image is used as an input to a second **Locator** tool which then checks for the presence of a small label printed on the PCB.

Board Location Model-Based

This application is similar to the Board Location example. However, this example uses a Model-Based Locator, instead of a Sampling tool, to identify subfeatures on the object.

Bracket Inspection

This application identifies, locates and inspects four different models of flat brackets. The Blob Analyzer tool finds the center of mass of each hole on the bracket. Parts are rejected if the position of any of its holes are not within tolerance ranges.

Calibration Demo

This application shows how to use the `HSCalibrationXYScaleInterface`, `HSCalibrationPerspectiveInterface`, and `HSCalibrationDistortionInterface` objects to implement automated calibration programmatically, without showing the interactive user interface.

Color Locator Demo

This demo shows the potential of the Locator in regards with Color detection. Not only can the Locator detect edges that may be invisible when converted to greyscale, but it can also differentiate the instance based on the color of the model.

Color Target Inspection

This application demonstrates the use of the `ColorFinder` tool to inspect the color of specific areas of the image. By adding filters corresponding to particular colors, you can easily say if a colored area is within the acceptable tolerance defined.

Contour Detection

This application shows how to access the calibrated and uncalibrated contour data extracted from an image by the Locator. The application also shows how

to use properties of the extracted contour data, such as perimeter and area, to manually detect circular shapes in the image. This can be used, for example, to detect dots during a custom calibration procedure using a non-standard target.

Contour Draw Demo

This application shows how to manually draw the detected contours (model and instance) into a standard Visual Basic or Visual C++ graphics object by directly accessing the contour data generated by the Locator through the HsScene object.

Image Processing

This application shows how the Image Processing tool can be used to apply operations to an image, such as noise filtering, before the image is output to the Locator tool.

Locator Demo

This demo provides a ready-built application, useful for testing HexSight with your own images. This demo also contains configurations to illustrate Model Optimization and the Shading consistency feature.

Finder Demo

This demo provides an example of the use of HexSight Finder Tools: Arc Finder, Line Finder, Point Finder and Builder.

Model Demo

This application shows how to use the HsModelEditorInterface object to implement automated model teaching programmatically, without showing the interactive user interface.

Model Editor Demo

This demo shows how to implement a custom model editor.

Pad Inspection

This application identifies and locates brake pads on the work surface and performs two measurements on each located pad. This application provides an example of how the Caliper tool can be used to perform linear measurements.

Part Sort

This application provides the type, position and orientation of 4 types of parts that are randomly placed on the work surface. Multiple parts can be identified and located at the same time. These parts do not have to be of the same type. In addition to multi-model capabilities, this application also illustrates how to configure the Locator tool for robustness to occlusions

Pulley Location

This application identifies and locates a pulley on the work surface. The Locator identifies the four possible rotational symmetries. An Edge Locator tool locates the pulley's keyway to identify the correct orientation. An Arc Edge Locator is used to measure the angular position of the four spokes of the pulley, with respect to its keyway.

Regulator Inspection

This application sorts and inspects electronic parts. The Pattern Locator tool is used to disambiguate the two models of T0220 regulators that are used. Distances between the legs of each regulator are measured using the Edge Locator tool and part is rejected if the distance between two legs is not within tolerance ranges.

OEMLocatorGeneric

This application shows typical Locator integration for OEM projects using ActiveX layer.

OEMLocatorDLL

This application shows typical Locator integration for OEM projects using DLL layer only. Since DLL layer do not expose any Graphical User Interface, this example also shows how to display HexSight images and Locator results. It also includes an implementation of a custom model edition window using programmatic calls to the DLL interface.

OEMLocator.NET

This application shows typical Locator integration for OEM projects using only the .NET assembly layer.

Demonstration Mode

The HexSight Demonstration Mode is provided for the purpose of evaluating our software. This mode is automatically enabled when you run HexSight without the hardware protection key that is provided with the purchase of a HexSight software package.

This demonstration mode has some features removed, yet will allow you to evaluate the functionality and power of the HexSight vision software.

Opening and Running in Demo Mode

When you first open HexSight in demo mode, a pop-up window appears, advising you that the hardware protection key was not found and that the software will run in the demonstration mode.

The application then pauses for approximately 10 seconds. After this delay, the application resumes once you click **OK**.

This reminder pop-up window will appear again at random intervals during execution; simply click OK and continue your evaluation.

Demo Mode Limitations

In the Demo Mode, the application development capabilities of the HexSight environment are deactivated.

- You can load and run the examples that are installed from the HexSight CD.
- You cannot operate in design mode i.e. you cannot build an application by selecting and instantiating tools.
- You cannot save a configuration file. You can however open an existing application.
- Since you cannot run in design mode you cannot fully do the tutorial. You can however open the Tutorial files that are installed from the HexSight CD.

Using the Demo Mode with the Tutorial

Even though you cannot use the demo mode to build an application as explained and illustrated through the tutorials, you can run any of the applications provided on the HexSight CD, either in the Examples or the Tutorial folders.

To best familiarize yourself with the functionalities of HexSight, we recommend that you start out with the tutorial application, as follows:

- Open the final application provided for the tutorial in Visual Studio:
HexSight / Tutorial / VisualC++ / Hook Inspection / Part6 / Hook Inspection.sln
- Run the application then, in the Hook Inspection interface, click the **Inspect** button to launch a first execution. See Figure 8.

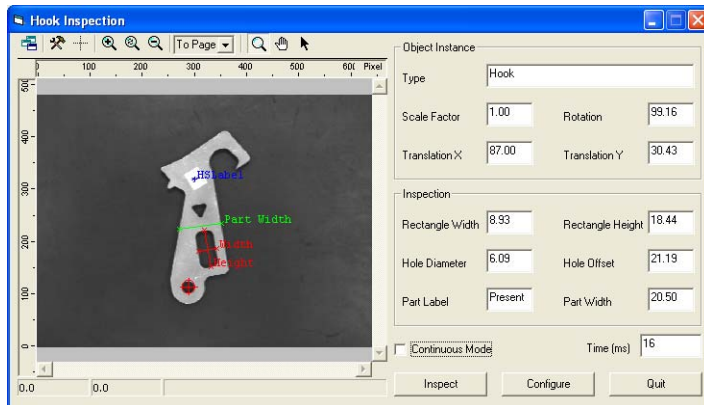


Figure 8 Hook Inspection Interface

Color Support and Functionality in HexSight

HexSight provides support for acquiring and using color images. Additionally, the optional color license adds color capabilities to various HexSight tools and processes.

This section provides an overview of color support in HexSight, described under the following topics:

[Color Image Acquisition](#)

[Color Calibration](#)

[Color Locator](#)

[Color Processing with Inspection Tools](#)

[Color Finder Tool](#)

Color Image Acquisition

The Acquisition Device tool accepts and outputs color images provided by supported color cameras. Moreover, HexSight provides an optional color calibration that can ensure that the Acquisition Device tool provides accurate color images.

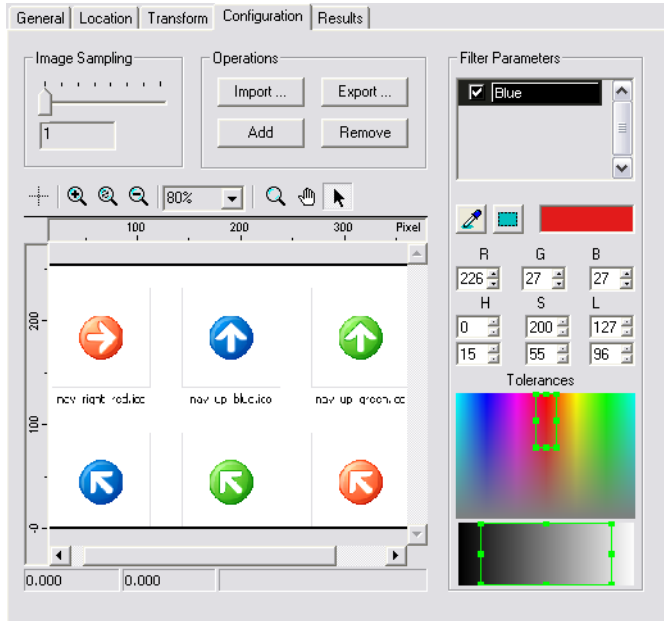


Figure 9 HexSight Application with Color

Color Calibration

In applications where accurate color is required, color calibration ensures that the camera will provide images that contain accurate color information.

HexSight includes a Color Calibration dialog that allows you to perform a color calibration, using a standard GretagMacbeth ColorChecker target.

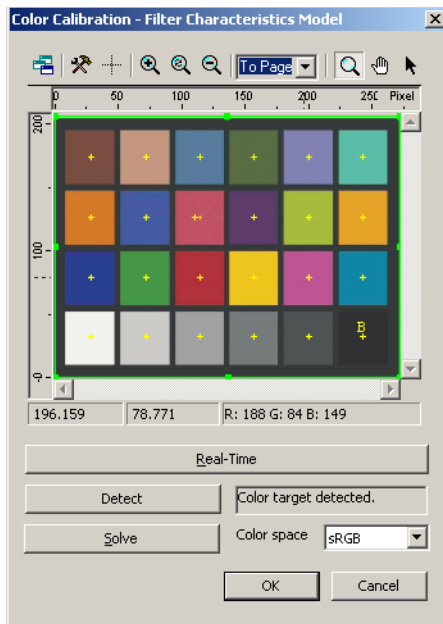
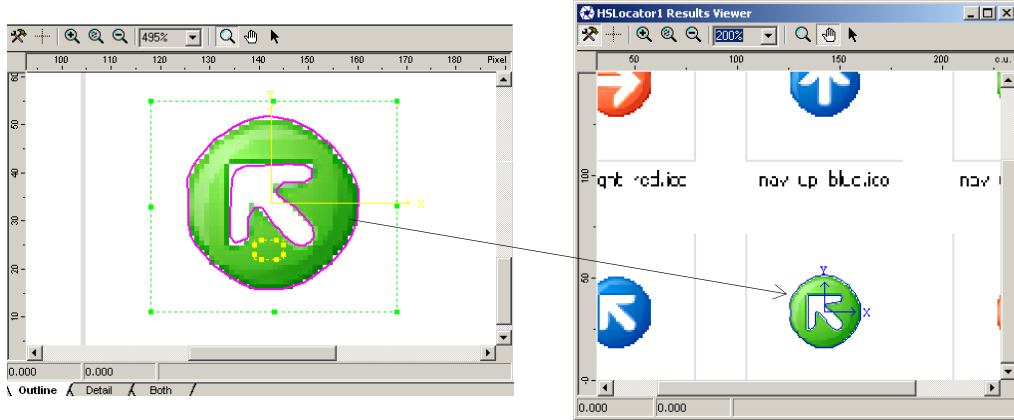


Figure 10 HexSight Color Calibration Dialog

Color Locator

The Locator can be configured to differentiate between objects based on their color. In the model-building process, a custom color shading area can be defined for each object. This shading area allows the Locator to use color information when locating objects.



Custom shading area in model defines the color of the object

Figure 11 Using Custom Shading Area to Find Parts According to Color

Configuring a Color Locator Tool

The Locator can be configured to find and differentiate objects based on their color information. This is useful when locating objects that are similar in shape but of different colors. The color that distinguishes an object is defined when creating a model for the object: a custom shading area must be defined in the Model. The Locator search process must also be configured to find the objects based on their shading, through the Shading Consistency mode of the Instance Ordering parameter.

Basic Steps for Configuring a Color Locator

1. Create a Model.
2. Use the Model editor to define a custom shading area, which specifies the color of the object.
3. In the **Advanced Parameters**
 - On the General tab, verify that processing format is set to native format (Process In Grey-Scale is unchecked).

- On the Search tab, set Instance Ordering to Shading Consistency.

Configuring Color Shading Area in a Model

To enable the Locator to differentiate between object based on their color, a custom color shading area can be defined for each object in the model-edition process. This is done by defining an area in the model called a “custom shading area”. The color information in this area is stored as part of the model, and enables the Locator to use color information when locating objects.

To configure a color for a specific object:

1. Follow the normal steps to create a model.
2. On the Model tab, click Edit to access the model editor parameters.
3. On the Model Contents, Parameters tab, enable (check) **Custom Shading Area**.
4. Drag and resize the custom shading area (Yellow) bounding box.

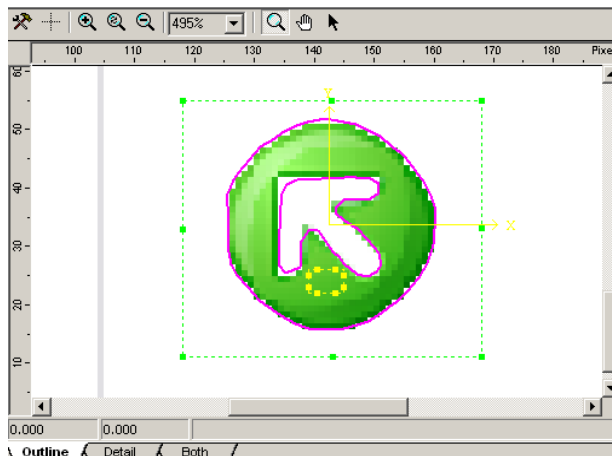


Figure 12 Setting the Custom Shading Area

5. The bounding box should cover an area that is on the typical, distinctive color for the object. For example, avoid including shadows or light reflections in this area.

6. Click Apply to apply the settings to the Model and then click OK to exit.

To configure the Locator to search for models based on their color:

1. Expand the **Advanced Parameters** section of the Locator interface.
2. On the General tab, make sure that **Process In Grey-Scale** is unchecked (not selected). See [Processing Format](#) for more details.
3. On the **Search** tab, set **InstanceOrdering** to **Shading Consistency**. See [Shading Consistency](#) for more details.

Configuring Advanced Parameters for Color

The Locator can be configured to differentiate between object based on their color. In the model-building process, a custom color shading area can be defined for each object. This shading area allows the Locator to use color information when locating objects.

Processing Format

The processing format defines the format applied to process images provided by the camera. To process color images, the Process In Grey-Scale option must be unchecked (not selected). This ensures that the Locator is processing color images in their native color format, not in grey-scale mode.

Shading Consistency

Shading Consistency orders instances according to the custom shading area created in the model. If no Custom Shading Area is defined in the model, the Locator uses the entire model area for shading analysis.

To enable color processing of models (i.e. models are recognized on the basis of their color) you must set this parameter to Shading Consistency.

Color Processing with Inspection Tools

Most inspection and finder tools can process images on the basis of color. An example of this is the detection of edges based on color values.

The polarity of the color edges cannot be determined, due to their nature, and they are always considered to be a rising edge, or a dark to light edge, in the tools. It is recommended to use the Dark to Light polarity settings when using color images. The Either setting is also an acceptable setting for color edges detection.

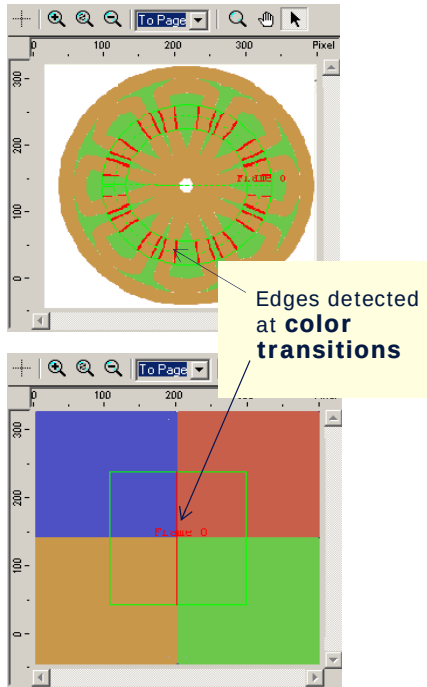
Because color processing can increase execution time, when color processing is not required, you can enable grey-scale processing (by selecting Process In Grey-Scale) to improve execution time.

Note:Color processing tool requires a Color License.

The following inspection tools support color processing:

- **Edge Tools:** The Edge Locator, Arc Edge Locator, Caliper, and Arc Caliper tools use color information to extract edges.
- **Finder Tools:** The Arc Finder, Line Finder, and Point Finder tools use color information to extract edges for finding geometric entities (arc, lines, points.)
- **Blob Analyzer:** The Blob Analyzer can use Hue, Saturation, and Luminance values to define image segmentation.

Color image



Grey-scale version of the image

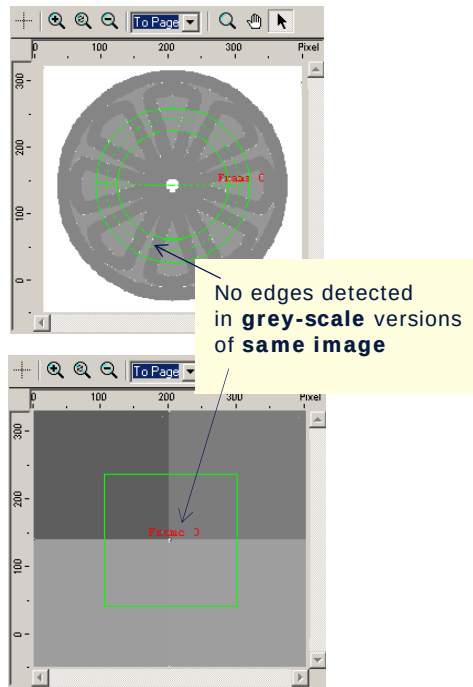


Figure 13 Color Processing with Finder and Edge Tools

Color Finder Tool

The Color Finder tool allows you to easily define filters to extract and analyze color areas in an input image.

The Color Finder tool can identify the presence or absence of defined colors, and analyze the predominance of colors in images. For example, to inspect and differentiate similar objects of different colors.

For more information on this tool, see [Color Finder](#).

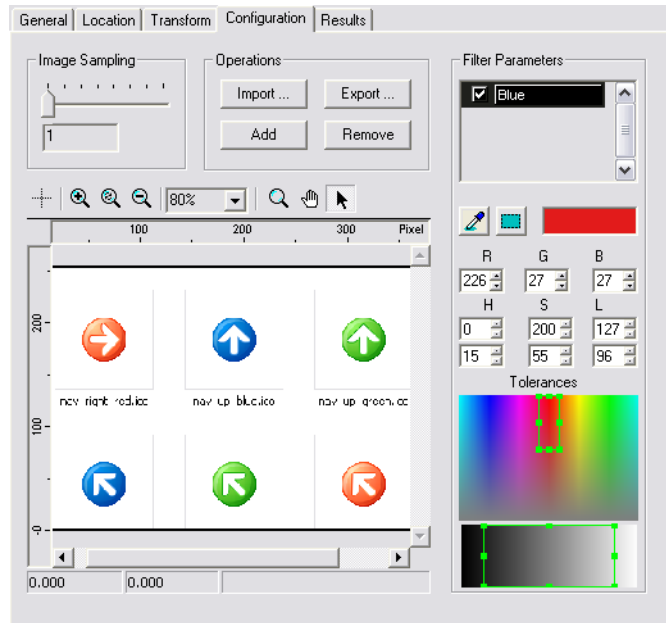


Figure 14 Example of the Color Finder Tool interface

VB .NET Tutorials

The tutorials in this chapter will guide you through the development of a HexSight application in Visual Basic .NET (VB .NET). As you follow the steps for each tutorial you will build an object location application to which you will add and configure a full range of inspection tools.

Note: This tutorial assumes you have a basic beginner's knowledge of VB .NET.

Chapter Outline

[VB .NET Tutorial 1: Locator](#)

[VB .NET Tutorial 2: Display](#)

[VB .NET Tutorial 3: Caliper](#)

[VB .NET Tutorial 4: Blob Analyzer](#)

[VB .NET Tutorial 5: Pattern Locator](#)

[VB .NET Tutorial 6: Edge Locator](#)

[VB .NET Tutorial 7: Calibration](#)

VB .NET Tutorial 1: Locator

This first tutorial shows you how to build a basic HexSight application that will locate a model-defined object at whatever angle and displacement it appears.

Creating the Main Form

In this section, you will create the Main Form of the application, add basic lines of code to interact with the interface, and add a HexSight Application Control, which you will edit to build the application.

Constructing the Main Form Interface

Here you will build the interface that will allow you to interact with your application and visualize results.

1. Start Microsoft Visual Studio. Open a new **Visual Basic Windows Application** project called **HookInspection**. From **Explorer** View, change the name of the form to *HookInspectionMainform.vb*.
2. Make the toolbox and the properties window visible, using the View menu.
3. Set the form *Text* property to **Hook Inspection**. Set the *FormBorderStyle* property to **FixedSingle** and the *MaximizeBox* property to **False**. Set the *Width* to approximately 355 and the *Height* to 275.
4. On the form, add a group box named **InstanceFrame**. Add 5 text boxes respectively named **TypeText**, **ScaleText**, **RotationText**, **TranslationXText** and **TranslationYText**. Also set the frame caption, the text box dimensions and add labels. The text property for the text boxes should be empty. Your form should look like the form in [Figure 15](#).

The image shows a Windows form titled "Hook Inspection". Inside the form, there is a group box labeled "Object Instance". Within this group box, there are five text input fields arranged in two rows: "Type" (top left), "Scale Factor" (middle left), "Rotation" (middle right), "Translation X" (bottom left), and "Translation Y" (bottom right).

Figure 15 VB .NET Form with Object Pose Information

5. At the bottom of the form, add 3 command buttons named **InspectButton**, **ConfigureButton** and **QuitButton**. Set the caption of these buttons respectively to: **Inspect**, **Configure** and **Quit**.

The image shows the same "Hook Inspection" form as in Figure 15, but with additional controls. Below the "Object Instance" group box, there is a checkbox labeled "Continuous Mode" and a text box labeled "Time (ms)". At the very bottom of the form, there are three command buttons labeled "Inspect", "Configure", and "Quit" from left to right.

Figure 16 VB .NET Form with Command Buttons

6. Add a check box named **ContinuousCheck**, and set its caption to **Continuous Mode**. Finally, add a text box named **TimeText** and an appropriate label. Place everything to resemble [Figure 16](#).
7. The interface is almost complete. Save your work.

Adding a HexSight Application Control

You will now add a new component to the Microsoft Visual Studio toolbox: the HexSight Application Control, which is the core of each HexSight application.

1. From the toolbox context menu, select: *Choose Items...* Move to the COM Components tab and scroll down to find **HSApplication Control**. Enable its check box and click **OK**. The HexSight Application Control appears as a lens shutter icon in the toolbox.
2. Click the **HS Application Control** icon and drag a small box on the form with your cursor. The default name of this control is `AxHSApplication1`. In the Properties window change its name to `ApplicationControl`. The interface of the project is now complete. Save your work. You will configure the HexSight Application Control in subsequent tutorials.

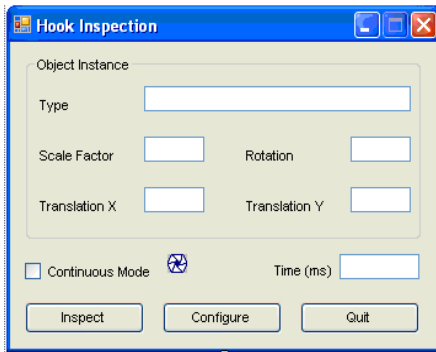


Figure 17 VB .NET Form with a HexSight Application Control

Adding Basic Code to the Main Form

You need to add basic code to be able to interact with your application. As you will add new features throughout the other tutorial sections, you will also add lines to this code.

1. Double-click **Quit** and insert the line of code shown in bold:

```
Private Sub QuitButton_Click(...) Handles QuitButton.Click
    Application.Exit()
```

End Sub

This will terminate the application when the **Quit** button is clicked.

2. Double-click **Configure** and add the line of code shown in bold:

```
Private Sub ConfigureButton_Click(...) Handles
    ConfigureButton.Click
        ApplicationControl.ShowInterface(False)
End Sub
```

This will show the HexSight Application Control's interface in a non-modal window when the **Configure** button is clicked.

3. From the Microsoft Visual Studio menu bar, select *Project>References...* Scroll down to find **HexSight Process Manager Control**, enable its check box and click **OK**.

The HexSight Process Manager is now added to the project's references and can thus be accessed programmatically. Note that in order to work properly, the Process Manager should only be set through the *Project>References...* menu and not through the *Projects>Components...* menu.

4. Double-click **Inspect** and add the lines of code shown in bold:

```
Private Sub InspectButton_Click(...) Handles
    InspectButton.Click
        Dim lManager As HSPROCESSMANAGERLib.HSProcessManager
        InspectButton.Enabled = False
        lManager = ApplicationControl.ProcessManager
        Do
            Application.DoEvents()
        Loop While ContinuousCheck.Checked
        InspectButton.Enabled = True
End Sub
```

This is the main loop of the inspection application. It loops until **Continuous Mode** is disabled in the interface.

5. Add **Load** and **FormClosed** event handlers as shown below:

```
Private Sub HookInspectionMainform_FormClosed(...)
    Handles Me.FormClosed

    QuitButton_Click(sender, e)

End Sub

Private Sub HookInspectionMainform_Load(...) Handles Me.Load

End Sub
```

You will later place initialization code for the empty *Form_Load* Sub. Clicking the **Close** button calls the *Form_FormClosed* Sub and ends the application.

You have completed the code needed to use your application. Run and test the application. When you are sure that everything works fine, save your work and go on to the next section.

Adding an Acquisition Device Tool

In this section, you will learn how to instantiate a process in the HexSight Application Control. You will also learn how to configure an Acquisition Device tool. This tool will acquire an image from an existing database file to simulate real image acquisition.

Instantiating the Acquisition Device

In order to build a vision application, HexSight processes are instantiated using the Application Control's Process Manager.

1. On the main form, right-click the **HexSight Application Control**. Select **ActiveX -Edit** from the menu. The HexSight Application Control's interface opens and the active tab is the Process Manager. In the **Available Processes** list of the Process Manager, select

HSAcquisitionDevice and click **Add Process**. The Acquisition Device Properties window appears.

At the end of this tutorial, the HexSight Application Control should look like the illustration in [Figure 18](#).

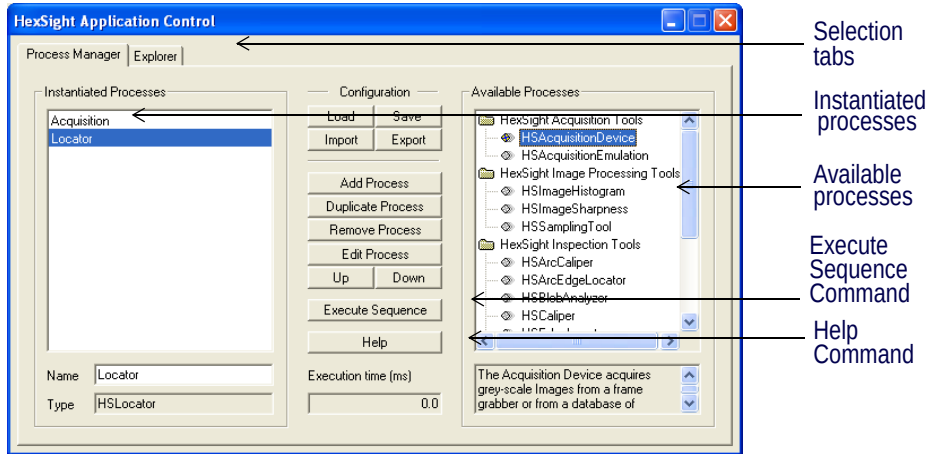


Figure 18 HexSight Application Control — Process Manager

Configuring the Acquisition Device

You will use an existing database of images to simulate a real image acquisition. The database used for this tutorial is constituted of images of the same objects, rotated at different angles.

1. In the Acquisition Device **General** panel, enter *Acquisition* as the **Process Name** (instead of HSAcquisitionDevice0).
2. In the **Configuration** panel, select the **Input** tab. Select **File Emulation** from the list.
3. Click **Load** then load the following file:
InstallationPath\Tutorial\VBNET\HookInspection\Data\Hook.hdb.
4. Disable all the images of the emulation database except the first one: *Image0*. This ensures that this image will be used to create the model later in this tutorial.

5. Leave the default settings for the Calibration and Output sub-panels.
Click **OK** to return to the Process Manager of the HexSight Application Control.

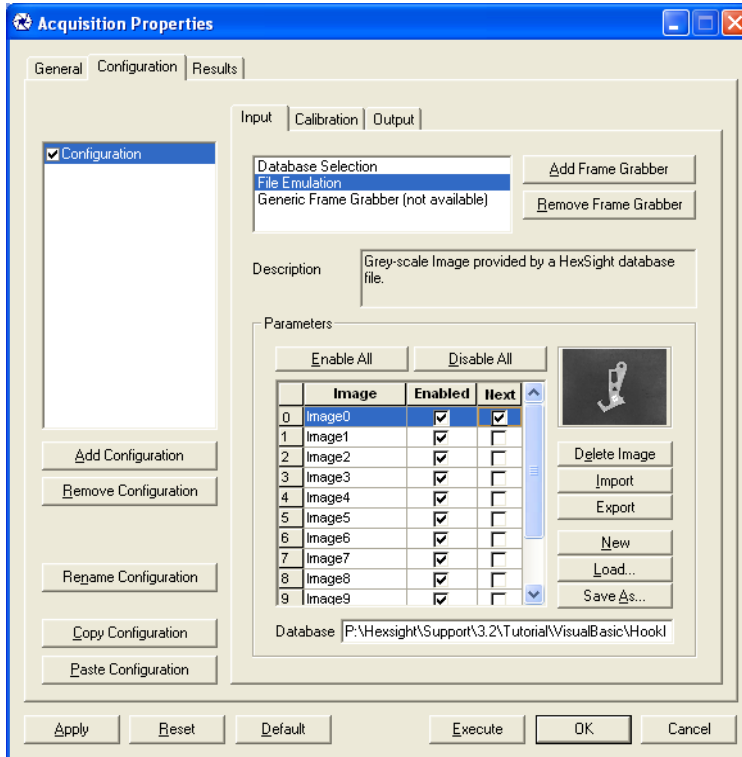


Figure 19 Acquisition Device — Configuration Panel

To sequentially execute the processes instantiated in the Process Manager, click **Execute Sequence**. At this stage, each time you click **Execute Sequence**, the Acquisition Device enables and outputs a new emulation image to the application runtime database.

Viewing Scenes and Images

The Explorer panel in the HexSight Application Control shows the application's runtime database as a hierarchical list.

1. Select the **Explorer** panel in the **HexSight Application Control**.
2. Click on + symbols to expand **RunTime** and **Acquisition**.
3. Click **Grey-Scale Image** to display the current image output by the acquisition process.

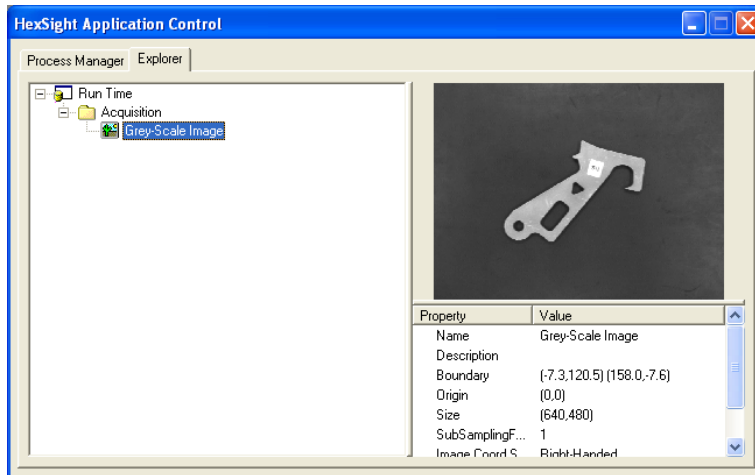


Figure 20 HexSight Application Control — Explorer

Adding a Locator Tool

The Locator locates model-defined objects within the input Image, using the geometry of object contours.

For this tutorial you will use only the basic Locator properties. Therefore, the **Advanced Parameters** check box should be left unchecked.

Instantiating the Locator

Here you add a Locator process the same way you added the Acquisition Device.

1. In the HexSight Application Control Process Manager, select **HSLocator** from the **HexSight Locator Tools** folder. Click **Add Process**.

2. In the properties window of the newly added Locator process, select the **General** tab.
3. Enter the name *Locator* instead of HSLocator1.
4. In the **Input** frame, click on the selection icon at right to select **Image** from **Acquisition** view.

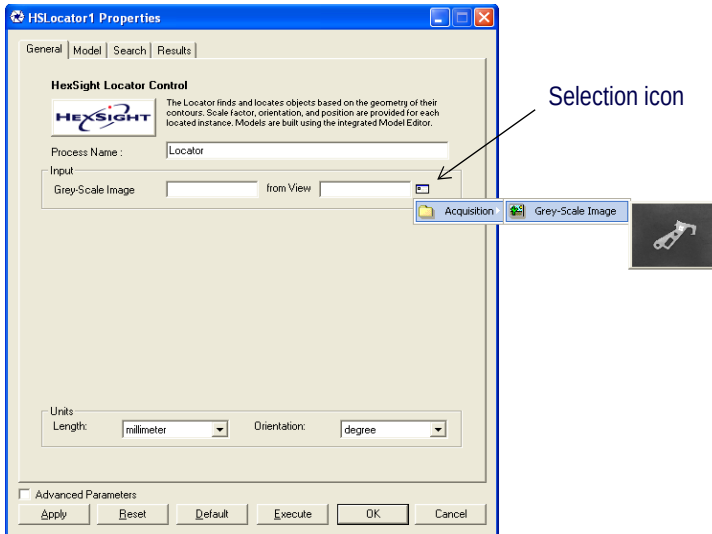


Figure 21 Database Selection Icon

5. Click **Execute**.

Creating a Model

In this section, you will learn how to create a Model of an object using the Model Editor. A Model is a reference or “ideal” representation of an object’s contours that will be used by the Locator’s search process to recognize and locate instances of the object. Models can be stored in database files for later use.

1. Select the **Model** tab, then click **Add** in the **Models** frame. The **Create New Model** prompt appears.

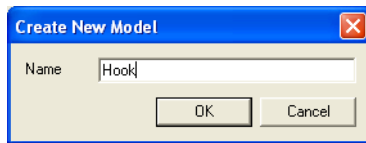


Figure 22 Create New Model Dialog

2. Name the Model *Hook*.
3. Click **OK** to open the Model Editor.

Using the Model Editor

The Model Editor is used to create new models and to modify existing models. Once you have created a model of the hook for this application, you will save it for later use with the rest of the tutorial.

Creating the Source Contours and Selecting Model Features

The Locator uses two coarseness levels to recognize and locate the object: the **Outline Level** and the **Detail Level**. The outline coarseness level is used to rapidly identify potential instances of the object. The detail coarseness level is used to confirm that it is the correct object and to refine its location within the image. Source contours will be created at the selected **Outline Level** and **Detail Level**. Features will be selected from these source contours to create the Model.

When the **Automatic Levels** check box is enabled, the two contour detection parameters (**Outline Level** and **Detail Level**) are automatically optimized by analyzing the Model's bounding area. Automatic contour detection parameters should usually be used when building a Model.

Contrast Threshold is used to set the sensitivity to contrast when generating the source contours. **Contrast Threshold** can be set manually, by selecting **Fixed Value** with the combo box, or automatically by selecting **Adaptive Low Sensitivity**, **Adaptive Normal Sensitivity** or **Adaptive High Sensitivity**.

Once the source contours are detected, features must be selected to build the Model. Features can be selected manually, by setting the **Feature Selection**

slider to **none**, or automatically by selecting **less**, **normal**, **more** or **all**. Setting the slider to **normal** will select the optimal amount of features at both the Outline and Detail Levels. Increasing the setting will add more features, which may slow the search process. Decreasing the setting may cause recognition problems because the Model may lack essential features.

In this tutorial, we will use the default contour detection and feature selection parameters to build the model, as shown in [Figure 23](#).

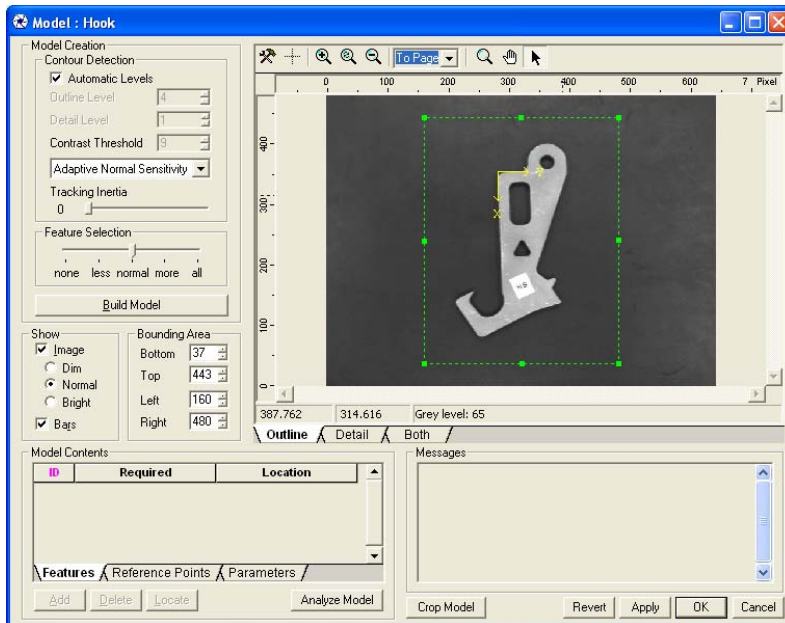


Figure 23 Model Editor Interface

1. In the **Contour Detection** frame, leave the **Automatic Levels** check box enabled, and **Contrast Threshold** combo box to **Adaptive Normal Sensitivity**.
2. In the **Feature Selection** frame, leave the **slider** in the default **normal** position.
3. In the display, drag and resize the green dotted rectangle (the Model's bounding box) to enclose the entire object.
4. Click **Build Model**.

The new Model is now created. Contour detection parameters have been optimized and features were selected at both the Outline and Detail levels. To verify that no problem was encountered while building a model, check the **Messages** text box.

You can view and edit the Model in the display. Select the **Outline** tab at the bottom of the display to view the Model at the Outline Level. Outline features are shown as bold magenta lines. Unused contours are shown as thin blue lines. Select the **Detail** tab at the bottom of the display to view Detail Level features. Detail features are shown as bold green lines. As with the Outline Level, unused contours are shown as thin blue lines. The **Both** tab displays both Outline and Detail Levels as superimposed.

Editing Models

You can now use the Model to locate instances of the object. We will however edit it to show how the Model Editor can be used to customize a model for a specific application.

Removing Features from a Model

The application we are building in this tutorial will be used to inspect features on the hook with respect to its outer contour. The steps will be to locate the part using its outer contour and apply tools to inspect the rounded rectangle, the hole and the label. These features need to be inspected because they might be missing on actual parts, be the wrong size, or be mislocated with respect to the outer contour. Because these features are unstable, they should be removed from the model to ensure the best possible accuracy for locating the outer contour of the object.

The first step will be to remove unwanted features at the Outline Level.

1. Click the **Outline** tab at the bottom of the display. It will enable you to edit the Model at the Outline Level. The grid in the **Model Contents** frame displays the **Features** sub-panel for the Outline level of the Model.
2. Click the **Features** tab in the **Model Contents** frame to activate the features grid.

3. Select the rounded rectangle feature by either clicking it in the display or by selecting it in the **Features** grid.
4. Click **Delete** to remove this feature from the Model.

The next step will be to remove unwanted features at the Detail Level.

5. Click the **Detail** tab at the bottom of the display. It will enable you to edit the Model at the Detail Level. The grid in the **Model Contents** frame displays the **Features** sub-panel for the Detail level of the Model.
6. Select all the features, except the outer contour, by clicking them while holding the **Ctrl** key either in the display or in the **Features** grid.
7. Click **Delete** to remove these features from the Model.

The unstable features are now removed from the model. To view the outline and Detail Levels superimposed, click the **Both** tab at the bottom of the display. In this viewing mode, the model is not editable.

Setting the Object Coordinate System for a Model

The part inspection process requires that the measurements of the features be expressed with respect to the outer contour of the part. To fix a reference frame to express these inspection results, you will define an Object coordinate system for the part.

1. Click the **Outline** or the **Detail** tab at the bottom of the display. It will enable you to edit the Object coordinate system.
2. Click the **Coordinate System** tab in the **Model Contents** frame to activate the coordinate system panel.

3. Use the mouse to position the coordinate system marker (yellow axes) over the object, as shown in [Figure 24](#).

To rotate the object coordinate system, click on the arrow of the X or the Y-axis and drag with the mouse.

To move the coordinate system, click the intersection of the X- and Y-axes and drag with the mouse.

You can also drag the arrow of the axes to stretch the marker. Doing so has no effect on the Object coordinate system itself, but can help you align the marker over long features. Fine-tune the final position with the text boxes in the **Coordinate System** panel.

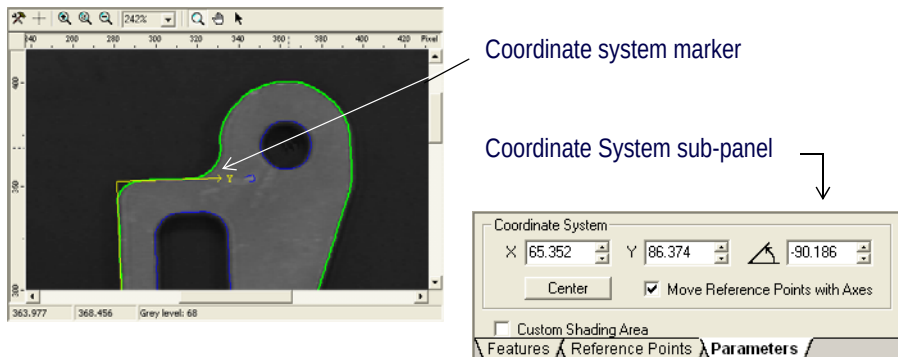


Figure 24 Adjusting the Position and Orientation of the Model Coordinate System

Adding Reference Points to a Model

For some applications, it is sometimes useful to have the position of specific points on the Model transposed in the World coordinate system for each object instance located. To do so without having to compute coordinate transforms manually, reference points can be defined on a Model. In this application, we will add a reference point above the circular hole.

1. Click the **Outline** or the **Detail** tab at the bottom of the display. It will enable you to edit the reference points.
2. Click the **Reference Points** tab in the **Model Contents** frame to activate the reference points panel.

3. Click **Add** to create a reference point in the display.
4. Click and drag the target marker to position the reference point above the circular hole in the part. Zoom in for more precision. Fine-tune the position using the text boxes in the **Reference Points** panel.

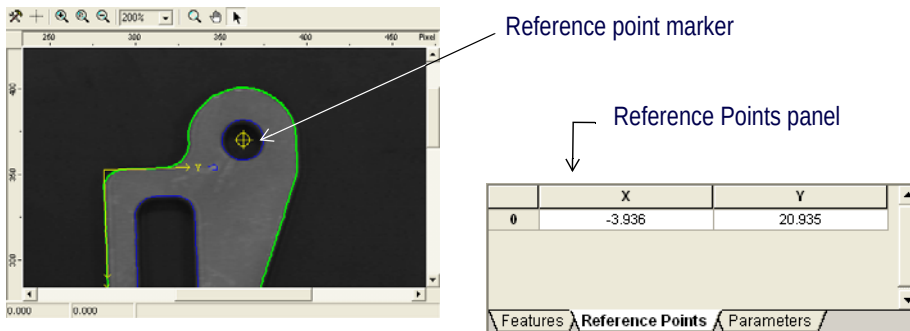


Figure 25 Adding a Reference Point with the Model Editor

This reference point will not be used to locate the part, only contour features are used for locating objects. However, for each instance of the object that will be located, the reference point's position will be returned with respect to the scale, rotation and translation of this instance. You can add as many reference points as you need on a Model.

Manually Adding Features to a Model

Sometimes, it may be useful to manually add a feature to a Model. In Model Editor, you can select features from the unused contours at the Outline Level as well as the Detail Level. To add a feature at the Outline Level, follow the procedure below.

1. Select the **Outline** tab at the bottom of the display to edit the Model at the Outline Level.
2. Click the **Features** tab in the **Model Contents** frame to activate the features grid.
3. Disable the **Image** check box in the **Show** frame to show only the contours. This will make the blue unused contours more visible.

4. In the display, click one of the unused contours, the rounded rectangle for example. The contour turns light blue.
5. Click a starting point on the contour. Hold the **Ctrl** key and click another point on the contour. The selected feature appears in red.
6. To invert your selection, hold the **Shift** key while clicking on the contour. Click **Add** in the **Model Contents** frame. The new feature appears in the features grid.
7. Select the newly added feature in the grid and click **Delete** to remove it from the Model.
8. To add the entire rounded rectangle as a feature, double-click it in the display so the whole contour is selected. Click **Add** to add this feature to the model.
9. Select the newly added feature in the grid and click **Delete** to remove it from the Model, since it is considered unstable for this application.

Analyzing a Model

When a Model has been edited manually, it is important to verify that it will work correctly with the Locator. To do so, click **Analyze Model** in the **Model Contents** frame. Any problem will be displayed in the **Messages** frame.

Saving a Model Database

When a Model is completed, it must be saved in a model database, otherwise it will be lost when you exit the application. To save the Model, follow the procedure below.

1. Click **OK** at the bottom right of the Model Editor. This will transfer the newly created Model to the active model database of the Locator.
2. In the Locator's properties window, select the **Model** tab.
3. Click **Save As...** in the **Database** frame. Save your Model as *HookModel* in your working directory. You will find it convenient to store Models close to other HexSight files used in a same application, such as image emulation files, the Process Manager configuration file, etc.

4. You can return edit your Model at any time by clicking **Edit** in the **Models** frame.

Configuring Search Parameters

Search parameters are used to constraint the search and configure object recognition and positioning algorithms.

1. Select the **Search** tab.
2. Leave the **Nominal Scale** check box enabled. Its value in the text box should be 1.0.
3. Leave the **Nominal Rotation** check box disabled, with a range between -180.0 to 180.0.
4. Leave all the other parameters to their default setting also, as shown in [Figure 26](#).

Search Constraints		Nominal	Minimum	Maximum
Scale	☒	1.000	0.500	2.000
Rotation	☐	0.000	-180.000	180.000
Instances to Find	☒	1		

Recognition	
Minimum Model Percentage	50

Figure 26 Search Panel of the Locator Tool

Executing the Locator and Verifying the Results

Once the Model is created and the search is configured, the first step is to verify that the Locator finds the object on the image that was used to create the model.

1. Select the **Results** tab.
2. Click **Execute** at the bottom of the Locator's properties window.
3. Verify in the grid that the instance was located correctly.

4. Click **Show Results Viewer** to view the located instance superimposed on the Image, as shown in [Figure 27](#).
5. Move the Locator properties window and the **Results Viewer** to a corner of your screen.

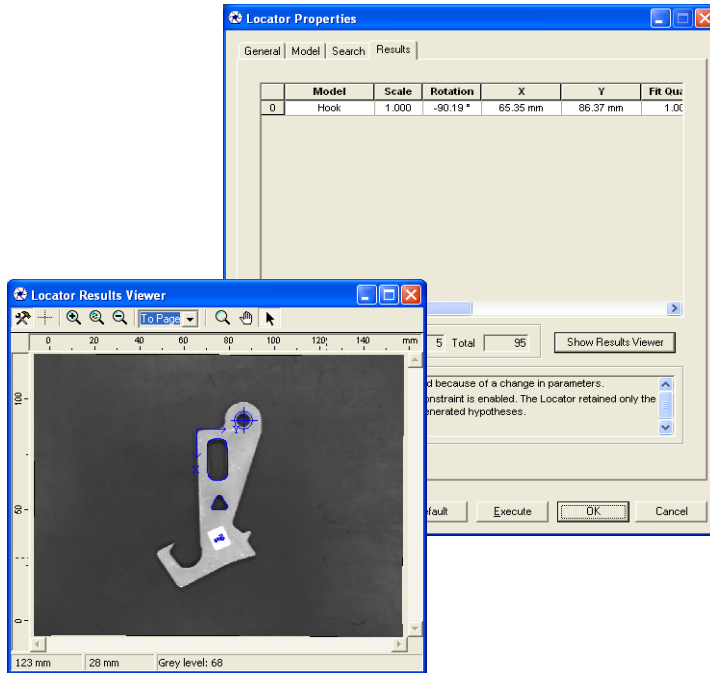


Figure 27 Results Viewer and Results Panel of the Locator Tool

The next step is to verify that the object is located in all the images in the emulation database.

1. In the Process Manager, open the Acquisition Device properties window. Select the **Configuration** panel, then click **Enable All** to enable all the images in the emulation database. Click **OK** to close the Acquisition Device's properties window.
2. Return to the Process Manager and click **Execute Sequence**.
3. In the **Results** panel of the Locator properties window, check if the instance was located correctly. Return to the Process Manager and click

Execute Sequence several times to apply the Locator to each of the images.

4. Close the Results Viewer window, return to the Locator Properties window and click **OK**.
5. Now is a good time to return to the Visual Basic form and save your work. Close the window to exit the Process Manager. To return to the application, you will have to right-click the HexSight icon and select **Edit** (or **Show Application Interface** if the icon is still selected).

Adding Code for the Locator

In this section, you will add code to output the properties of the instance found by the Locator in your application interface.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Locator Control**, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
Dim lManager As HSPROCESSMANAGERLib.HSProcessManager
Dim lLocator As HSLOCATORLib.HSLocator
InspectButton.Enabled = False
lManager = ApplicationControl.ProcessManager
lLocator = lManager.Process("Locator")
```

This code creates a reference to the Locator process contained in the HexSight Application Control. This reference will be used to access the tool programmatically.

2. In the existing "Do Loop" of the InspectButton_Click procedure, add the lines of code shown in bold:

```
Do

lManager.Execute("Acquisition", "Locator")

If lLocator.InstanceCount > 0 Then
```

```

' An instance of the object is found

TypeText.Text = lLocator.InstanceModelName(0)

' Output the properties of the located instance

ScaleText.Text = String.Format("{0:F2}",
    lLocator.InstanceScaleFactor(0))
RotationText.Text = String.Format("{0:F2}",
    lLocator.InstanceRotation(0))
TranslationXText.Text = String.Format("{0:F2}",
    lLocator.InstanceTranslationX(0))
TranslationYText.Text = String.Format("{0:F2}",
    lLocator.InstanceTranslationY(0))

Else

' No object is found
TypeText.Text = ""
ScaleText.Text = ""
RotationText.Text = ""
TranslationXText.Text = ""
TranslationYText.Text = ""

End If

' Compute elapsed time

TimeText.Text = String.Format("{0:F2}",
    lManager.ElapsedTime)

Application.DoEvents()

```

Loop While ContinuousCheck.Checked

Each time the loop is executed, this code executes the Application Control's sequence of process from "Acquisition" to "Locator". If no parameter is specified to the Execute method of the Process Manager, the whole sequence of processes will be executed, which in this case will yield the same result. After HexSight processes are executed, the pose (scale, rotation and translation) and type (model name) of the located instance are displayed in the interface.

3. Save your work and move on to the next step.

Testing the Application

You are ready to test your application.

1. To start the running Mode, press the **F5** key. Click **Inspect** a few times. The properties of the found instances displayed in the text boxes should be updated every time you press **Inspect**.
2. Enable the **Continuous Mode** check box and click **Inspect**. The application should run in continuous mode. Exit the Continuous Mode by disabling the check box. Exit the Run mode by closing the window or by clicking **Quit**.

This concludes this tutorial. After debugging, save your work and move on to the next tutorial where you will add a display to your application.

VB .NET Tutorial 2: Display

This second tutorial shows you how to add a Display to your application interface. A Display allows you to view Images and Scenes processed by the HexSight Application Control.

Adding the Display to the Interface

First, you will modify the interface of your application by adding the Display.

1. Start by extending the form. Set the width to approximately 735 and the height to approximately 420. Reorganize the text boxes and buttons to look like the form in [Figure 28](#).

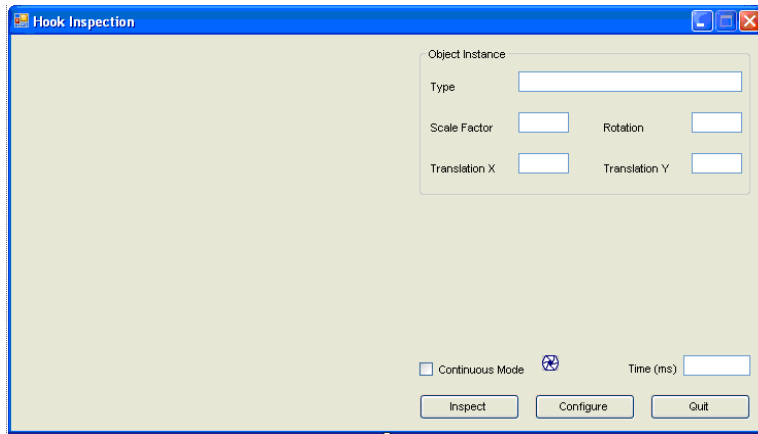


Figure 28 Resized VB .NET Form for Tutorial 2

2. From the toolbox context menu, select: *Choose Items...* Move to the COM Components tab and scroll down to find **HSDisplay Control**. Enable its check box and click **OK**. The HexSight Application Control appears as a lens shutter icon in the toolbox. The HexSight Display appears as a new icon in the toolbox.
3. Click the **HSDisplay Control** icon and drag a small box on the form. Set the width to 378 and height to 365. The default name is `AxHSDisplay1`. Rename it *ApplicationDisplay*. The interface of the project is now complete and should look like the form in [Figure 29](#).

4. Save your work.

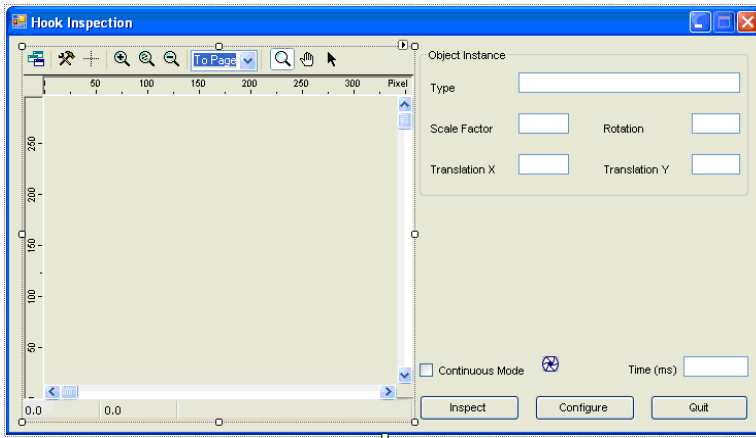


Figure 29 VB .NET Form for Tutorial 2

Adding Code for the Display

Code must now be added to update the Display output every time the inspection loop is executed.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Acquisition Device Control**, select it and click **OK**.
2. Insert the lines of code shown in bold into the existing `Form_Load` procedure:

```
Private Sub HookInspectionMainform_Load(...) Handles Me.Load
```

```
Dim lAcquisition As HSACQUISITIONDEVICElib.  
    HSAcquisitionDevice
```

```
lAcquisition = ApplicationControl.ProcessManager.  
    Process("Acquisition")
```

```
' Inits the Application Display
```

```
ApplicationDisplay.set_ImageDatabase(0,  
ApplicationControl.Database.Handle)
```

```
If Not lAcquisition Is Nothing Then
```

```
ApplicationDisplay.set_ImageViewName(0,  
lAcquisition.OutputView())  
ApplicationDisplay.set_ImageName(0,  
lAcquisition.OutputImage())
```

```
End If
```

```
ApplicationDisplay.Zoom = -1
```

```
' To ensure releasing of local references to Hexsight  
' ActiveX objects
```

```
lAcquisition = Nothing
```

```
GC.WaitForPendingFinalizers()
```

```
End Sub
```

This initialization code configures the Display Control to display the Image provided by the Acquisition Device output. It also sets the zoom to page mode, which automatically adjusts the zoom factor to view the entire image.

3. In the `InspectButton_Click` procedure, insert the lines of code shown in bold:

```
...
```

```
lManager.Execute("Acquisition", "Locator")
```

```
ApplicationDisplay.RemoveAllMarker()
```

```
If lLocator.InstanceCount > 0 Then
...
ApplicationDisplay.RefreshDisplay()

Application.DoEvents()
...
```

The RemoveAllMarker call removes all overlay graphic markers that could have been added to the previous image. The refresh call updates the display to show the new Image.

4. Save your work and test the application. Now along with the properties of the found instance, you should see an image of the object for the current instance.

VB .NET Tutorial 3: Caliper

In this tutorial, you will learn how to set up, configure and use a Caliper tool to precisely measure the distance between parallel edges on an object.

Modifying your Application Form

First you need to add two edit boxes to your application for the interface to display the measurement output by the Caliper.

1. On your application form, add a group box named **InspectionFrame** and add two text boxes, the first text box named **WidthText** and the second named **HeightText**. Set up your form as illustrated in [Figure 30](#).

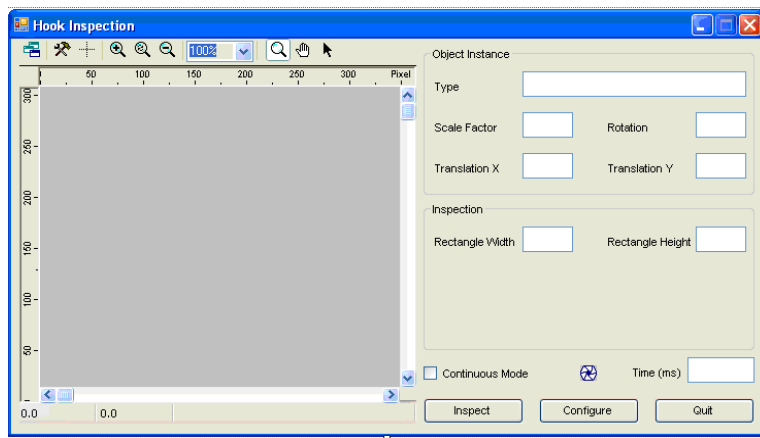


Figure 30 VB .NET Form for Tutorial 3

Placing the Caliper

Right-click the HexSight Application Control icon on the form and select **ActiveX -Edit** from the context menu. Click **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add an **HSCaliper** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *WidthCaliper* as the **Process Name** (instead of HSCaliper2).

2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Leave the **Lock Location Transform on a Selected Instance** check box enabled.
6. For the **Input Instance Scene**, select **Instance Scene** from **Locator** view.

The above settings will make your application fully model-based (automatic model-based mode). The Caliper tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Caliper is applied to an area on the object bounded by the Caliper's Rectangle, which appears in green. The Caliper will be used to measure the width of the rectangle-shaped hole on the part.

8. Place the Rectangle on the part as shown in [Figure 31](#).
9. Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees (the Y-axis is used to adjust the skew).
10. Click the center of the Rectangle and drag it into position.
11. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position and the orientation with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

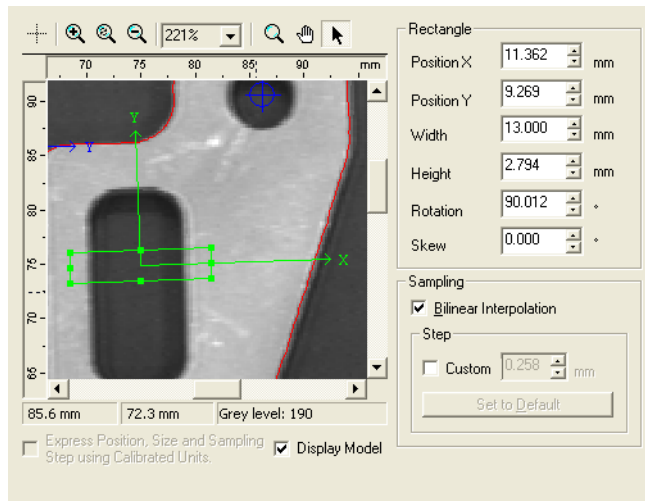


Figure 31 Adjusting the Size, Position and Orientation of the Caliper Rectangle

Note: The Caliper must always be as perpendicular as possible to the edges to be measured. For the current example, the tool had to be rotated to 90 degrees the skew was left at its default value of 0 degrees.

- Go to the **Transform** panel. The image shown is that of the current instance of the object with the tool transposed to the correct position on this instance. To see the tool applied to other instances, leave the Caliper window open and click **Execute Sequence** in the Process Manager. Yellow lines indicate where the Caliper has detected potential edges. Edge selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

Configuring the Caliper

The Caliper locates potential edge pairs. You will set up constraints that will allow the Caliper to measure only valid edge pairs, that is, those that meet the criteria you have determined.

The display in the Configuration panel represents the part of the Image bounded by the Rectangle. The Caliper processes information within the

Image along pixel lines, called projection paths, parallel to the Rectangle's Y-axis. In the display you can observe the following:

- Dotted yellow lines indicate potential edges.
- The blue curve shows a graphical representation of a projection that averages each column of pixels in the Rectangle to create a one-dimensional signal. Sharp changes on this curve indicate potential edges.
- The green magnitude curve is the first derivative of the projection curve. High and low peaks in this curve identify potential edges.
- Horizontal red lines indicate the magnitude threshold for the detection of potential edges.

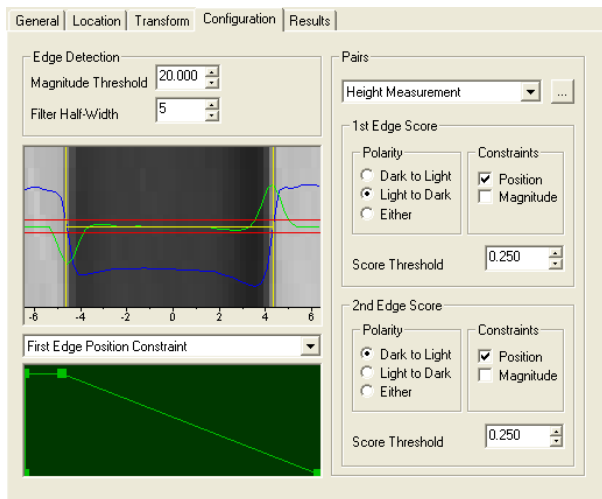


Figure 32 Caliper Tool — Configuration Panel

In this example, only one measurement pair is needed so the default pair named *Pair0* will be used. The constraints used for both edges will be polarity and position constraints. These constraints will be used to score each edge.

Polarity indicates changes in light values, from left to right, within the Rectangle.

Position constraints score each edge according to its position as defined by the Position Constraint histogram. **Magnitude** constraints, not used in this

tutorial, score each edge according to its magnitude as defined by the Magnitude Constraint histogram.

The Caliper attributes a score to each edge pair and to each individual edge. Scores range from 0 to 1 with the minimum acceptable score being determined by the **Score Threshold**, set by default at 0.250. Histogram functions for each constraint determine how the Caliper scores edges.

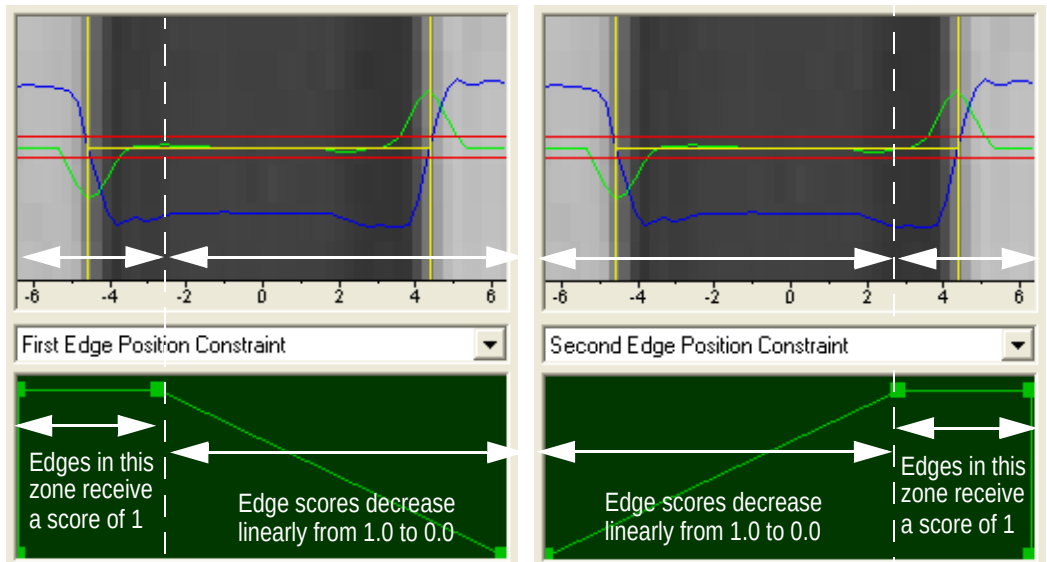


Figure 33 Adjusting Position Constraint Histograms

1. In the **1st Edge Score** frame, select **Polarity** as **Light to Dark** and enable the **Position** check box.
2. In the **2nd Edge Score** frame, select **Polarity** as **Dark to Light** and enable the **Position** check box.

Next you must set up the Position constraint histogram for each edge, in the bottom left window of the **Configuration** panel.

3. Use the combo box above the histogram window to select a constraint.
4. Use the mouse to drag the handles in the histogram window. Set the 1st and the 2nd edge position constraints as illustrated in [Figure 33](#).

5. Click **Execute**. The yellow dotted lines become solid and are joined by a solid horizontal line to indicate that the pair has been detected and measured.

Caliper Results

You will now observe the Caliper's results for various instances of the object.

Access the Caliper Results panel to observe the following:

- Valid edges are displayed as parallel vertical yellow lines.
- The Caliper measure is displayed as a horizontal yellow line, perpendicular to the edges.
- The Pair Count box indicates that the Caliper was configured to measure one pair.
- The Coordinate System combo box indicates that the positions are expressed with respect to the World coordinate system.

The grid of results indicates:

- The Caliper measure, named Size, indicated in calibrated units.
- The score for the edge pair as well as for each edge.
- Use the scrollbar to view the entire grid of results.

Now that the Caliper is properly configured to measure the rectangular hole feature, you will want to test and observe the Caliper on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the next step.

Adding Code for the Caliper

You will finish up by adding code to your application form that will display the measurement of the Caliper in the interface. You will also add a line marker on the display to show the measurement.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight**

Caliper Control, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
...
Dim lLocator As HSLOCATORLib.HSLocator
Dim lWidthCaliper As HSCALIPERLib.HSCaliper

Dim lMarkerName As String

InspectButton.Enabled = False

lManager = ApplicationControl.ProcessManager

lLocator = lManager.Process("Locator")
lWidthCaliper = lManager.Process("WidthCaliper")

Do
    ...
```

This will create a reference that will be used for programmatic access to the WidthCaliper process instantiated in the HexSight Application Control.

2. Modify the existing line, shown in italic, for the one below it:

```
lManager.Execute("Acquisition", "Locator")
lManager.Execute("Acquisition", "WidthCaliper")
```

This will execute the entire sequence of processes.

3. Now, to display the Caliper's measurement in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```
...
TranslationYText.Text = String.Format("{0:F2}",
lLocator.InstanceTranslationY(0))

' Display width caliper properties and marker
```

```

If lWidthCaliper.PairScore(0) > 0 Then

    lWidthCaliper.CoordinateSystem =
        HSCALIPERLib.hsCoordinateSystem.hsCoordinateSystemWorld

    WidthText.Text = String.Format("{0:F2}",
        lWidthCaliper.PairSize(0))

    lMarkerName = "Width"

    ApplicationDisplay.AddLineMarker(lMarkerName,
        lWidthCaliper.Edge1PositionX(0),
        lWidthCaliper.Edge1PositionY(0),
        lWidthCaliper.Edge2PositionX(0),
        lWidthCaliper.Edge2PositionY(0), True)

    ApplicationDisplay.set_MarkerAnchorStyle(lMarkerName,
        HSDISPLAYLib.hsMarkerAnchorStyle.hsCross)
    ApplicationDisplay.set_LineMarkerConstraints
        (lMarkerName, HSDISPLAYLib.hsLineMarkerConstraints.
        hsLineNoEdit)
    ApplicationDisplay.set_MarkerColor(lMarkerName,
        HSDISPLAYLib.hsColor.hsRed)

    ApplicationDisplay.set_MarkerDisplayName(lMarkerName,
        True)

End If

```

...

If the first pair of the Caliper (Index = 0) is found (Score > 0.250), its measurement is displayed in the WidthText text box. Using the World coordinate system, the calibrated X-Y position of the first and second edge of the pair is used to draw a non-editable line marker on the application display.

4. In case no instance is found, add the lines of code shown in bold:

```

...
TranslationYText.Text = ""

```



```
WidthText.Text = ""
```

```
End If
```

```
...
```

5. Save your work and test the application. You should see the measurement in the RectangleWidth text box of your application. You should also see the line marker you added on the Display highlighting the measurement of WidthCaliper.

Adding a Second Caliper

Using the previous steps, try adding a second Caliper that will measure the height of the rectangular hole in the hook. Add the appropriate code to display the result in the remaining text box (Rectangle Height) on your application form. Also add the code to draw another line marker on the Display.

Refer to the Application Example in the following folder:
InstallationPath\Tutorial\VBNET\HookInspection\Part3.

VB .NET Tutorial 4: Blob Analyzer

In this tutorial, you will learn how to set up, configure and use a Blob Analyzer tool to find, label and analyze irregular shaped features on the part.

Modifying your Application Form

First you need to add two text boxes to your application so you can observe the measurement output by the Blob Analyzer.

1. On your application form, add two text boxes. Name the first one **DiameterText** and the second **OffsetText**, and add appropriate labels. Set up your form as illustrated in [Figure 34](#).

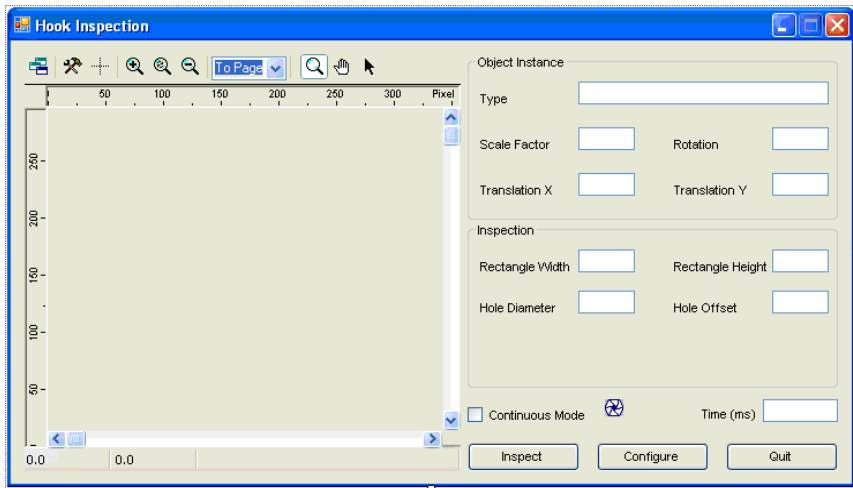


Figure 34 VB .NET Form for Tutorial 4

Placing the Blob Analyzer

Return to the Process Manager (right-click the HexSight Application Control icon, select **ActiveX -Edit**). Click **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add a **HSBlobAnalyzer** process from the **HexSight Inspection Tools** folder of the Process Manager. Set the tool's General panel as follows:

1. Enter *HoleBlob* as the **Process Name** (instead of HSBlobAnalyzer4).
2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Leave the **Create Blob Image** check box disabled.
4. Enable the **Apply Transform to a Model-defined Location** check box.
5. Select **Hook** from **Model** view as the **Input Model Scene**.
6. Enable the **Lock Location Transform on a Selected Instance** check box.
7. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Blob Analyzer tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

8. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Blob Analyzer will be applied to an area on the object bounded by the Blob Analyzer's Rectangle, which appears in green. The Blob Analyzer will be used to inspect the diameter and offset of the circular hole on the part.

9. Place the Rectangle on the part as shown in [Figure 35](#). Click the center of the Rectangle and drag it into position. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position with the text boxes if needed. Once the Rectangle is placed, click **Execute**.
10. Go to the **Transform** panel. The image shown is that of the current instance of the object as located in the second image of the emulation database (if you clicked twice on **Execute Sequence** before creating the Blob Analyzer tool). The red target marker indicates the center of the blobs detected by the Blob Analyzer. Blob selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

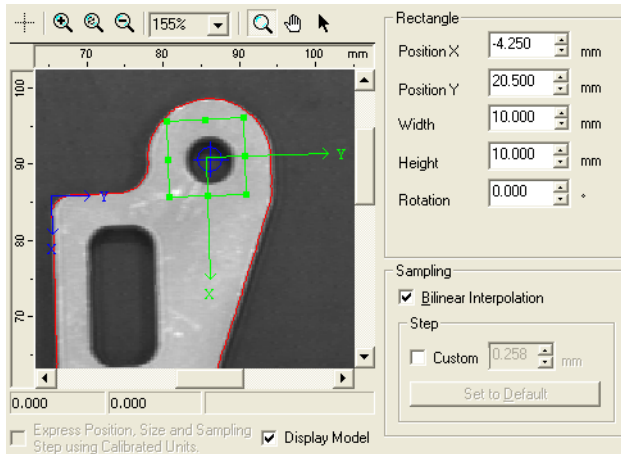


Figure 35 Adjusting the Size, Position and Orientation of the Blob Analyzer Rectangle

Configuring the Blob Analyzer

You will set up constraints that will allow the Blob Analyzer to find only valid blobs, that is, those that meet the criteria you have determined.

The display in the Configuration panel represents the part of the Image bounded by the Rectangle. The Blob Analyzer processes information within this Rectangle. In the display you can observe the following:

- The blob, as defined by the Image Segmentation Mode, is represented in green.
- A blob marker appears in red in the center of each blob.
- When displaying the Pixel Weight Image, you can see the weight value of each pixel as defined by the graphical function of the Image Segmentation box. White pixels have a weight of 1 while black pixels have a weight of 0. Pixels having a weight between 0 and 1 are displayed with a proportional greylevel.

1. First, set the Blob Constraints: **Minimum Area** to 0 and **Maximum Area** to 100, which corresponds to the area defined by the Rectangle.

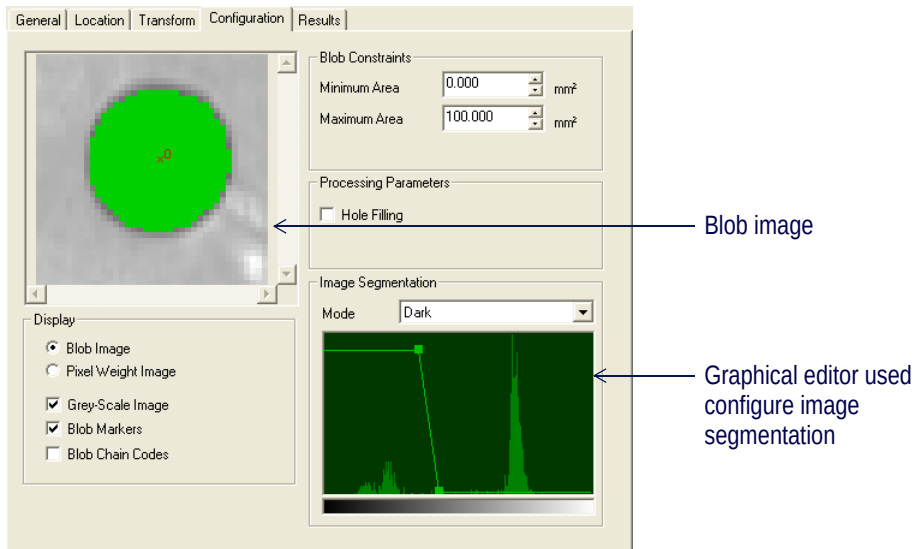


Figure 36 Blob Analyzer — Configuration Panel

Now you must set the segmentation function using the graphical editor in the Image Segmentation box. The graphic shows a histogram of the distribution of pixels inside the Blob Analyzer's Rectangle. Black (0) is at the far left and white (255) at the far right. In this example, there is a small peak in the black portion (around 58) representing the blob, and a higher peak (around 186) representing the part color. You want to include some pixels in the transition, with a weight function proportional to the grey-scale value.

2. Since we are looking for a dark blob on a white background, set the **Mode** combo box to **Dark**.
3. Use the mouse to drag the handles in the graphic function editor. Set the Image Segmentation weight function as illustrated in [Figure 37](#).
4. Click **Execute**. The segmented blob now corresponds to the hole in the part.

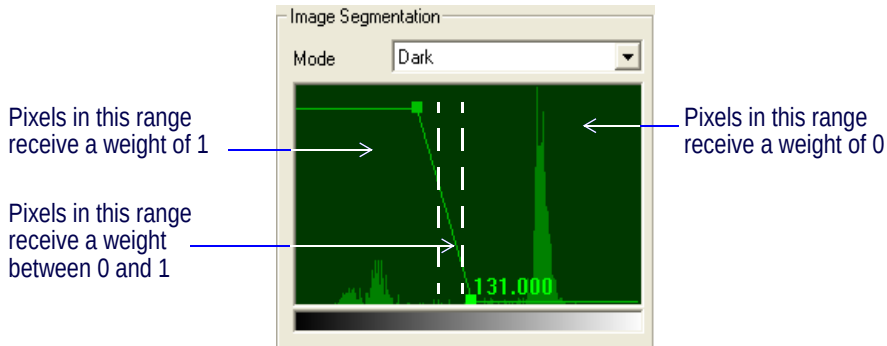


Figure 37 Image Segmentation Function Editor

Blob Analyzer Results

You will now observe the Blob Analyzer's results for various instances of the object.

1. In the **Results** tab, enable the **Sort Results** check box and select **Area** with the **Sort by** combo box.
2. Click **Execute**.

Use the Blob Analyzer Results panel to observe the following:

- The Blob Count box indicates that the Blob Analyzer has found one blob.
- An image is displayed with a blob marker indicating the center of each blob found for this instance of the object.
- The Coordinate System combo box selection indicates that the positions are returned with respect to the World coordinate system.
- A list of the selected types of blob properties that are computed: General, Perimeter, Intrinsic Inertia, etc.

The grid of results shows:

- The position of the blob according to the selected coordinate system

- The blob area indicated in calibrated units
- The properties computed for the blob. To view the complete set of properties, use the grid's tabs and scrollbar.

Now that the Blob Analyzer is properly configured to inspect the circular hole feature, you will want to test and observe the Blob Analyzer on other instances of the object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the final step.

Adding Code for the Blob Analyzer

You will finish up by adding code to your application form that will display the measurements of the Blob Analyzer in the interface. You will also add a target marker on the display to show the position and diameter of the hole.

1. From the Visual Studio menu bar, select Project>References.... Scroll down to find HexSight Blob Analyzer Control, enable its check box and click **OK**. Double-click **Inspect**. Insert the lines of code show in bold:

```
...
Dim lHeightCaliper As HSCALIPERLib.HSCaliper
Dim lHoleBlob As HSBLOBANALYZERLib.HSBlobAnalyzer

Dim lMarkerName As String
Dim lHoleDiameter As Single
Dim lHoleOffset As Single

...
lHeightCaliper = lManager.Process("HeightCaliper")
lHoleBlob = lManager.Process("HoleBlob")
...
```

This will create a reference that will be used for programmatic access to the hole blob process instantiated in the HexSight Application Control.

2. Modify the existing line, shown in italic, for the one below it:

```
lManager.Execute("Acquisition", "HeightCaliper")
lManager.Execute("Acquisition", "HoleBlob")
```

This will execute the entire sequence of processes.

3. Now to display the Blob Analyzer's measurements in the text boxes and to draw the marker on the application display, insert the lines of code shown in bold:

```

If lHeightCaliper.PairScore(0) > 0 Then
    ...
End If

' Display hole blob properties and marker

If lHoleBlob.BlobCount > 0 Then

    lHoleDiameter = System.Math.Sqrt(4 *
        lHoleBlob.BlobArea(0) / System.Math.PI)

    DiameterText.Text = String.Format("{0:F2}",
        lHoleDiameter)

    lHoleBlob.CoordinateSystem = HSBLOBANALYZERLib.
        hsCoordinateSystem.hsCoordinateSystemObject

    lHoleOffset = System.Math.Sqrt(lHoleBlob.
        BlobPositionX(0) ^ 2 + lHoleBlob.BlobPositionY(0)
        ^ 2)

    OffsetText.Text = String.Format("{0:F2}",
        lHoleOffset)

    lHoleBlob.CoordinateSystem = HSBLOBANALYZERLib.
        hsCoordinateSystem.hsCoordinateSystemWorld

    lMarkerName = "Hole"

```



```

ApplicationDisplay.AddTargetMarker(lMarkerName,
    lHoleBlob.BlobPositionX(0),
    lHoleBlob.BlobPositionY(0), lHoleDiameter / 2,
    True)
ApplicationDisplay.set_TargetMarkerConstraints
(lMarkerName, HSDISPLAYLib.hsTargetMarker
Constraints.hsTargetNoEdit)
ApplicationDisplay.set_MarkerColor(lMarkerName,
    HSDISPLAYLib.hsColor.hsRed)

```

End If

4. In case no instance is found, add the lines of code shown in bold:

```

...
HeightText.Text = ""
DiameterText.Text = ""
OffsetText.Text = ""

```

End If

If the blob is found, the offset of the hole from the origin of the Object coordinate system and the diameter of the hole are computed and displayed in the OffsetText and DiameterText text boxes. Using the World coordinate system, the calibrated X-Y position of the blob is used to draw a non-editable target marker on the application display.

5. Save your work and test the application. You should see the computed diameter and offset of the hole in the text boxes of your application. You should also see the target marker you added on the Display, highlighting the position and the diameter of the hole.

VB .NET Tutorial 5: Pattern Locator

In this tutorial, you will learn how to set up, configure and use a Pattern Locator tool to find and locate instances of the “HS” printed on the part.

Modifying your Application Form

First you need to add one text box to your application so you can display the result output by the Pattern Locator.

1. On your application form, add one text box. Name it **LabelText** and add an appropriate label. Set up your form as illustrated in [Figure 38](#).

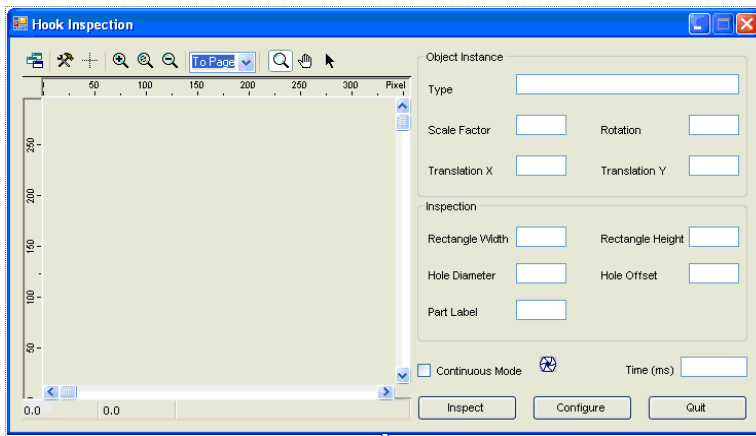


Figure 38 VB .NET Form for Tutorial 5

Placing the Pattern Locator

Select the HexSight Application Control on the form. With the right-click menu, select **ActiveX -Edit**. Click on **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add the **HSPatternLocator** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *LabelLocator* as the **Process Name** (instead of HSPatternLocator5).

2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Pattern Locator tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Pattern Locator will be applied to an area on the object bounded by the tool's Rectangle. The Pattern Locator will be used to verify the presence of the "HS" label printed on the part.

8. Place the Rectangle on the part as shown in the illustration. Click the center of the Rectangle and drag it to the center of the "HS" label. Use the corner and side handles to adjust the width and height of the search rectangle to fit the label. Zoom in on the label and by clicking and dragging the arrow on the tool's X-axis, adjust the rotation to fit the orientation of the label. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.

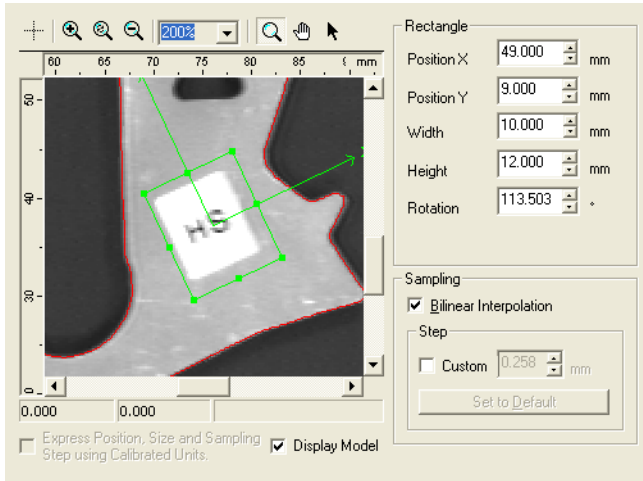


Figure 39 Adjusting the Size, Position and Orientation of the Location Rectangle

9. Go to the **Transform** panel. The image shown is that of the current instance of the object as located in the second image of the emulation database (if you clicked twice on **Execute Sequence** before creating the Pattern Locator tool). To see the tool applied to the other instances, leave the Pattern Locator open and click **Execute Sequence** in the Process Manager. The Pattern Locator will be configured to check for the presence of the label in the next step. Open the **Pattern** panel and move on to the next step.

Placing the Pattern Rectangle

You will set up the rectangle for defining the pattern that you want the Pattern Locator find within the Image. In this tutorial, the Pattern Locator will be used to verify the presence of the "HS" label printed on the part.

1. Place the Rectangle on the part as shown in the illustration.
2. Click the center of the rectangle and drag it to the center of the "HS" label. Use the corner and side handles to adjust the width and height of the Pattern Rectangle to fit the label.

3. Zoom in on the label and by clicking and dragging the arrow on the tool's X-axis, adjust the rotation to fit the orientation of the label.
4. Readjust the size of the Pattern Rectangle to include only the "HS" characters in the pattern. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.
5. Open the **Configuration** panel and move on to the next step.

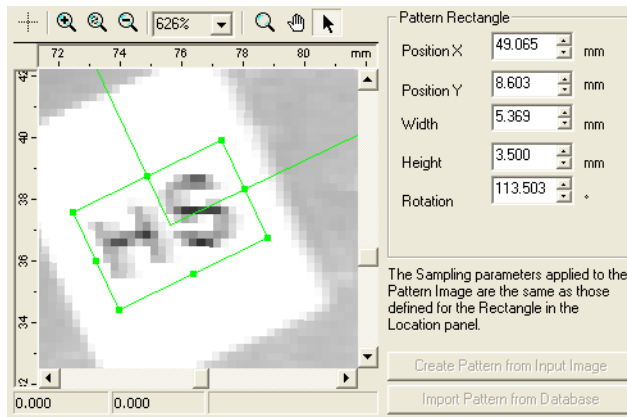


Figure 40 Adjusting the Size, Position and Orientation of the Pattern Rectangle

Configuring the Pattern Locator

You will set up the parameters of the tool in order to detect the grey-scale pattern on the part. Select the Configuration tab and set the following parameters:

1. Set the **Instance To Find** value to 1. This sets the maximum number of pattern instances to be found.
2. Set the **Match Threshold** value to 0.500. This sets the minimum strength score required to accept a valid match
3. Leave the **Automatic Coarseness Selection** check box enabled. The Pattern Locator will automatically determines the **Search Coarseness** and **Positioning Coarseness** levels.

4. Click **Execute**.

Pattern Locator Results

You will now observe the Pattern Locator's results for various instances of the object. Select the **Results** tab to observe the following:

- Each valid pattern is displayed with a blue rectangle marker indicating the position of the pattern detected in the Search Rectangle.
- The left display shows the image of the Search Rectangle.
- The right display shows the image of the reference pattern.

The Coordinate system combo indicates that the positions are returned with respect to the World coordinate system.

The grid of results indicates:

- The strength of the match.
- The position of the pattern according to the selected coordinate system.
- The rotation of the pattern in calibrated rotation units.

Use the scrollbar to view the entire grid of results.

Now that the Pattern Locator is configured properly to check the presence of the "HS" label, you will want to test and observe the Pattern Locator on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the final step.

Adding Code for the Pattern Locator

You will finish up by adding code to your Application form that will display the result of the pattern search in the interface. You will also add a point marker on the display to show the position of the label located by the Pattern Locator.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight**

Pattern Locator Control, select it and click **OK**. Double-click **Inspect**.

2. Insert the lines of code shown in bold:

```
...
Dim lHoleBlob As HSBLOBANALYZERLib.HSBlobAnalyzer
Dim lLabelLocator As HSPATTERNLOCATORLib.HSPatternLocator
...
lHoleBlob = lManager.Process("HoleBlob")
lLabelLocator = lManager.Process("LabelLocator")
```

This will create a reference that will be used for programmatic access to the LabelLocator process instantiated in the HexSight Application Control.

3. Modify the existing line, shown in italic, for the one below it:

```
lManager.Execute("Acquisition", "HoleBlob")
lManager.Execute("Acquisition", "LabelLocator")
```

This will execute the entire sequence of processes.

4. Now to display the result of the search in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```
If lHoleBlob.BlobCount > 0 Then
...
End If

' Display label locator properties and marker

If lLabelLocator.MatchCount > 0 Then

    LabelText.Text = "Present"

    lLabelLocator.CoordinateSystem = HSPATTERNLOCATORLib.
hsCoordinateSystem.hsCoordinateSystemWorld

    lMarkerName = "HSLabel"
```

```

ApplicationDisplay.AddPointMarker(lMarkerName,
    lLabelLocator.MatchPositionX(0),
    lLabelLocator.MatchPositionY(0), True)

ApplicationDisplay.set_MarkerDisplayName(lMarkerName,
    True)
ApplicationDisplay.set_MarkerColor(lMarkerName,
    HSDISPLAYLib.hsColor.hsBlue)

```

Else

```
LabelText.Text = "Absent"
```

End If

5. In case no instance is found, add the line of code shown in bold:

```

...
OffsetText.Text = ""
LabelText.Text = ""

```

End If

6. Save your work and test the application. You should see the presence/absence of the "HS" label in the text box of your application. You should also see the point marker you added on the Display, highlighting the position of the label.

VB .NET Tutorial 6: Edge Locator

In this tutorial, you will learn how to set up, configure and use two Edge Locator tools to measure non-parallel edges on the part.

Modifying your Application Form

First you need to add one text box to your application so you can observe the result output by the Edge Locator processes.

1. On your application form, add one text box. Name it **PartWidthText** and add an appropriate label. Set up your form as illustrated in [Figure 41](#).

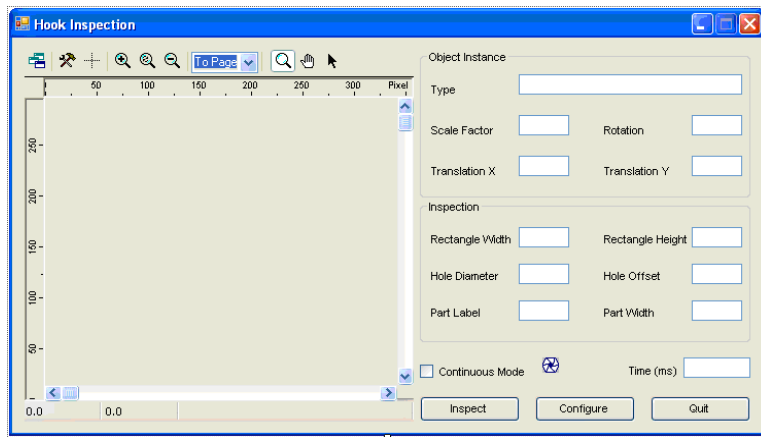


Figure 41 VB .NET Form for Tutorial 6

Placing the Edge Locator

Select the HexSight Application Control on the form. With the right-click menu select **ActiveX -Edit**. Click **Execute Sequence** twice to locate the object in the second Image of the emulation database. Using the Process Manager, add the **HSEdgeLocator** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *PartLeftEdge* as the **Process Name** (instead of *HSEdgeLocator6*).
2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Edge Locator tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the Location panel.

The Location display shows the Model View. The Edge Locator will be applied to an area on the object bounded by the tool's Rectangle, which appears in green. The Edge Locator will be used to measure the position of the left edge of the part.

8. Place the Rectangle on the part as shown in [Figure 42](#). Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees (the Y-axis is used to adjust the skew). Click the center of the Rectangle and drag it into position. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.

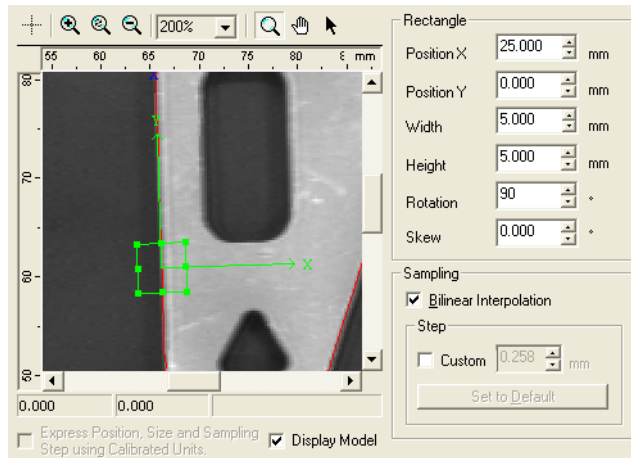


Figure 42 Adjusting the Size, Position and Orientation of the Edge Locator Rectangle

Note: The Edge Locator must always be as perpendicular as possible to the edges to be measured. For the current example, the tool had to be rotated to 90. Here the skew was left at its default value of 0 degrees.

9. Go to the **Transform** panel. The Image shown is that of the current instance of the object, as located in the second Image of the emulation database (if you clicked on **Execute Sequence** twice before creating the Edge Locator tool as indicated previously). To see the tool applied to other instances, leave the Edge Locator window open and click **Execute Sequence** in the Process Manager. Yellow lines indicate where the Edge Locator has detected potential edges. Edge selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

Configuring the Edge Locator

The Edge Locator detects potential edges. You will set up constraints that will allow the Edge Locator to locate only valid edges, that is, those that meet the criteria you have determined.

The graphic display in the Configuration panel represents the image information contained in the Edge Locator Rectangle:

- Yellow lines indicate detected edges.
- The blue curve shows a graphical representation of a projection that averages each column of pixels in the Rectangle to create a one-dimensional signal. Sharp changes on this curve indicate potential edges.
- The green magnitude curve is the first derivative of the projection curve. High and low peaks in this curve identify potential edges.
- Horizontal red lines indicate the magnitude threshold for the detection of potential edges.

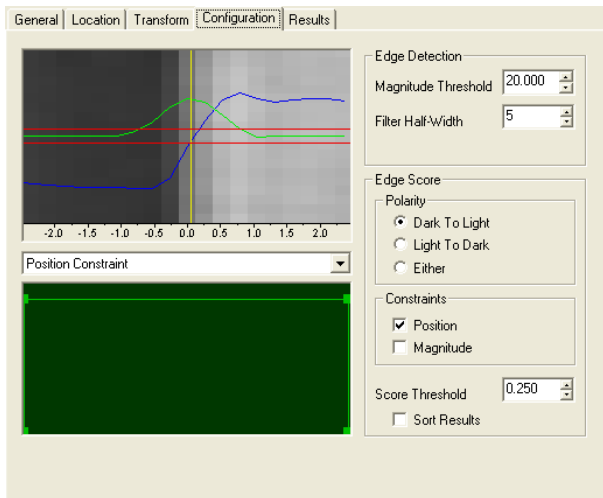


Figure 43 Edge Locator— Configuration Panel

In this example, the constraints used to score edges will be polarity and position constraints.

Polarity indicates changes in light values, from left to right, within the Rectangle.

Position constraints score each edge according to its position as defined by the Position constraint histogram. **Magnitude** constraints, not used in this

tutorial, score each edge according to its magnitude as defined by the Magnitude Constraint histogram.

The Edge Locator attributes a score to each individual edge. Scores range from 0 to 1 with the minimum acceptable score being determined by the **Score Threshold**, set by default at 0.250. Histogram functions for each constraint determine how the Edge Locator scores edges.

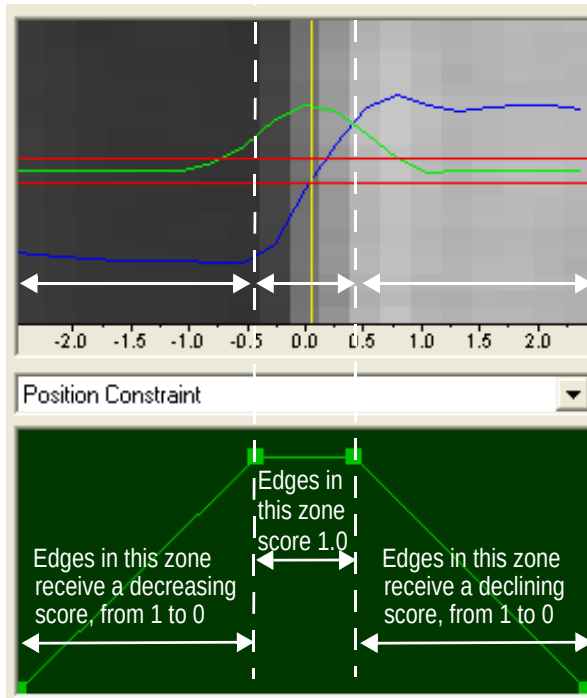


Figure 44 Position Constraint Histogram

1. In the **Edge Score** box, select **Polarity** as **Dark to Light** and enable the **Position** check box.

Next you must set up the Position Constraint histogram for the edge, in the bottom left window of the Configuration panel.

2. Use the mouse to drag the handles in the histogram editor of the selected constraint. Set the Edge Position Constraint histogram as illustrated in [Figure 44](#).

3. Click **Execute**.

Edge Locator Results

You will now observe the Edge Locator's results for various instances of the object. Access the Edge Locator Results panel to observe the following:

- Valid edges are displayed as vertical yellow lines.
- The Edge Count box indicates that the Edge Locator has found an edge.
- The Coordinate System combo box indicates that the positions are expressed with respect to the World coordinate system.

The grid of results indicates:

- The score of the found edges.
- The position of the edges, expressed with respect to the selected coordinate system.
- The rotation of the edges.
- The scores of each individual constraint.

Use the scrollbar to view the entire grid of results.

Now that the Edge Locator is configured to locate the left edge of the part, you will want to test and observe the Edge Locator on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the next step.

Placing the Second Edge Locator

Using the Process Manager, add a second HSEdgeLocator process. Set the tool's General panel as follow:

1. Enter *PartRightEdge* as the **Process Name** (instead of HSEdgeLocator7).
2. Select **Image** from **Acquisition** view as the **Input Image**.

3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.
7. Click **Execute**. Open the **Location** panel.
8. Place the Rectangle on the part as shown in [Figure 45](#). Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees. Click the center of the Rectangle and drag it into position. Since the edge to be located is not orthogonal to the tool's X-axis, adjust the skew by clicking and dragging the arrow of the Y-axis to make it parallel to the right edge of the part. Fine-tune the position with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

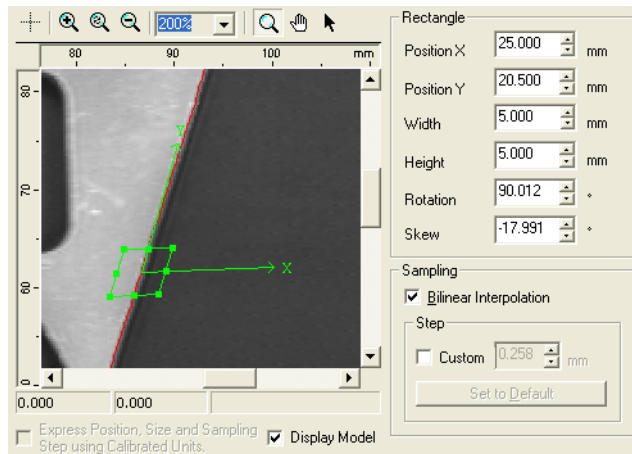


Figure 45 Adjusting Rectangle for the Hook's Right Edge

9. In the **Edge Score** box of the Configuration panel, select **Polarity** as **Light to Dark** and enable the **Position** check box.

10. Set up the Position Constraint using the histogram editor, just as that of the PartLeftEdge Edge Locator.
11. Save your work and move on to the final step.

Adding Code for the Edge Locator

You will finish up by adding code to your application form that will display the width of the part computed from the results of the two Edge Locator tools. You will also add a line marker on the display to show the measure on the part.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Edge Locator Control**, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
...
Dim lLabelLocator As HSPATTERNLOCATORLib.HSPatternLocator
Dim lPartLeftEdge As HSEdgeLocatorLib.HSEdgeLocator
Dim lPartRightEdge As HSEdgeLocatorLib.HSEdgeLocator
...
Dim lHoleOffset As Single
Dim lPartWidth As Single
...
lLabelLocator = lManager.Process("LabelLocator")
lPartLeftEdge = lManager.Process("PartLeftEdge")
lPartRightEdge = lManager.Process("PartRightEdge")
```

This will create a reference that will be used for programmatic access to the two Edge Locator processes instantiated in the HexSight Application Control.

2. Modify the existing line, shown in *italic*, for the one below it:

```
lManager.Execute("Acquisition", "LabelLocator")
lManager.Execute("Acquisition", "PartRightEdge")
```

This will execute the entire sequence of processes.

3. Now to display the measurement in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```

...
Else

    LabelText.Text = "Absent"

End If

' Display part edge left and part edge right properties
' and marker

If lPartLeftEdge.EdgeCount > 0 And
lPartRightEdge.EdgeCount > 0 Then

    lPartLeftEdge.CoordinateSystem = HSEdgeLocatorLib.
hsCoordinateSystem.hsCoordinateSystemObject
    lPartRightEdge.CoordinateSystem = HSEdgeLocatorLib.
hsCoordinateSystem.hsCoordinateSystemObject

    lPartWidth = lPartRightEdge.EdgePositionY(0) -
        lPartLeftEdge.EdgePositionY(0)

    PartWidthText.Text = String.Format("{0:F2}", lPartWidth)

    lPartLeftEdge.CoordinateSystem = HSEdgeLocatorLib.
hsCoordinateSystem.hsCoordinateSystemWorld
    lPartRightEdge.CoordinateSystem = HSEdgeLocatorLib.
hsCoordinateSystem.hsCoordinateSystemWorld

    lMarkerName = "Part Width"

    ApplicationDisplay.AddLineMarker(lMarkerName,
        lPartLeftEdge.EdgePositionX(0),
        lPartLeftEdge.EdgePositionY(0),

```

```

        lPartRightEdge.EdgePositionX(0),
        lPartRightEdge.EdgePositionY(0), True)

ApplicationDisplay.set_MarkerAnchorStyle(lMarkerName,
    HSDISPLAYLib.hsMarkerAnchorStyle.hsCross)
ApplicationDisplay.set_LineMarkerConstraints
(lMarkerName, HSDISPLAYLib.hsLineMarkerConstraints.
    hsLineNoEdit)
ApplicationDisplay.set_MarkerColor(lMarkerName,
    HSDISPLAYLib.hsColor.hsGreen)

ApplicationDisplay.set_MarkerDisplayName(lMarkerName,
    True)

```

End If

4. In case no instance is found, add the line of code shown in bold:

```

...
LabelText.Text = ""
PartWidthText.Text = ""

```

5. Save your work and test the application. You should see the part width in the Part Width text box of your application. You should also see the line marker you added on the Display highlighting the measurement.

VB .NET Tutorial 7: Calibration

In this tutorial, you will learn how to calibrate the system to perform acquisitions of live images. Calibration is an important step that can correct certain perspective and lens distortions. It also allows you to specify the basic length units you will use throughout your entire application.

Camera Calibration

The calibration is performed in the Acquisition Device. In this example, the calibration is performed with the largest calibration target provided with your HexSight package. This target has a dot pitch of 10 mm.

1. Open the Acquisition Device Properties. Select the **Configuration** panel. On the Input sub-panel, select **Copy Configuration** and then select **Paste Configuration**.
2. In the displayed window, enter *Camera* as the new configuration name.
3. Select the newly created *Camera* configuration in the list.
4. In the **Input** sub-panel, select **Generic Frame Grabber** and click **Apply**.
5. Go to the **Calibration** sub-panel. Select **Distortion Model**. Click **Show Calibration Dialog**. The **Distortion Model** dialog is displayed (see [Figure 46](#)).

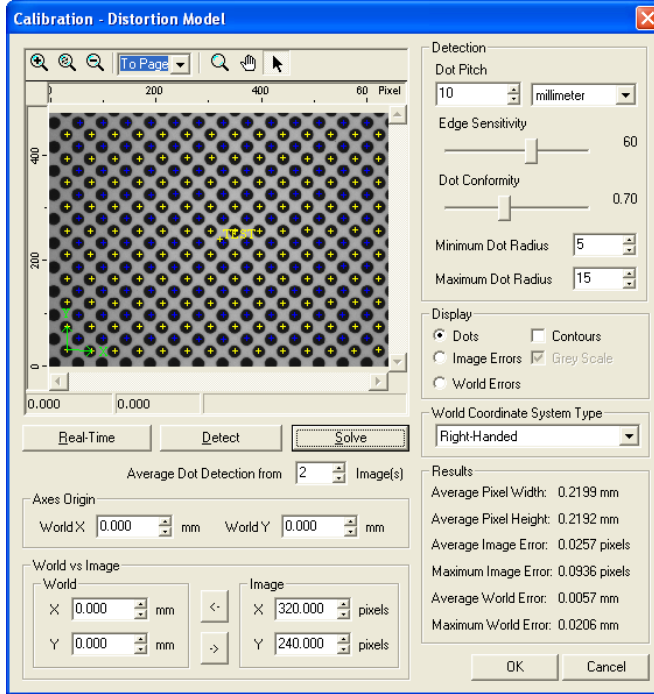


Figure 46 Distortion Model Calibration Dialog

6. Click **Real-Time** to display the image from the camera. Position the grid of dots to cover the whole area. Set the focus and other settings of your camera to obtain the best image possible. Then click **Real-Time** again to stop real time acquisitions
7. Click **Detect** to detect the dots on the target.
8. The dot pitch is 10.0 mm. Enter the dot pitch in the **Dot Pitch** box.
9. Zoom in at the lower left corner of the display. A green coordinate system marker should be displayed in the bottom left, as in [Figure 47](#). Note that this coordinate system is displaceable programmatically or with the mouse; leave it as is for this tutorial.

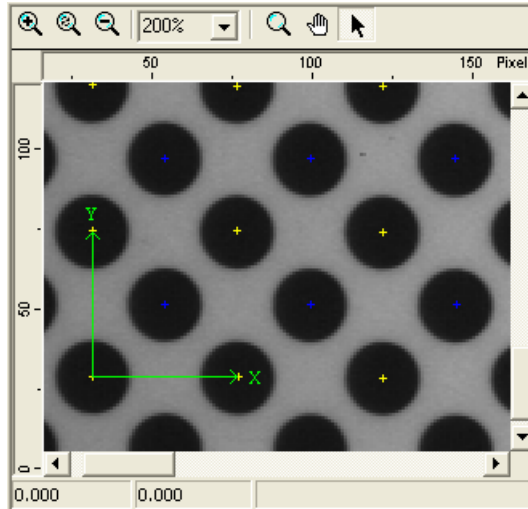


Figure 47 Zoom-in on the Coordinate Marker in the Calibration Dialog

10. Click **Solve**. The statistics appear in the **Results** frame. Click **OK**.

11. The calibration procedure is completed. The computed calibration parameters appear in the **Parameters** grid. Click **Apply** to apply the calibration to the *Camera* configuration.

12. Click **OK** to close the Acquisition Device's properties window.

13. You are now ready to run the application with live images using the Paper Hook provided with your HexSight package.

Visual C++ Tutorials

The tutorials in this chapter will guide you through the development of a HexSight application in Visual C++. As you follow the steps for each tutorial you will build an object location application to which you will add and configure a full range of inspection tools.

Note: This tutorial presumes you have a basic beginner's knowledge of Visual C++, MFC and COM. To compile the tutorials, the MFC source code must be installed and the "VisualC++_Path/MFC/src" directory must be added to the list of search paths used to find Included files (select the menu Tools>Options >Directories to append the list of search paths).

Chapter Outline

[Visual C++ Tutorial 1: Locator](#)

[Visual C++ Tutorial 2: Display](#)

[Visual C++ Tutorial 3: Caliper](#)

[Visual C++ Tutorial 4: Blob Analyzer](#)

[Visual C++ Tutorial 5: Pattern Locator](#)

[Visual C++ Tutorial 6: Edge Locator](#)

[Visual C++ Tutorial 7: Calibration](#)

Visual C++ Tutorial 1: Locator

This first tutorial shows you how to build a basic HexSight application that will locate a model-defined object at whatever angle and displacement it appears.

Creating the Project

In this section, you will create the Project of the application, add basic lines of code to interact with the interface, and add a HexSight Application Control, which you will edit to build the application.

Constructing the Project

Here you will build the project that will allow you to specify which type of program you want and which files to use to build it.

1. Start Visual Studio. Create a new **MFC AppWizard (exe)** project. Name it *HookInspection* and specify a working folder in the location field, then click **OK**.
2. In Step 1 of **MFC AppWizard**, select **Dialog based**.
3. In Step 2, disable the **About box** and specify **Hook Inspection** for the title. Click **Finish**.
4. Developer Studio shows you which options will be used to generate the sketch of the project of your program. Click **OK**.
5. The most important files generated by the wizard are **HookInspectionDlg.h** and **HookInspectionDlg.cpp**.

You can already compile this basic project, but you must add some lines of code and create the interface for your application to make this a useful project.

Constructing the Program Interface

Here you will build the interface that will allow you to interact with your application and visualize results.

1. Click on the **ResourceView** tab shown at the bottom left of the Workspace window to open the dialog template of the project.
2. Select the dialog template and resize it to approximately 226 X 149. Look in the status bar at the bottom of the screen; the size is shown at right.
3. On the dialog template, remove all existing dialog items and add a Group Box. Add 5 edit boxes respectively named **IDC_EDIT_TYPE**, **IDC_EDIT_SCALE**, **IDC_EDIT_ROTATION**, **IDC_EDIT_TRANSLATION_X** and **IDC_EDIT_TRANSLATION_Y**. Also set the Group Box caption, the edit box dimensions and add labels. The text for the edit boxes should be empty. Your dialog template should look like the dialog in [Figure 48](#).

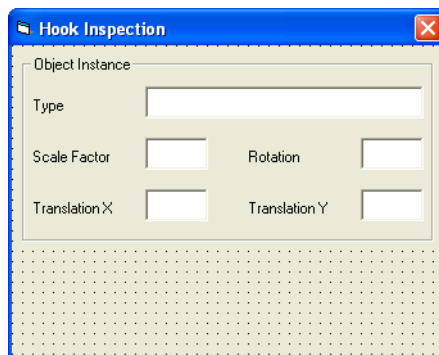


Figure 48 Visual C++ Dialog with Object Pose Information

4. At the bottom of the dialog template, add 3 command buttons named **IDC_BUTTON_INSPECT**, **IDC_BUTTON_CONFIGURE** and **IDC_BUTTON_QUIT**. Set the caption of these buttons respectively to: **Inspect**, **Configure** and **Quit**.
5. Add a check box named **IDC_CHECK_CONTINUOUS**, and set its caption to **Continuous Mode**. Finally, add an edit box named **IDC_EDIT_TIME** and an appropriate label. Place everything to resemble [Figure 49](#)
6. The interface is almost complete. Save your work.

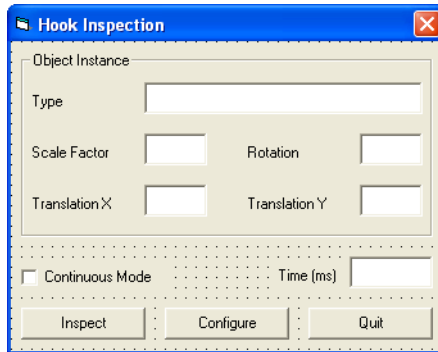


Figure 49 Visual C++ Dialog with Command Buttons

Adding the Member Variables

To simplify the interaction with the interface, we will now add some member variables with the help of MFC ClassWizard.

1. Press **CTRL+W**; the MFC ClassWizard window will appear.
2. Click on the **Member Variables** tab and select **CHookInspectionDlg** in the **Class Name** combo box. You will see something like the dialog in [Figure 50](#).

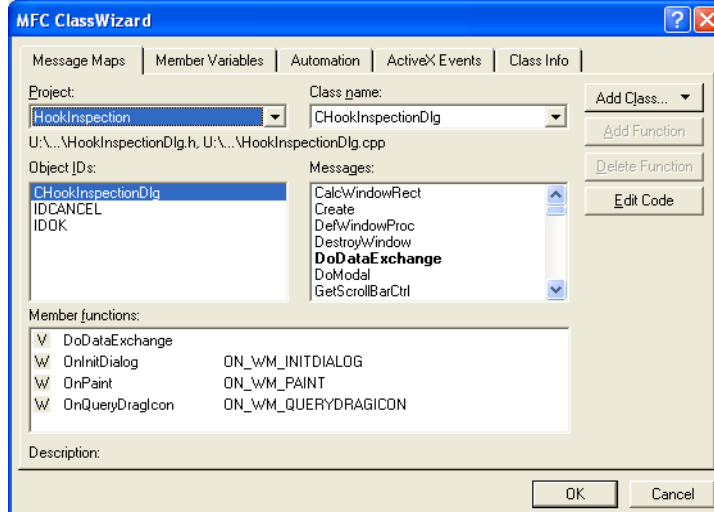


Figure 50 MFC ClassWizard Member Variables Editor

3. For each Control ID, you must add a member variable. To add a member variable, double-click the Control ID and complete the form. [Figure 51](#) shows the completed form for **IDC_BUTTON_CONFIGURE**.

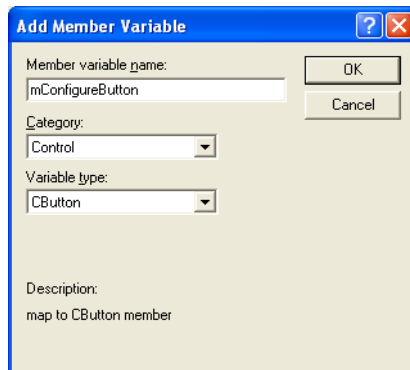


Figure 51 Adding a Member Variable with MFC ClassWizard

4. Add the other member variables according to the list in .

Table 2 Member Variables

Control ID	Type	Member
IDC_BUTTON_INSPECT	CButton	mInspectButton
IDC_BUTTON_QUIT	CButton	mQuitButton
IDC_CHECK_CONTINUOUS	CButton	mContinuousCheck
IDC_EDIT_ROTATION	CEdit	mRotationText
IDC_EDIT_SCALE	CEdit	mScaleText
IDC_EDIT_TIME	CEdit	mTimeText
IDC_EDIT_TRANSLATION_X	CEdit	mTranslationXText
IDC_EDIT_TRANSLATION_Y	CEdit	mTranslationYText
IDC_EDIT_TYPE	CEdit	mTypeText

Adding a HexSight Application Control

You will now add an ActiveX Control on the dialog template: the HexSight Application Control, which is the core of each HexSight application.

1. Right-click in the dialog template. In the pop-up window, select **Insert ActiveX Control...**
2. Scroll down to find **HSApplication Control**. Select it and click **OK**. The HexSight Application Control appears as a new icon in the dialog template. Place it as shown in [Figure 52](#).

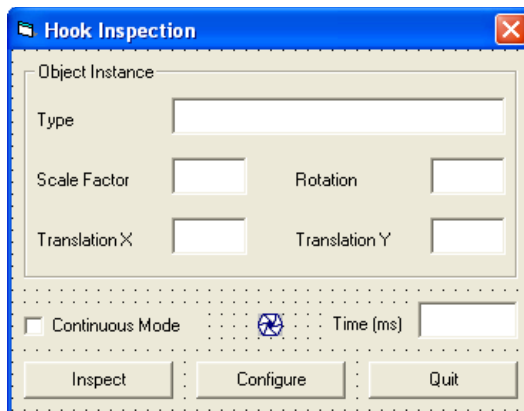


Figure 52 Visual C++ Dialog with HexSight Application Control

3. Right-click the **HexSight Application Control** icon and select **Properties** in the pop up window. Name it **IDC_CONTROL_APPLICATION**.
4. You must manually add a data member in the class.

IMPORTANT: DO NOT USE THE CLASSWIZARD TO ADD VARIABLES REPRESENTING HEXSIGHT CONTROLS. THE WRAPPERS GENERATED BY THE CLASSWIZARD MAY CAUSE COMPATIBILITY PROBLEMS WHEN USED FOR HEXSIGHT ACTIVE X CONTROLS. YOU **MUST** USE THE WRAPPERS DISTRIBUTED WITH HEXSIGHT. ALWAYS USE THE FOLLOWING PROCEDURE WHEN YOU ADD A HEXSIGHT CONTROL.
5. Open the HookInspectionDlg.h file. Insert the lines shown in bold:

```

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "HSAApplication.h"

[....]

//{{AFX_DATA(CHookInspectionDlg)
enum{IDD=IDD_HOOKINSPECTION_DIALOG};

```

```
HSAApplication mApplicationControl;
```

6. Open the HookInspectionDlg.cpp file and insert the line shown in bold into the DoDataExchange method:

```
//{{AFX_DATA_MAP(CHookInspectionDlg)
DDX_Control(pDX, IDC_CONTROL_APPLICATION, _
mApplicationControl);
```

7. Select the **FileView** tab at the bottom of the screen.
8. Right-click on HookInspection files node and select **Add Files to Project**.
9. Go to HexSight's installation directory and open the Wrappers directory. *Make a note of the entire path for this directory since you will need it later.*
10. Select the following files: HSAApplication.h, HSAApplication.cpp, HSCalibration.h, HSCalibration.cpp, HSDatabase.h, HSDatabase.cpp, HSErrorObject.h, HSErrorObject.cpp, HSExplorer.h, HSExplorer.cpp, HSIImage.h, HSIImage.cpp, HSLocator.h, HSLocator.cpp, HSModelEditorInterface.h, HSModelEditorInterface.cpp, HSProcessManager.h, HSProcessManager.cpp, HSScene.h, HSScene.cpp, HSView.h and HSView.cpp. Click **OK**.
11. Before compiling, add the Wrappers path in the list of directories used for the search of included files. Use the Project/Setting... Menu. Select the C/C++ tab, then indicate the Category **Preprocessor**. Modify the combo box **Setting For** to **All Configurations**. Add the Wrappers path (previously noted) in the **Additional include directories** edit box. Click **OK**.
12. Now you can compile your project without receiving error messages.

Adding Basic Code to the Main Form

You need to add basic code to be able to interact with your application. As you will add new features throughout the other tutorial sections, you will also add lines to this code.

1. Return to the Resource Editor and display the Dialog Box of your application.
2. Double-click **Quit**.
3. A window will appear (if not, this is because the *Double-click in dialog editor edits code* option is not selected in **Tools>Options>Compatibility** tab); double-click the message **BN_CLICKED**. Developer Studio will ask you to confirm the name of the function to add. Accept the default name (OnButtonQuit).

4. Click **Edit Existing** and add the lines of code shown in bold into the function:

```
void CHookInspectionDlg::OnButtonQuit()
{
    WaitForCompletion();

    EndDialog( IDOK );
}
```

This will terminate the application when the **Quit** button is clicked.

5. You must add the **WaitForCompletion** function in your project. Click on the **ClassView** tab and right-click on the **CHookInspectionDlg** class. Select the **Add Member Function...** command. The function's type is **void** and it is declared as **WaitForCompletion**. Indicate that this function is **private**. Click **OK** and add the lines of code shown in bold into the function:

```
void CHookInspectionDlg::WaitForCompletion()
{
    if ( mContinuousCheck.GetCheck() != 0 )
    {
        mContinuousCheck.SetCheck( 0 );
    }
}
```

6. Return to the Resource Editor, double-click **Configure** (follow the same as steps for the **Quit** button) and add the following lines of code:

```

void CHookInspectionDlg::OnButtonConfigure()
{
    ColeVariant lVariant;

    lVariant.vt = VT_ERROR;

    mApplicationControl.ShowInterface(FALSE, lVariant);

}

```

This will show the HexSight Application Control's interface in a non-modal window when the **Configure** button is clicked.

7. In the Resource Editor, double-click **Inspect** (follow the same as steps for the **Quit** button) and add the following lines of code:

```

void CHookInspectionDlg::OnButtonInspect()
{
    try
    {
        HSProcessManager lManager;

        mInspectButton.EnableWindow( FALSE );

        lManager = mApplicationControl.GetProcessManager();

        // Inspection loop

        do
        {
            DoEvents();
        }
        while( mContinuousCheck.GetCheck() == 1 );

        mInspectButton.EnableWindow( TRUE );
    }
}

```



```

    }
    catch( ... )
    {
        mInspectButton.EnableWindow( TRUE );

        throw ;
    }
}

```

This is the main loop of the inspection application. It loops until **Continuous Mode** is disabled in the interface.

8. In C++, there is no equivalent for the DoEvents function in Visual Basic; you must create it. You must add these lines at the beginning of the HookInspectionDlg.cpp file, just before the declaration of the constructor of CHookInspectionDlg:

```

void DoEvents( void )
{
    MSG lMessage;

    while ( PeekMessage( &lMessage, NULL, 0, 0, PM_REMOVE )
        != FALSE )
    {
        TranslateMessage( &lMessage );
        DispatchMessage( &lMessage );
    }
}

```

9. Lastly, you must add the Message handlers to stop the Continuous mode before exiting the program. Press **CTRL+W**; MFC ClassWizard will appear. Click on the **Message Maps** tab. Select CHookInspectionDlg in **ClassName**. Select CHookInspectionDlg in **Object IDs**. In the list of messages, find and double-click **WM_CLOSE** and **WM_DESTROY**. Add the following lines of code in the new functions:

```

void CHookInspectionDlg::OnClose()

```

```

{
    WaitForCompletion();

    CDialog::OnClose();
}

void CHookInspectionDlg::OnDestroy()
{
    CDialog::OnDestroy();

    WaitForCompletion();
}

```

You have completed the code needed to use your application. Run and test the application. When you are sure that everything works fine, save your work and go on to the next section.

Adding an Acquisition Device Tool

In this section, you will learn how to instantiate a process in the HexSight Application Control. You will also learn how to configure an Acquisition Device tool. This tool will acquire image from an existing database file to simulate real image acquisition.

Instantiating the Acquisition Device

In order to build a vision application, HexSight processes are instantiated using the Application Control's Process Manager.

1. On the dialog template, right-click the **HexSight Application Control**. Select **Edit** from the menu. The HexSight Application Control's interface opens and the active tab is the Process Manager. In the **Available Processes** list of the Process Manager, select **HSAcquisitionDevice** and click **Add Process**. The Acquisition Device Properties window appears.

At the end of this tutorial, the HexSight Application Control should look like the illustration in [Figure 53](#).

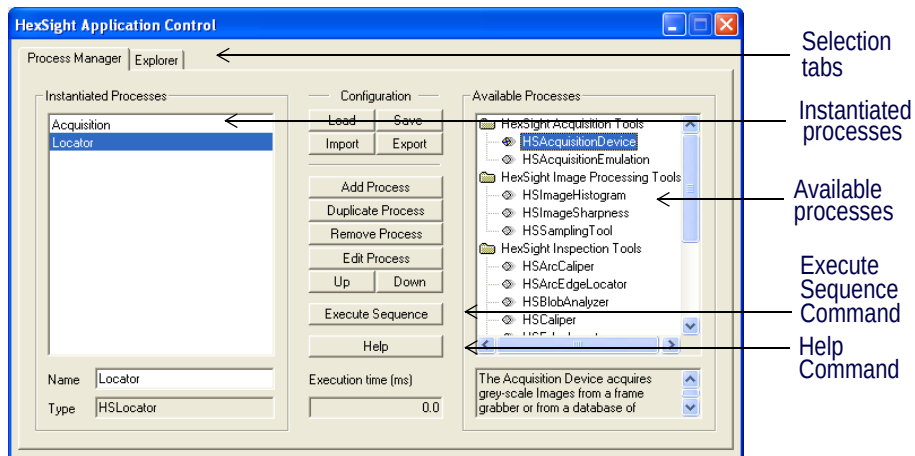


Figure 53 HexSight Application Control — Process Manager

Configuring the Acquisition Device

You will use an existing database of images to simulate a real image acquisition. The database used for this tutorial is constituted of images of the same objects, rotated at different angles.

1. In the Acquisition Device **General** panel, enter *Acquisition* as the **Process Name** (instead of HSAcquisitionDevice0).
2. In the **Configuration** panel, select the **Input** tab. Select **File Emulation** from the list.
3. Click **Load** then load the following file:
InstallationPath\Tutorial\VisualC++\HookInspection\Data\Hook.hdb.
4. Disable all the images of the emulation database except the first one: *Image0*. This ensures that this image will be used to create the model later in this tutorial.
5. Leave the default settings for the Calibration and Output sub-panels. Click **OK** to return to the Process Manager of the HexSight Application Control

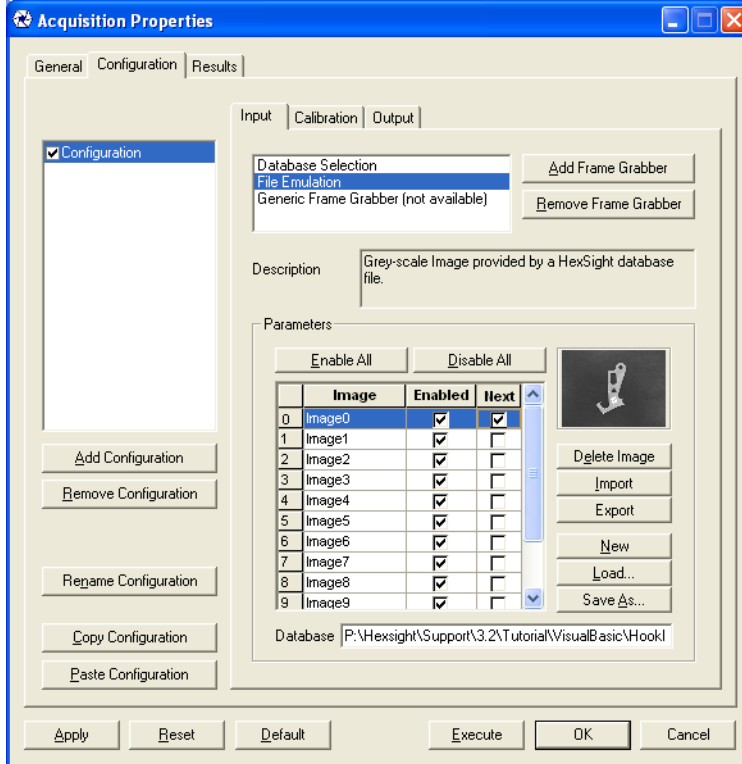


Figure 54 Acquisition Device — Configuration Panel

To sequentially execute the processes instantiated in the Process Manager, click **Execute Sequence**. At this stage, each time you click **Execute Sequence**, the Acquisition Device enables and outputs an new emulation images to the applications runtime database.

Viewing Scenes and Images

The Explorer panel in the HexSight Application Control shows the application's runtime database as a hierarchical list.

1. Select the **Explorer** panel in the **HexSight Application Control**.
2. Click on + symbols to expand **RunTime** and **Acquisition**.

- Click **Grey-Scale Image** to display the current Image output by the acquisition process.

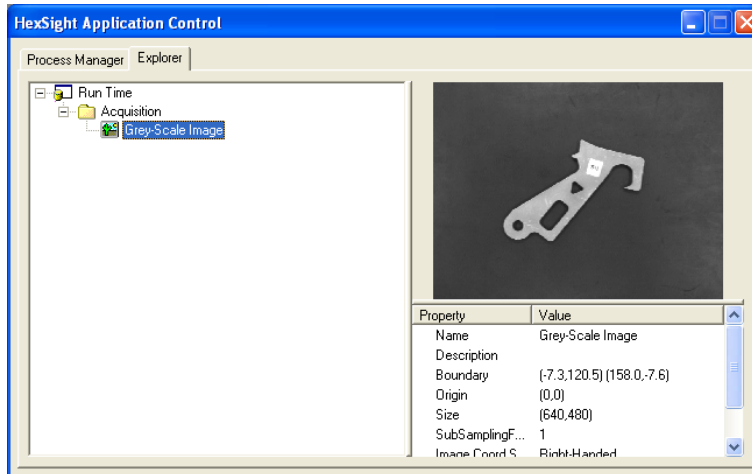


Figure 55 HexSight Application Control — Explorer

Adding a Locator Tool

The Locator locates model-defined objects within the input Image, using the geometry of object contours.

For this tutorial you will use only the basic Locator properties, therefore the **Advanced Parameters** check box should be left unchecked.

Instantiating the Locator

Here you add a Locator process the same way you added the Acquisition Device.

- In the HexSight Application Control Process Manager, select **HSLocator** from the **HexSight Locator Tools** folder. Click **Add Process**.
- In the properties window of the newly added Locator process, select the **General** tab.
- Enter the name *Locator* instead of HSLocator1.

4. In the **Input** frame, click on the selection icon at right to select **Acquisition>Grey-Scale Image**.

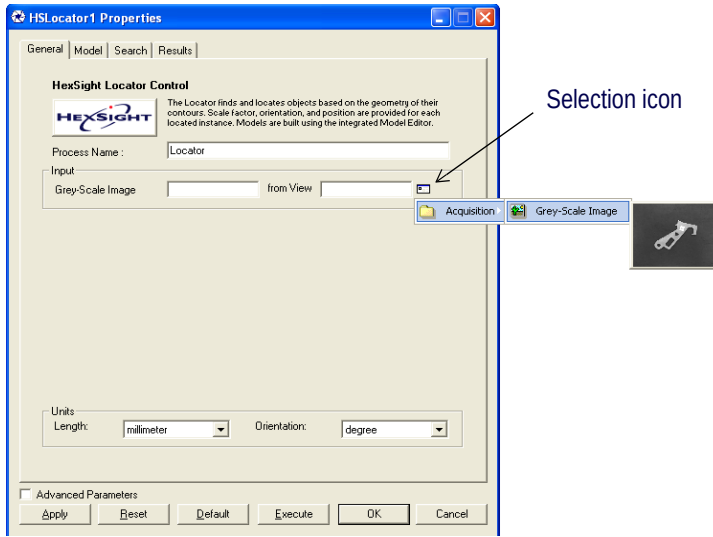


Figure 56 Database Selection Icon

5. Click **Execute**.

Creating a Model

In this section, you will learn how to create a Model of an object using the Model Editor. A Model is a reference or “ideal” representation of an object’s contours that will be used by the Locator’s search process to recognize and locate instances of the object. Models can be stored in database files for later use.

1. Select the **Model** tab, then click **Add** in the **Models** frame. The **Create New Model** prompt appears.

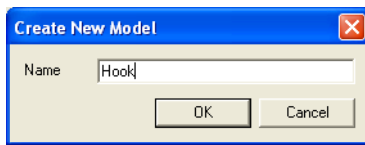


Figure 57 Create New Model Dialog

2. Name the Model *Hook*.
3. Click **OK** to open the Model Editor.

Using the Model Editor

The Model Editor is used to create new models and to modify existing models. Once you have created a model of the hook for this application, you will save it for later use with the rest of the tutorial.

Creating the Source Contours and Selecting Model Features

The Locator uses two coarseness levels to recognize and locate the object: the **Outline Level** and the **Detail Level**. The outline coarseness level is used to rapidly identify potential instances of the object. The detail coarseness level is used to confirm that it is the correct object and to refine its location within the image. Source contours will be created at the selected **Outline Level** and **Detail Level**. Features will be selected from these source contours to create the Model.

When the **Automatic Levels** check box is enabled, the two contour detection parameters (**Outline Level** and **Detail Level**) are automatically optimized by analyzing the Model's bounding area. Automatic contour detection parameters should usually be used when building a Model.

Contrast Threshold is used to set the sensitivity to contrast when generating the source contours. **Contrast Threshold** can be set manually, by selecting **Fixed Value** with the combo box, or automatically by selecting **Adaptive Low Sensitivity**, **Adaptive Normal Sensitivity** or **Adaptive High Sensitivity**.

Tracking Inertia is used to close small gaps that may appear in the source contours. It defines the longest gap that can be closed to connect two edge elements when building the source contours.

Once the source contours are detected, features must be selected to build the Model. Features can be selected manually, by setting the **Feature Selection** slider to **none**, or automatically by selecting **less**, **normal**, **more** or **all**. Setting the slider to **normal** will select the optimal amount of features at both

the Outline and Detail Levels. Increasing the setting will add more features, which may slow the search process. Decreasing the setting may cause recognition problems because the Model may lack essential features.

In this tutorial, we will use the default contour detection and feature selection parameters to build the model, as shown in [Figure 58](#).

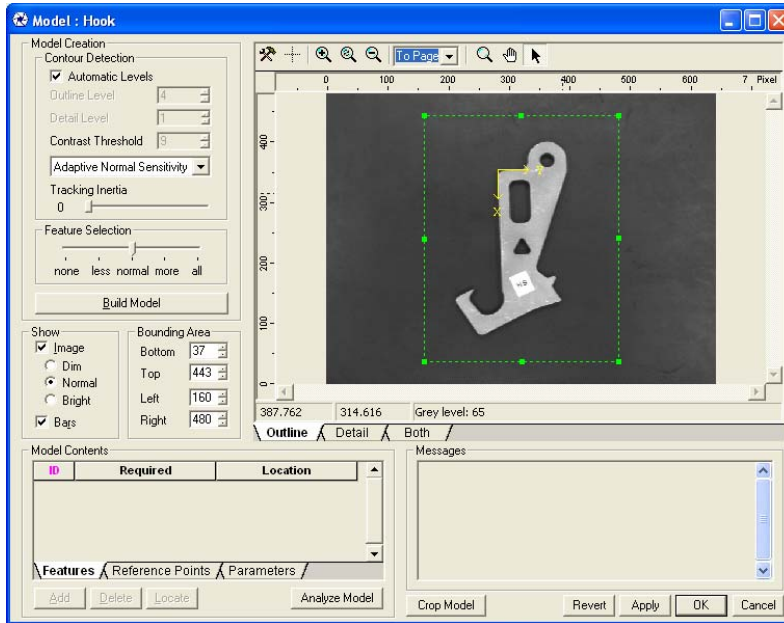


Figure 58 Model Editor Interface

1. In the **Contour Detection** frame, leave the **Automatic Levels** check box enabled, **Tracking Inertia** to 0 and **Contrast Threshold** combo box to **Adaptive Normal Sensitivity**.
2. In the **Feature Selection** frame, leave the **slider** in the default **normal** position.
3. In the display, drag and resize the green dotted rectangle (the Model's bounding box) to enclose the entire object.
4. Click **Build Model**.

The new Model is now created. Contour detection parameters have been optimized and features were selected at both the Outline and Detail levels. To verify that no problem was encountered while building a model, check the **Messages** text box.

You can view and edit the Model in the display. Select the **Outline** tab at the bottom of the display to view the Model at the Outline Level. Outline features are shown as bold magenta lines. Unused contours are shown as thin blue lines. Select the **Detail** tab at the bottom of the display to view Detail Level features. Detail features are shown as bold green lines. As with the Outline Level, unused contours are shown as thin blue lines. The **Both** tab displays both Outline and Detail Levels as superimposed.

Editing Models

You can now use the Model to locate instances of the object. We will however edit it to show how the Model Editor can be used to customize a model for a specific application.

Removing Features from a Model

The application we are building in this tutorial will be used to inspect features on the hook with respect to its outer contour. The steps will be to locate the part using its outer contour and apply tools to inspect the rounded rectangle, the hole and the label. These features need to be inspected because they might be missing on actual parts, be the wrong size, or be mislocated with respect to the outer contour. Because these features are unstable, they should be removed from the model to ensure the best possible accuracy for locating the outer contour of the object.

The first step will be to remove unwanted features at the Outline Level.

1. Click the **Outline** tab at the bottom of the display. It will enable you to edit the Model at the Outline Level. The grid in the **Model Contents** frame displays the **Features** sub-panel for the Outline level of the Model.
2. Click the **Features** tab in the **Model Contents** frame to activate the features grid.

3. Select the rounded rectangle feature by either clicking it in the display or by selecting it in the **Features** grid.
4. Click **Delete** to remove this feature from the Model.

The next step will be to remove unwanted features at the Detail Level.

5. Click the **Detail** tab at the bottom of the display. It will enable you to edit the Model at the Detail Level. The grid in the **Model Contents** frame displays the **Features** sub-panel for the Detail level of the Model.
6. Select all the features, except the outer contour, by clicking them while holding the **Ctrl** key either in the display or in the **Features** grid.
7. Click **Delete** to remove these features from the Model.

The unstable features are now removed from the model. To view the outline and Detail Levels superimposed, click the **Both** tab at the bottom of the display. In this viewing mode, the model is not editable.

Setting the Object Coordinate System for a Model

The part inspection process requires that the measurements of the features be expressed with respect to the outer contour of the part. To fix a reference frame to express these inspection results, you will define an Object coordinate system for the part.

1. Click the **Outline** or the **Detail** tab at the bottom of the display. It will enable you to edit the Object coordinate system.
2. Click the **Coordinate System** tab in the **Model Contents** frame to activate the coordinate system panel.
3. Use the mouse to position the coordinate system marker (yellow axes) over the object, as shown in [Figure 59](#).

To rotate the object coordinate system, click on the arrow of the X or the Y-axis and drag with the mouse.

To move the coordinate system, click the intersection of the X- and Y-axes and drag with the mouse.

You can also drag the arrow of the axes to stretch the marker. Doing so has no effect on the Object coordinate system itself, but can help you align the marker over long features. Fine-tune the final position with the text boxes in the **Coordinate System** panel.

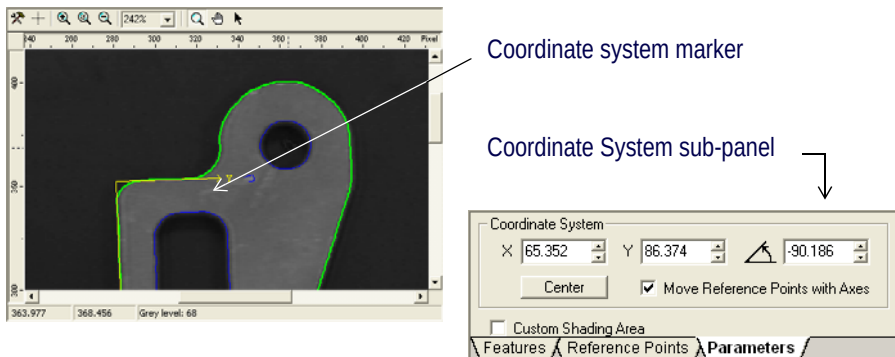


Figure 59 Adjusting the Position and Orientation of the Model Coordinate System

Adding Reference Points to a Model

For some applications, it is sometimes useful to have the position of specific points on the Model transposed in the World coordinate system for each object instance located. To do so without having to compute coordinate transforms manually, reference points can be defined on a Model. In this application, we will add a reference point above the circular hole.

1. Click the **Outline** or the **Detail** tab at the bottom of the display. It will enable you to edit the reference points.
2. Click the **Reference Points** tab in the **Model Contents** frame to activate the reference points panel.
3. Click **Add** to create a reference point in the display.
4. Click and drag the target marker to position the reference point above the circular hole in the part. Zoom in for more precision. Fine-tune the position using the text boxes in the **Reference Points** panel.

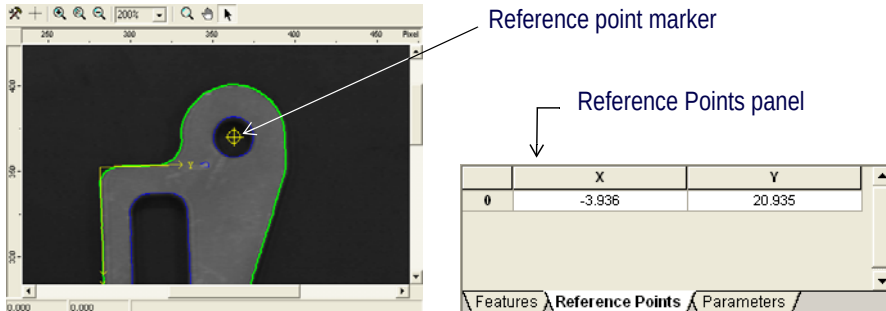


Figure 60 Adding a Reference Point with the Model Editor

This reference point will not be used to locate the part, only contour features are used for locating objects. However, for each instance of the object that will be located, the reference point's position will be returned with respect to the scale, rotation and translation of this instance. You can add as many reference points as you need on a Model.

Manually Adding Features to a Model

Sometimes, it may be useful to manually add a feature to a Model. In Model Editor, you can select features from the unused contours at the Outline Level as well as the Detail Level. To add a feature at the Outline Level, follow the procedure below.

1. Select the **Outline** tab at the bottom of the display to edit the Model at the Outline Level.
2. Click the **Features** tab in the **Model Contents** frame to activate the features grid.
3. Disable the **Image** check box in the **Show** frame to show only the contours. This will make the blue unused contours more visible.
4. In the display, click one of the unused contours, the rounded rectangle for example. The contour turns light blue.
5. Click a starting point on the contour. Hold the **Ctrl** key and click another point on the contour. The selected feature appears in red.

6. To invert your selection, hold the **Shift** key while clicking on the contour. Click **Add** in the **Model Contents** frame. The new feature appears in the features grid.
7. Select the newly added feature in the grid and click **Delete** to remove it from the Model.
8. To add the entire rounded rectangle as a feature, double-click it in the display so the whole contour is selected. Click **Add** to add this feature to the model.
9. Select the newly added feature in the grid and click **Delete** to remove it from the Model, since it is considered unstable for this application.

Analyzing a Model

When a Model has been edited manually, it is important to verify that it will work correctly with the Locator. To do so, click **Analyze Model** in the **Model Contents** frame. Any problem will be displayed in the **Messages** frame.

Saving a Model Database

When a Model is completed, it must be saved in a model database, otherwise it will be lost when you exit the application. To save the Model, follow the procedure below.

1. Click **OK** at the bottom right of the Model Editor. This will transfer the newly created Model to the active model database of the Locator.
2. In the Locator's properties window, select the **Model** tab.
3. Click **Save As...** in the **Database** frame. Save your Model as *HookModel* in your working directory. You will find it convenient to store Models close to other HexSight files used in a same application, such as image emulation files, the Process Manager configuration file, etc.
4. You can return edit your Model at any time by clicking **Edit** in the **Models** frame.

Configuring Search Parameters

Search parameters are used to constraint the search and configure object recognition and positioning algorithms.

1. Select the **Search** tab.
2. Leave the **Nominal Scale** check box enabled. Its value in the text box should be 1.0.
3. Leave the **Nominal Rotation** check box disabled, with the range between -180.0 to 180.0.
4. Leave all the other parameters to their default setting also, as shown in [Figure 61](#).

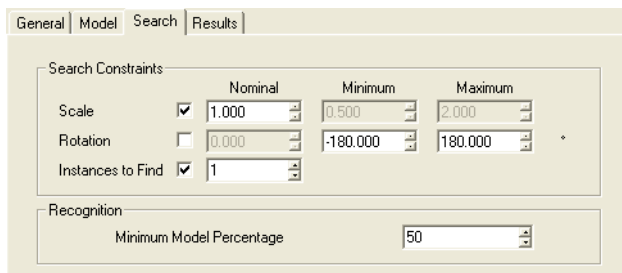


Figure 61 Search Panel of the Locator Tool

Executing the Locator and Verifying the Results

Once the Model is created and the search is configured, the first step is to verify that the Locator finds the object on the image that was used to create the model.

1. Select the **Results** tab.
2. Click **Execute** at the bottom of the Locator's properties window.
3. Verify in the grid that the instance was located correctly.
4. Click **Show Results Viewer** to view the located instance superimposed on the Image, as shown in [Figure 62](#).
5. Move the Locator properties window and the **Results Viewer** to a corner of your screen.

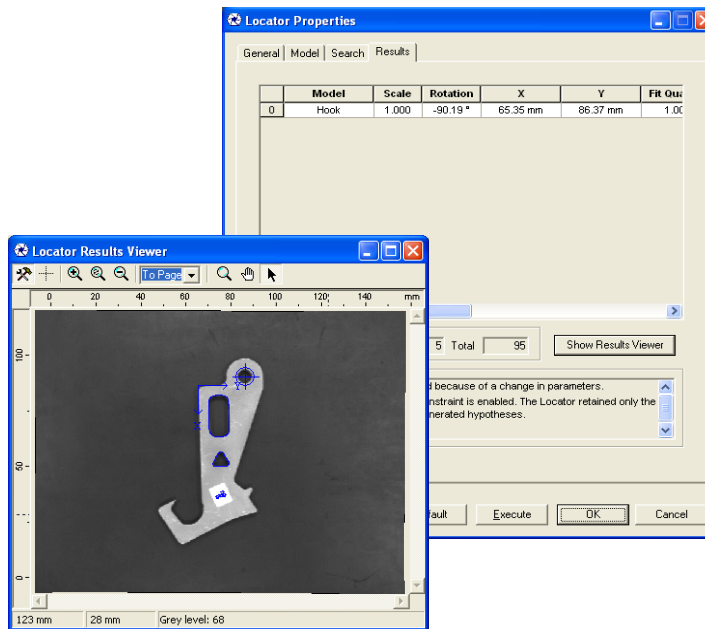


Figure 62 Results Viewer and Results Panel of the Locator Tool

The next step is to verify that the object is located in all the images in the emulation database.

1. In the Process Manager, open the Acquisition Device properties window. Select the **Configuration** panel, then click **Enable All** to enable all the images in the emulation database. Click **OK** to close the Acquisition Device's properties window.
2. Return to the Process Manager and click **Execute Sequence**.
3. In the **Results** panel of the Locator properties window, check if the instance was located correctly. Return to the Process Manager and click **Execute Sequence** several times to apply the Locator to each of the images.
4. Close the Results Viewer window, return to the Locator Properties window and click **OK**.

5. Now is a good time to return to the Visual C++ dialog and save your work. Close the window to exit the Process Manager. To return to the application, you will have to right-click the HexSight icon and select **Edit** (or **Show Application Interface** if the icon is still selected).

Adding Code for the Locator

In this section, you will add code to output the properties of the instance found by the Locator in your application interface.

1. Add this line following the include files already in HookInspectionDlg.cpp:

```
#include "HSLocator.h"
```
2. Locate the **OnButtonInspect** function and insert the lines of code shown in bold:

```
HSProcessManager lManager;
HSLocator      lLocator;

DWORD lStartTime;
DWORD lStopTime;

mInspectButton.EnableWindow( FALSE );

lManager = mApplicationControl.GetProcessManager();
lLocator = lManager.GetProcess( COleVariant( "Locator" ) );
```

This code creates a reference to the Locator process contained in the HexSight Application Control. This reference will be used to access the tool programmatically.

3. In the existing “do” loop of the **OnButtonInspect** procedure, add the lines of code shown in bold:

```
do
{
    lStartTime = GetTickCount();
```



```

// Execute processes in the process list

lManager.Execute( COleVariant( "Acquisition" ),
                  COleVariant( "Locator" ) );

if ( lLocator.GetInstanceCount() > 0 )
{
    CString lBuffer;

    // An instance of the object is found
    // Output the properties of the located instance in
    // text boxes
    mTypeText.SetWindowText(
        lLocator.GetInstanceModelName(0) );
    lBuffer.Format( "%.2f",
        lLocator.GetInstanceScaleFactor(0) );
    mScaleText.SetWindowText( lBuffer );
    lBuffer.Format( "%.2f",
        lLocator.GetInstanceRotation(0) );
    mRotationText.SetWindowText( lBuffer );
    lBuffer.Format( "%.2f",
        lLocator.GetInstanceTranslationX(0) );
    mTranslationXText.SetWindowText( lBuffer );
    lBuffer.Format( "%.2f",
        lLocator.GetInstanceTranslationY(0) );
    mTranslationYText.SetWindowText( lBuffer );
}
else
{
    // No object is found

    mTypeText.SetWindowText( "" );
    mScaleText.SetWindowText( "" );
    mRotationText.SetWindowText( "" );
}

```

```

        mTranslationXText.SetWindowText( "" );
        mTranslationYText.SetWindowText( "" );
    }

    // Compute elapsed time

    lStopTime = GetTickCount();

    UpdateElapsedTime( lStartTime, lStopTime );

    DoEvents();
}
while( mContinuousCheck.GetCheck() == 1 );

```

Each time the loop is executed, this code executes the Application Control's sequence of process from "Acquisition" to "Locator". If no parameter is specified to the Execute method of the Process Manager, the whole sequence of processes will be executed, which in this case will yield the same result. After HexSight processes are executed, the pose (scale, rotation and translation) and type (model name) of the located instance are displayed in the interface.

4. You must add a **UpdateElapsedTime** private member function declared as follows:

```

void
CHookInspectionDlg::UpdateElapsedTime(
    DWORD const pStart,
    DWORD const pStop )
{
    DWORD    lElapsed;
    CString  lBuffer;

    if ( pStop < pStart )
    {

```

```

        lElapsed = (DWORD)(-1) - pStart + pStop + 1;
    }
    else
    {
        lElapsed = pStop - pStart;
    }

    lBuffer.Format( "%lu", lElapsed );
    mTimeText.SetWindowText( lBuffer );
}

```

This function allows bypassing of the **GetTickCount** limit, which is limited to 49.7 days. If you need to know the elapsed time in an industrial application, you must use another solution beside **GetTickCount**.

5. At this moment, your application interface and code should look almost exactly like the example in

InstallationPath\Tutorial\VisualC++\HookInspection\Part1. Save your work and move on to the next step.

Testing the Application

You are ready to test your application.

1. To start the running Mode, press the **F5** key. Click **Inspect** a few times.

The properties of the found instances displayed in the text boxes should be updated every time you press **Inspect**.

2. Enable the **Continuous Mode** check box and click **Inspect**. The application should run in continuous mode. Exit the Continuous Mode by disabling the check box. Exit the Run mode by closing the window or by clicking **Quit**.

This concludes this tutorial. After debugging, save your work and move on to the next tutorial where you will add a display to your application.

Visual C++ Tutorial 2: Display

This second tutorial shows you how to add a Display to your application interface. A Display allows you to view Images and Scenes processed by the HexSight Application Control.

Adding the Display to the Interface

First, you will modify the interface of your application by adding the Display.

1. Start by extending the dialog template. Set the width to approximately 486 and the height to approximately 239. Reorganize the text boxes and buttons to look like the dialog in [Figure 63](#).

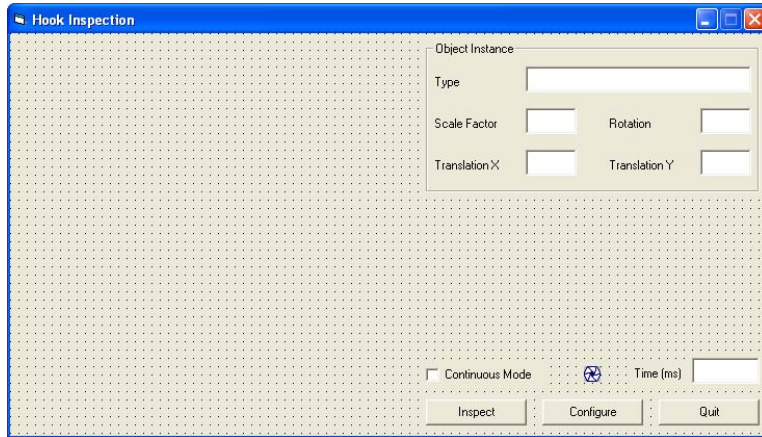


Figure 63 Resized Visual C++ Dialog for Tutorial 2

2. Right-click the dialog and select **Insert ActiveX Control...** Scroll down to find **HSDisplay Control**, select it and click **OK**. The HexSight Display appears on the dialog.
3. Set the width to 253 and height to 225. In its properties, change it ID to **IDC_CONTROL_DISPLAY**.
4. Manually add a **HSDisplay mApplicationDisplay** Member Variable. Do not use the ClassWizard. Follow the same procedure used in Tutorial 1 to manually add a data member for HSAApplication Control.

5. Finally add the **HSDisplay.h** and **HSDisplay.cpp** file to project, provided with HexSight, from the **Wrappers** directory.
6. The interface of the project is now complete and should look like the following dialog in [Figure 64](#).
7. Save your work.

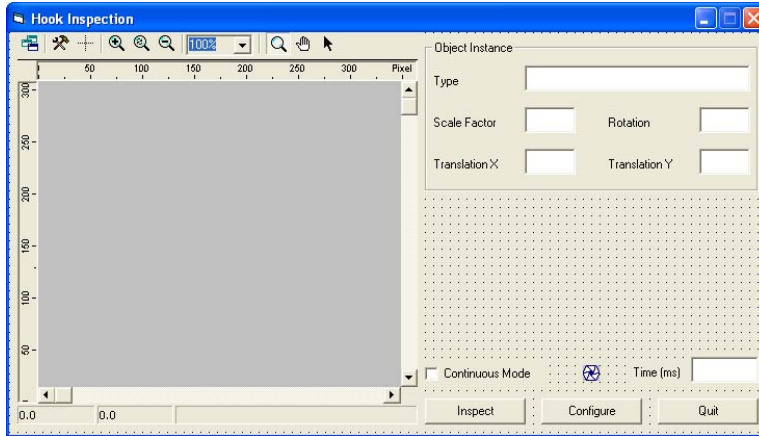


Figure 64 Visual C++ Dialog for Tutorial 2

Adding Code for the Display

Code must now be added to update the Display output every time the inspection loop is executed.

1. Add these lines after the include files already in HookInspectionDlg.cpp:

```
#include "HSAcquisitionDevice.h"
#include "HSDatabase.h"
```

2. Add these wrappers (.h and.cpp) in your project:

- HSAcquisitionDevice
- HSCalibrationDistortionInterface
- HSCalibrationPerspectiveInterface
- HSCalibrationXYScaleInterface

3. Insert the lines of code shown in bold into the **OnInitDialog** function:

```

SetIcon(m_hIcon, TRUE);    // Set big icon
SetIcon(m_hIcon, FALSE);  // Set small icon

HSAcquisitionDevice lAcquisition =
    mApplicationControl.GetProcessManager().GetProcess(_
        ColeVariant( "Acquisition" ) );

// Inits the Application Display

mApplicationDisplay.SetImageDatabase ( _
    0, mApplicationControl.GetDatabase().GetHandle() );

if ( lAcquisition.IsValid() )
{
    mApplicationDisplay.SetImageViewName( _
        0, lAcquisition.GetOutputView( HEXSIGHT_NONE ) );

    mApplicationDisplay.SetImageName( _
        0, lAcquisition.GetOutputImage( HEXSIGHT_NONE ) );
}

mApplicationDisplay.SetZoom( -1 );

return TRUE;    // return TRUE unless you set the focus to
                // a control

```

This initialization code configures the Display Control to display the Image provided by the Acquisition Device output. It also sets the zoom to page mode, which automatically adjusts the zoom factor to view the entire image. You must always verify the validity of the ActiveX objects with **IsValid()**. If the object is not valid, you will get random results if you try to use it. In this tutorial, we have omitted the validation of controls in order to simplify the code.

4. In the **OnButtonInspect** function, insert the lines of code shown in bold:

```

lStartTime = GetTickCount();

mApplicationDisplay.RemoveAllMarker();
...

UpdateElapsedTime( lStartTime, lStopTime );

mApplicationDisplay.Refresh();

DoEvents();
...

```

The `RemoveAllMarker` call removes all overlay graphic markers that could have been added to the previous image. The refresh call updates the display to show the new Image.

5. Save your work and test the application. Now along with the properties of the found instance, you should see an image of the object for the current instance.

Visual C++ Tutorial 3: Caliper

In this tutorial, you will learn how to set up, configure and use a Caliper tool to precisely measure the distance between parallel edges on an object.

Modifying your Application Dialog

First you need to add two edit boxes to your application for the interface to display the measurement output by the Caliper.

1. On your application dialog, add two edit boxes. Name the first one **IDC_EDIT_WIDTH** and the second **IDC_EDIT_HEIGHT**. Also add a group box and proper labels for your text boxes. Set everything to appear as illustrated in [Figure 65](#).
2. Add a member variable for each new edit box. Name them **mWidthText** and **mHeightText**. They must be of type **CEdit**

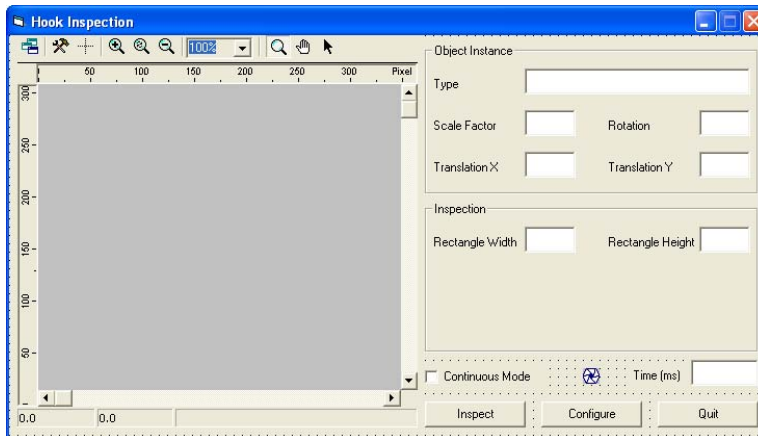


Figure 65 Visual C++ Dialog for Tutorial 3

Placing the Caliper

Right-click the HexSight Application Control icon on the dialog and select **Edit** from the context menu. Click **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add

an **HSCaliper** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *WidthCaliper* as the **Process Name** (instead of HSCaliper2).
2. Select Grey-Scale Image from Acquisition View as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select Hook from Model View as the **Input Model Scene**.
5. Leave the **Lock Location Transform on a Selected Instance** check box enabled.
6. For the **Input Instance Scene**, select Instance Scene from Locator View.

The above settings will make your application fully model-based (automatic model based mode). The Caliper tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Caliper is applied to an area on the object bounded by the Caliper's Rectangle, which appears in green. The Caliper will be used to measure the width of the rectangle-shaped hole on the part.

8. Place the Rectangle on the part as shown in [Figure 66](#).
9. Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees (the Y-axis is used to adjust the skew).
10. Click the center of the Rectangle and drag it into position.
11. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position and the orientation with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

Note: The Caliper must always be as perpendicular as possible to the edges to be measured. For the current example, the tool had to be rotated to 90 degrees and the skew was left at its default value of 0 degrees.

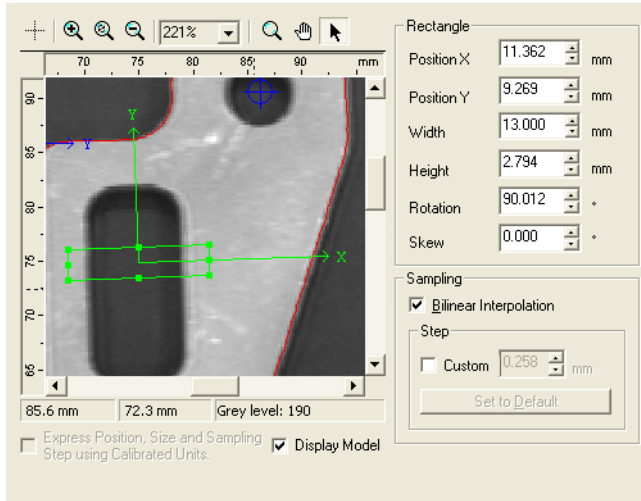


Figure 66 Adjusting the Size, Position and Orientation of the Caliper Rectangle

- Go to the **Transform** panel. The image shown is that of the current instance of the object, as located in the second image of the emulation database (if you clicked **Execute Sequence** twice before creating the Caliper tool as indicated previously). To see the tool applied to other instances, leave the Caliper window open and click **Execute Sequence** in the Process Manager. Yellow lines indicate where the Caliper has detected potential edges. Edge selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

Configuring the Caliper

The Caliper locates potential edge pairs. You will set up constraints that will allow the Caliper to measure only valid edge pairs, that is, those that meet the criteria you have determined.

The display in the **Configuration** panel represents the part of the Image bounded by the Rectangle. The Caliper processes information within the Image along pixel lines, called projection paths, parallel to the Rectangle's Y-axis. In the display you can observe the following:

- Dotted yellow lines indicate potential edges.
- The blue curve shows a graphical representation of a projection that averages each column of pixels in the Rectangle to create a one-dimensional signal. Sharp changes on this curve indicate potential edges.
- The green magnitude curve is the first derivative of the projection curve. High and low peaks in this curve identify potential edges.
- Horizontal red lines indicate the magnitude threshold for the detection of potential edges.

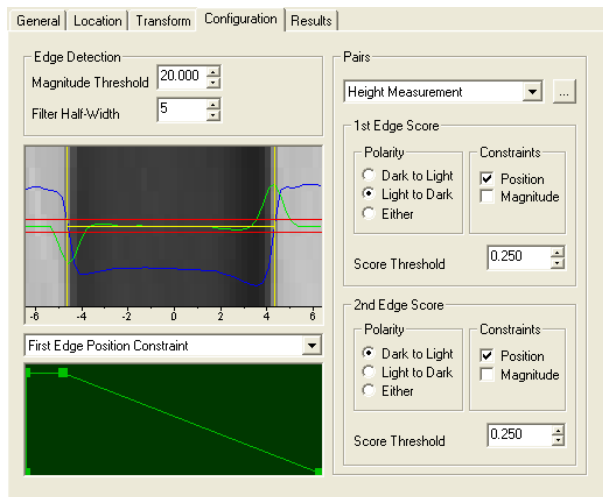


Figure 67 Caliper Tool — Configuration Panel

In this example, only one measurement pair is needed so the default pair named *Pair0* will be used. The constraints used for both edges will be polarity and position constraints. These constraints will be used to score each edge.

Polarity indicates changes in light values, from left to right, within the Rectangle.

Position constraints score each edge according to its position as defined by the Position Constraint histogram. **Magnitude** constraints, not used in this tutorial, score each edge according to its magnitude as defined by the Magnitude Constraint histogram.

The Caliper attributes a score to each edge pair and to each individual edge. Scores range from 0 to 1 with the minimum acceptable score being determined by the **Score Threshold**, set by default at 0.250. Histogram functions for each constraint determine how the Caliper scores edges.

1. In the **1st Edge Score** frame, select **Polarity** as **Light to Dark** and enable the **Position** check box.
2. In the **2nd Edge Score** frame, select **Polarity** as **Dark to Light** and enable the **Position** check box.

Next you must set up the Position constraint histogram for each edge, in the bottom left window of the **Configuration** panel.

3. Use the combo box above the histogram window to select a constraint.
4. Use the mouse to drag the handles in the histogram window. Set the 1st and the 2nd edge position constraints as illustrated in [Figure 68](#).
5. Click **Execute**. The yellow dotted lines become solid and are joined by a solid horizontal line to indicate that the pair has been detected and measured.

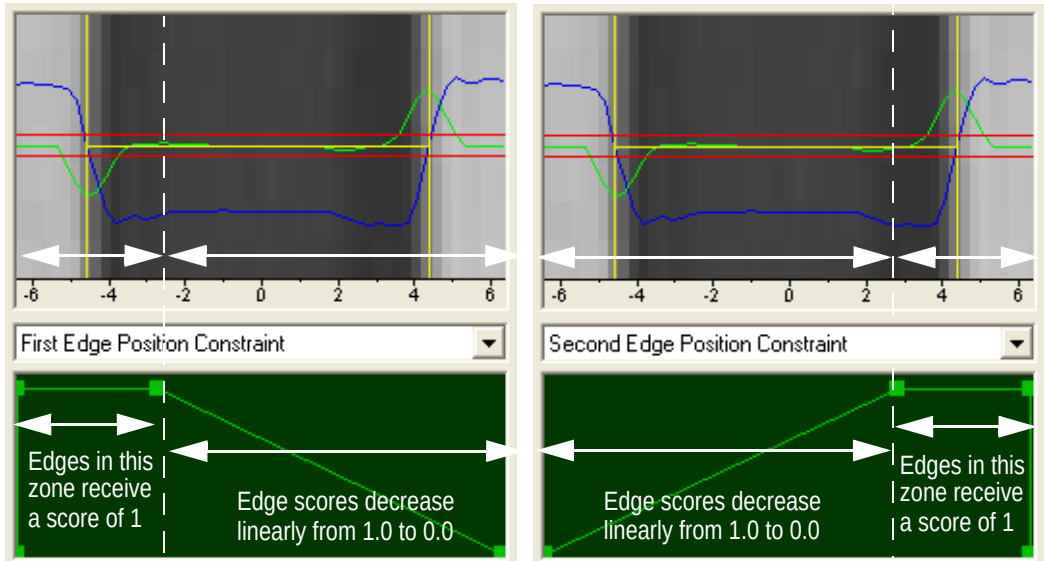


Figure 68 Adjusting Position Constraint Histograms

Caliper Results

You will now observe the Caliper's results for various instances of the object.

Access the Caliper **Results** panel to observe the following:

- Valid edges are displayed as parallel vertical yellow lines.
- The Caliper measure is displayed as a horizontal yellow line, perpendicular to the edges.
- The Pair Count box indicates that the Caliper was configured to measure one pair.
- The Coordinate System combo box indicates that the positions are expressed with respect to the Tool coordinate system (the origin centered on the tool's Rectangle).

The grid of results indicates:

- The Caliper measure, named Size, indicated in calibrated units.

- The score for the edge pair as well as for each edge.
- Use the scrollbar to view the entire grid of results.

Now that the Caliper is properly configured to measure the rectangular hole feature, you will want to test and observe the Caliper on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the next step.

Adding Code for the Caliper

You will finish up by adding code to your Visual C++ class that will display the measurement of the Caliper in the interface. You will also add a line marker on the display to show the measurement.

1. Add the following lines after the include file of the HookInspectionDlg.cpp file:

```
#include "HSCaliper.h"
```
2. Add this wrapper (.h and.cpp) in your project:
 - HSCaliper
3. Edit the **OnButtonInspect** function. Insert the lines of code shown in bold:

```
HSLocator lLocator;
HSCaliper lWidthCaliper;
CString lMarkerName;
...
lLocator = lManager.GetProcess( COleVariant( "Locator" ) );
lWidthCaliper = lManager.GetProcess(
    COleVariant ( "WidthCaliper" ) );
```

This will create a reference that will be used for programmatic access to the WidthCaliper process instantiated in the HexSight Application Control.

4. Modify the existing line, shown in *italic*, for the one below it:

```
lManager.Execute( COleVariant( "Acquisition" ),
    COleVariant( "Locator" ) );
```

```
lManager.Execute( COleVariant( "Acquisition" ),
                  COleVariant( "WidthCaliper" ) );
```

This will execute the entire sequence of processes.

- Now, to display the Caliper's measurement in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```
mTranslationYText.SetWindowText( lBuffer );

// Display width caliper properties and marker

if ( lWidthCaliper.GetPairScore( COleVariant(0L) ) > 0 )
{
    lWidthCaliper.SetCoordinateSystem(
        HSCaliper::hsCoordinateSystemWorld );
    lBuffer.Format( "%.2f",
        lWidthCaliper.GetPairSize( COleVariant(0L) ) );
mWidthText.SetWindowText( lBuffer );

    lMarkerName = "Width";
mApplicationDisplay.AddLineMarker( lMarkerName,
        lWidthCaliper.GetEdge1PositionX( COleVariant(0L) ),
        lWidthCaliper.GetEdge1PositionY( COleVariant(0L) ),
        lWidthCaliper.GetEdge2PositionX( COleVariant(0L) ),
        lWidthCaliper.GetEdge2PositionY( COleVariant(0L) ),
        TRUE );
mApplicationDisplay.SetMarkerAnchorStyle( lMarkerName,
        HSDisplay::hsCross );
mApplicationDisplay.SetLineMarkerConstraints(
        lMarkerName,
        HSDisplay::hsLineNoEdit );
mApplicationDisplay.SetMarkerColor( lMarkerName,
        HSDisplay::hsRed );
mApplicationDisplay.SetMarkerDisplayName( lMarkerName,
```

```

        TRUE );
    }

```

If the first pair of the Caliper (Index = 0) is found (Score > 0.250), its measurement is displayed in the WidthText text box. Using the World coordinate system, the calibrated X-Y position of the first and second edge of the pair is used to draw a non-editable line marker on the application display.

6. In case no instance is found, add lines of code shown in bold:

```

mTranslationYText.SetWindowText( "" );
mWidthText.SetWindowText( "" );
mHeightText.SetWindowText( "" );

```

7. Save your work and test the application. You should see the measurement in the RectangleWidth edit box of your application. You should also see the line marker you added on the Display highlighting the measurement of WidthCaliper.

Adding a Second Caliper

Using the previous steps, try adding a second Caliper that will measure the height of the rectangular hole in the hook. Add the appropriate code to display the result in the remaining text box (Rectangle Height) on your application form. Also add the code to draw another line marker on the Display.

Refer to the Application Example in the following folder:

InstallationPath\Tutorial\VisualC++\HookInspection\Part3.

Visual C++ Tutorial 4: Blob Analyzer

In this tutorial, you will learn how to set up, configure and use a Blob Analyzer tool to find, label and analyze irregular shaped features on the part.

Modifying your Application Form

First you need to add two edit boxes to your application so you can observe the measurement output by the Blob Analyzer.

1. On your application dialog, add two edit boxes. Name the first one **IDC_EDIT_DIAMETER** and the second **IDC_EDIT_OFFSET**. Add proper labels for your edit boxes. Set everything to appear as illustrated [Figure 69](#).
2. Add a member variable for the new edit boxes. Name them **mDiameterText** and **mOffsetText**. They must be of type CEdit.

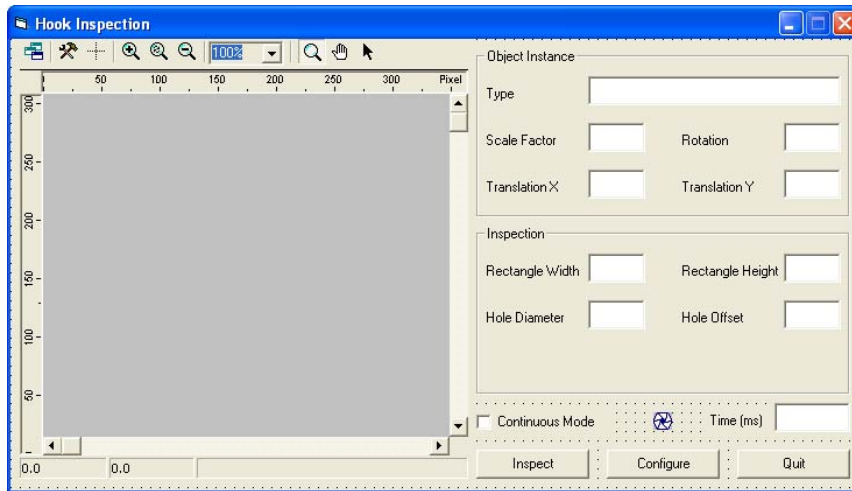


Figure 69 Visual C++ Dialog for Tutorial 4

Placing the Blob Analyzer

Return to the Process Manager (right-click the HexSight Application Control icon, select **HSApplication Control Object>Edit**). Click **Execute Sequence**

twice to locate the object in the second image of the emulation database. Using the Process Manager, add a **HSBlobAnalyzer** process from the **HexSight Inspection Tools** folder of the Process Manager. Set the tool's General panel as follows:

1. Enter *HoleBlob* as the **Process Name** (instead of HSBlobAnalyzer4).
2. Select Grey-Scale Image from Acquisition View as the **Input Image**.
3. Leave the **Create Blob Image** check box disabled.
4. Enable the **Apply Transform to a Model-defined Location** check box.
5. Select Hook from Model View as the **Input Model Scene**.
6. Enable the **Lock Location Transform on a Selected Instance** check box.
7. Select Instance Scene from Locator View as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Blob Analyzer tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

8. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Blob Analyzer will be applied to an area on the object bounded by the Blob Analyzer's Rectangle, which appears in green. The Blob Analyzer will be used to inspect the diameter and offset of the circular hole on the part.

9. Place the Rectangle on the part as shown in the illustration. Click the center of the Rectangle and drag it into position. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

10. Go to the **Transform** panel. The image shown is that of the current instance of the object as located in the second image of the emulation

database (if you clicked twice on **Execute Sequence** before creating the Blob Analyzer tool). The red target marker indicates the center of the blobs detected by the Blob Analyzer. Blob selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

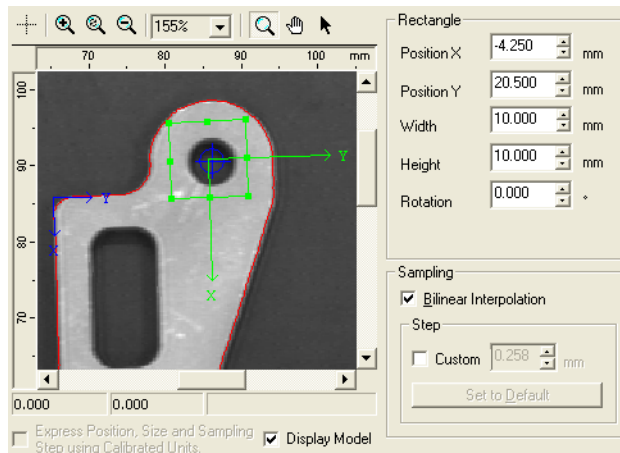


Figure 70 Adjusting the Size, Position and Orientation of the Blob Analyzer Rectangle

Configuring the Blob Analyzer

You will set up constraints that will allow the Blob Analyzer to find only valid blobs, that is, those that meet the criteria you have determined.

The display in the **Configuration** panel represents the part of the Image bounded by the Rectangle. The Blob Analyzer processes information within this Rectangle. In the display you can observe the following:

- The blob, as defined by the Image Segmentation Mode, is represented in green.
- A blob marker appears in red in the center of each blob.
- When displaying the Pixel Weight Image, you can see the weight value of each pixel as defined by the graphical function of the Image Segmentation box. White pixels have a weight of 1 while black pixels

have a weight of 0. Pixels having a weight between 0 and 1 are displayed with a proportional greylevel.

1. First, set the Blob Constraints: **Minimum Area** to 0 and **Maximum Area** to 100, which corresponds to the area defined by the Rectangle.

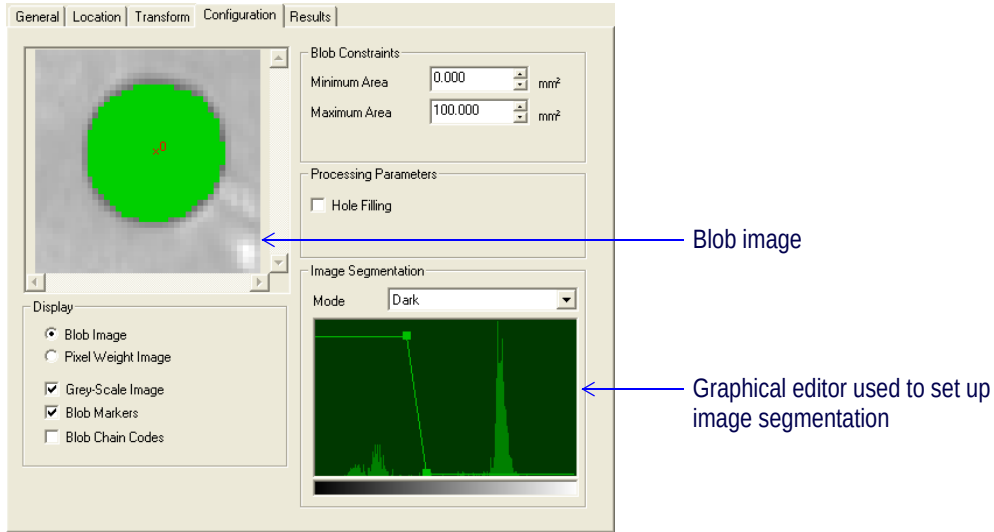


Figure 71 Blob Analyzer — Configuration Panel

Now you must set the segmentation function using the graphical editor in the Image Segmentation box. The graphic shows a histogram of the distribution of pixels inside the Blob Analyzer's Rectangle. Black (0) is at the far left and white (255) at the far right. In this example, there is a small peak in the black portion (around 58) representing the blob, and a higher peak (around 186) representing the part color. You want to include some pixels in the transition, with a weight function proportional to the grey-scale value.

2. Since we are looking for a dark blob on a white background, set the **Mode** combo box to **Dark**.
3. Use the mouse to drag the handles in the graphic function editor. Set the Image Segmentation weight function as illustrated in [Figure 72](#).
4. Click **Execute**. The segmented blob now corresponds to the hole in the part.

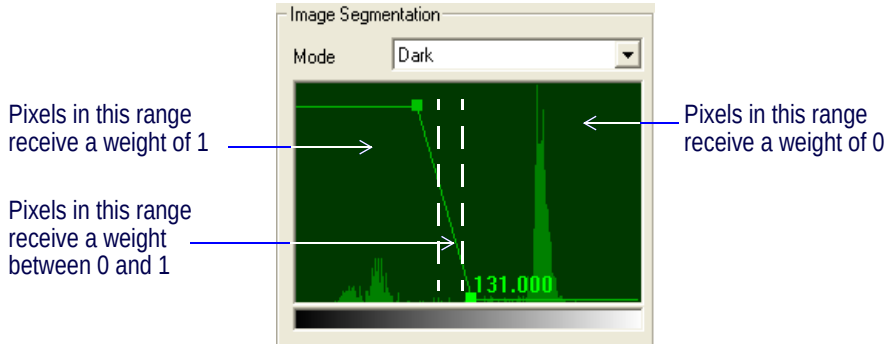


Figure 72 Image Segmentation Function Editor

Blob Analyzer Results

You will now observe the Blob Analyzer's results for various instances of the object.

1. In the **Results** tab, enable the **Sort Results** check box and select **Area** with the **Sort by** combo box.
2. Click **Execute**.

Use the Blob Analyzer Results panel to observe the following:

- The Blob Count box indicates that the Blob Analyzer has found one blob.
- An image is displayed with a blob marker indicating the center of each blob found for this instance of the object.
- The Coordinate System combo box selection indicates that the positions are returned with respect to the Tool coordinate system (the origin centered on the tool's Rectangle).
- A list of the selected types of blob properties that are computed: General, Perimeter, Intrinsic Inertia, etc.

The grid of results shows:

- The position of the blob according to the selected coordinate system

- The blob area indicated in calibrated units
- The properties computed for the blob. To view the complete set of properties, use the grid's tabs and scrollbar.

Now that the Blob Analyzer is properly configured to inspect the circular hole feature, you will want to test and observe the Blob Analyzer on other instances of the object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the final step.

Adding Code for the Blob Analyzer

You will finish up by adding code to your Visual C++ class that will display the measurements of the Blob Analyzer in the interface. You will also add a target marker on the display to show the position and diameter of the hole.

1. Add these include files following the others in the HookInspectionDlg.cpp file:

```
#include "HSBlobAnalyzer.h"
#include <math.h>
```

2. Add the following wrapper (.h and.cpp) in your project:

- HSBlobAnalyzer

3. Insert the lines of code shown in bold into the **OnButtonInspect** function:

```
HSCaliper      lHeightCaliper;
HSCaliper      lWidthCaliper;
HSBlobAnalyzer  lHoleBlob;
CString        lMarkerName;
double         lHoleDiameter;
double         lHoleOffset;
...
lWidthCaliper = lManager.GetProcess( COleVariant(
"WidthCaliper" ) );
lHeightCaliper = lManager.GetProcess( COleVariant(
"HeightCaliper" ) );
```

```
lHoleBlob = lManager.GetProcess( COleVariant( "HoleBlob" )
);
```

This will create a reference that will be used for programmatic access to the hole blob process instantiated in the HexSight Application Control.

4. Modify the existing line, shown in *italic*, for the one below it:

```
lManager.Execute( COleVariant( "Acquisition" ),
COleVariant ( "HeightCaliper" ) );
lManager.Execute( COleVariant( "Acquisition" ),
COleVariant( "HoleBlob" ) );
```

This will execute the entire sequence of processes.

5. Now to display the Blob Analyzer's measurements in the text boxes and to draw the marker on the application display, insert the lines of code shown in **bold**:

```
if ( lHeightCaliper.GetPairScore( sVarZero ) > 0 )
{
    ...
}

// Display hole blob properties and marker

if ( lHoleBlob.GetBlobCount() > 0 )
{
    lHoleDiameter = sqrt( 4 * lHoleBlob.GetBlobArea(0) /
3.14159 );
    lBuffer.Format( "%.2f", lHoleDiameter );
    mDiameterText.SetWindowText( lBuffer );

    lHoleBlob.SetCoordinateSystem(
HSBlobAnalyzer::hsCoordinateSystemObject );
    lHoleOffset = sqrt(
        lHoleBlob.GetBlobPositionX(0) *
        lHoleBlob.GetBlobPositionX(0) +
```

```

        lHoleBlob.GetBlobPositionY(0) *
        lHoleBlob.GetBlobPositionY(0) );
    lBuffer.Format("%.2f", lHoleOffset);
    mOffsetText.SetWindowText( lBuffer );

    lHoleBlob.SetCoordinateSystem(
    HSBlobAnalyzer::hsCoordinateSystemWorld );
    lMarkerName = "Hole";
    mApplicationDisplay.AddTargetMarker( lMarkerName,
        lHoleBlob.GetBlobPositionX(0),
        lHoleBlob.GetBlobPositionY(0),
        (float)(lHoleDiameter / 2), TRUE );
    mApplicationDisplay.SetTargetMarkerConstraints(
        lMarkerName, HSDisplay::hsTargetNoEdit );
    mApplicationDisplay.SetMarkerColor( lMarkerName,
        HSDisplay::hsRed );
}

```

6. In case no instance is found, add the lines of code shown in bold:

```

    mHeightText.SetWindowText( "" );
mDiameterText.SetWindowText( "" );
mOffsetText.SetWindowText( "" );

```

If the blob is found, the offset of the hole from the origin of the Object coordinate system and the diameter of the hole are computed and displayed in the OffsetText and DiameterText text boxes. Using the World coordinate system, the calibrated X-Y position of the blob is used to draw a non-editable target marker on the application display.

7. Save your work and test the application. You should see the computed diameter and offset of the hole in the text boxes of your application. You should also see the target marker you added on the Display, highlighting the position and the diameter of the hole.

Visual C++ Tutorial 5: Pattern Locator

In this tutorial, you will learn how to set up, configure and use a Pattern Locator tool to find and locate instances of the “HS” printed on the part.

Modifying your Application Form

First you need to add one edit box to your application so you can display the result output by the Pattern Locator.

1. On your application dialog, add one edit box. Name it **IDC_EDIT_LABEL** and add an appropriate label. Set everything to appear as illustrated in Figure 73.
2. Add a member variable named **mLabelText** of type CEdit.

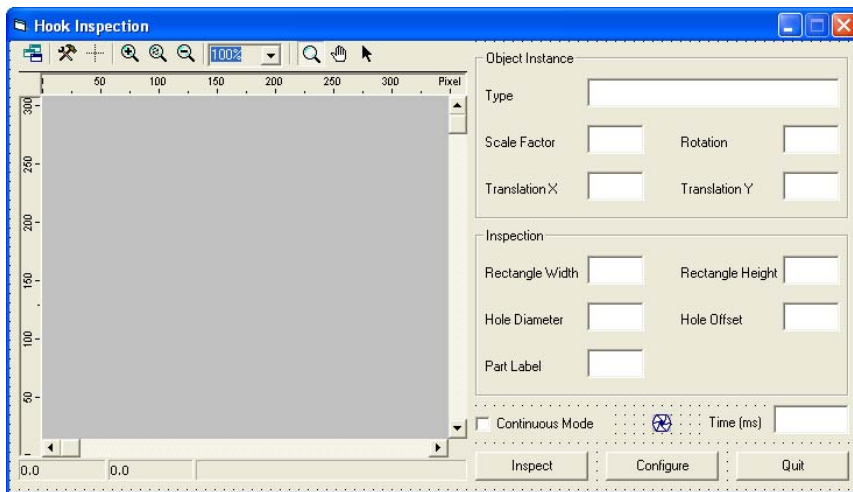


Figure 73 Visual C++ Dialog for Tutorial 5

Placing the Pattern Locator

Select the HexSight Application Control on the dialog. With the right-click menu, select **HSApplication Control Object>Edit**. Click on **Execute Sequence** twice to locate the object in the second Image of the emulation database. Using the Process Manager, add the **HSPatternLocator** process

from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *LabelLocator* as the **Process Name** (instead of *HSPatternLocator5*).
2. Select Grey-Scale Image from Acquisition View as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select Hook from Model View as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select Instance Scene from Locator View as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Pattern Locator tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Pattern Locator will be applied to an area on the object bounded by the tool's Rectangle. The Pattern Locator will be used to verify the presence of the "HS" label printed on the part.

8. Place the Rectangle on the part as shown in the illustration. Click the center of the Rectangle and drag it to the center of the "HS" label. Use the corner and side handles to adjust the width and height of the Search Rectangle to fit the label. Zoom in on the label and by clicking and dragging the arrow on the tool's X-axis, adjust the rotation to fit the orientation of the label. Readjust the green dotted search Rectangle to leave a small oversize around the pattern to provide for some misalignment of the label on some parts. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.

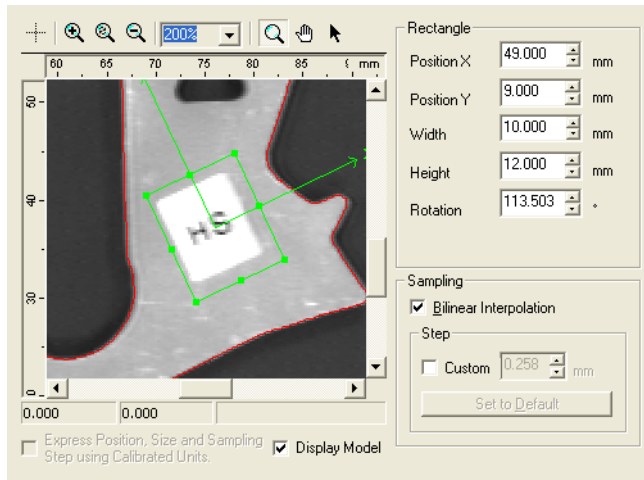


Figure 74 Adjusting the Size, Position and Orientation of the Location Rectangle

9. Go to the **Transform** panel. The Image shown is that of the current instance of the object as located in the second Image of the emulation database (if you clicked twice on **Execute Sequence** before creating the Pattern Locator tool). To see the tool applied to the other instances, leave the Pattern Locator open and click **Execute Sequence** in the Process Manager. The Pattern Locator will be configured to check for the presence of the label in the next step. Open the **Pattern** panel and move on to the next step.

Placing the Pattern Rectangle

You will set up the rectangle for defining the pattern that you want the Pattern Locator find within the Image. In this tutorial case, the Pattern Locator will be used to verify the presence of the "HS" label printed on the part.

1. Place the Rectangle on the part as shown in the illustration.
2. Click the center of the rectangle and drag it to the center of the "HS" label. Use the corner and side handles to adjust the width and height of the Pattern Rectangle to fit the label.

3. Zoom in on the label and by clicking and dragging the arrow on the tool's X-axis, adjust the rotation to fit the orientation of the label.
4. Readjust the size of the Pattern Rectangle to include only the "HS" characters in the pattern. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.
5. Open **Configuration** panel and move on to the next step.

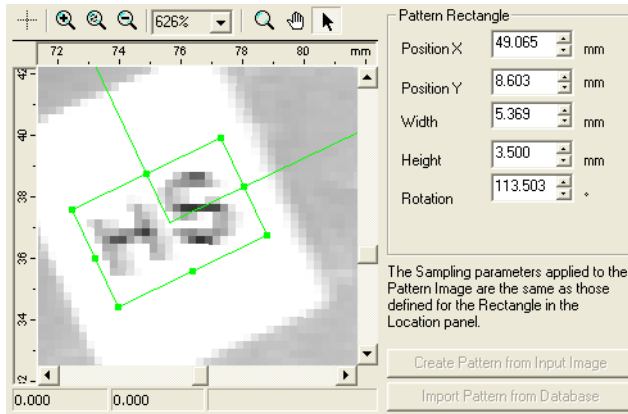


Figure 75 Adjusting the Size, Position and Orientation of the Pattern Rectangle

Configuring the Pattern Locator

You will set up the parameters of the tool in order to detect the grey-scale pattern on the part. Select the Configuration tab and set the following parameters:

1. Set the **Instance To Find** value to 1. This sets the maximum number of pattern instances to be found.
2. Set the **Match Threshold** value to 0.500. This sets the minimum strength score required to accept a valid match
3. Leave the **Automatic Coarseness Selection** check box enabled. The Pattern Locator will automatically determines the **Search Coarseness** and **Positioning Coarseness** levels.

4. Click **Execute**.

Pattern Locator Results

You will now observe the Pattern Locator's results for various instances of the object. Select the **Results** tab to observe the following:

- Each valid pattern is displayed with a blue rectangle marker indicating the position of the pattern detected in the Search Rectangle.
- The left display shows the Image of the Search Rectangle.
- The right display shows the Image of the reference pattern.

The Coordinate system combo indicates that the positions are returned with respect to the Tool coordinate system (the origin centered on the Pattern Rectangle).

The grid of results indicates:

- The strength of the match.
- The position of the pattern according to the selected coordinate system.
- The rotation of the pattern in calibrated rotation units.
- Use the scrollbar to view the entire grid of results.

Now that the Pattern Locator is configured properly to check the presence of the "HS" label, you will want to test and observe the Pattern Locator on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the final step.

Adding Code for the Pattern Locator

You will finish up by adding code to your Visual C++ class that will display the result of the pattern search in the interface. You will also add a point marker on the display to show the position of the label located by the Pattern Locator.

1. Add the following include file after the others in the HookInspectDlg.cpp file:

```
#include "HSPatternLocator.h"
```

2. Add this wrapper (.h and .cpp) in your project:
HSPatternLocator
3. Insert the lines of code shown in bold into the **OnButtonInspect** function:

```

HSBlobAnalyzer lHoleBlob;
HSPatternLocator lLabelLocator;
...
lHoleBlob = lManager.GetProcess( COleVariant( "HoleBlob" )
);
lLabelLocator = lManager.GetProcess( COleVariant(
"LabelLocator" ) );

```

This will create a reference that will be used for programmatic access to the LabelLocator process instantiated in the HexSight Application Control.

4. Modify the existing line, shown in italic, for the one below it:

```

lManager.Execute( COleVariant( "Acquisition" ),
COleVariant( "HoleBlob" ) );
lManager.Execute( COleVariant( "Acquisition" ),
COleVariant( "LabelLocator" ) );

```

This will execute the entire sequence of processes.

5. Now to display the result of the search in the edit box and to draw the marker on the application display, add the lines of code shown in bold:

```

if ( lHoleBlob.GetBlobCount() > 0 )
{
...
}

// Display label locator properties and marker

if ( lLabelLocator.GetMatchCount() > 0 )
{
    mLabelText.SetWindowText( "Present" );

```

```

lLabelLocator.SetCoordinateSystem(
HSPatternLocator::hsCoordinateSystemWorld );
lMarkerName = "HSLabel";
mApplicationDisplay.AddPointMarker( lMarkerName,
    lLabelLocator.GetMatchPositionX(0),
    lLabelLocator.GetMatchPositionY(0), TRUE );
mApplicationDisplay.SetMarkerDisplayName( lMarkerName,
TRUE );
mApplicationDisplay.SetMarkerColor( lMarkerName,
    HSDisplay::hsBlue );
}
else
{
    mLabelText.SetWindowText( "Absent" );
}

```

6. In case no instance is found, add the line of code shown in bold after the existing one:

```

mOffsetText.SetWindowText( "" );
mLabelText.SetWindowText( "" );

```

7. Save your work and test the application. You should see the presence/absence of the “HS” label in the text box of your application. You should also see the point marker you added on the Display, highlighting the position of the label.

Visual C++ Tutorial 6: Edge Locator

In this tutorial, you will learn how to set up, configure and use two Edge Locator tools to measure non-parallel edges on the part.

Modifying your Application Form

First you need to add one edit box to your application so you can observe the result output by the Edge Locator processes.

1. On your application dialog, add one edit box. Name it **IDC_EDIT_PART_WIDTH**. Add a proper label for your edit box. Set everything to appear as illustrated in [Figure 76](#).
2. Add a member variable named **mPartWidthText** of type CEdit for this edit box.

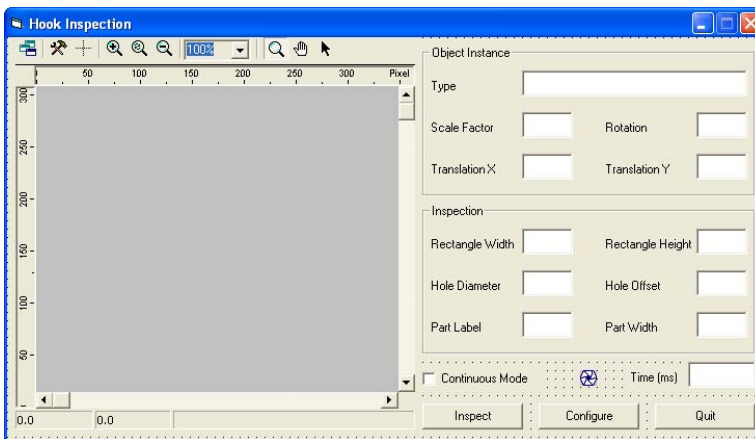


Figure 76 Visual C++ Dialog for Tutorial 6

Placing the Edge Locator

Select the HexSight Application Control on the dialog. With the right-click menu select **HSApplication Control Object>Edit**. Click **Execute Sequence** twice to locate the object in the second Image of the emulation database. Using the Process Manager, add the **HSEdgeLocator** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *PartLeftEdge* as the **Process Name** (instead of HSEdgeLocator6).
2. Select Grey-Scale Image from Acquisition View as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select Hook from Model View as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select Instance Scene from Locator View as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Edge Locator tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Edge Locator will be applied to an area on the object bounded by the tool's Rectangle, which appears in green. The Edge Locator will be used to measure the position of the left edge of the part.

8. Place the Rectangle on the part as shown in the illustration. Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees (the Y-axis is used to adjust the skew). Click the center of the Rectangle and drag it into position. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.

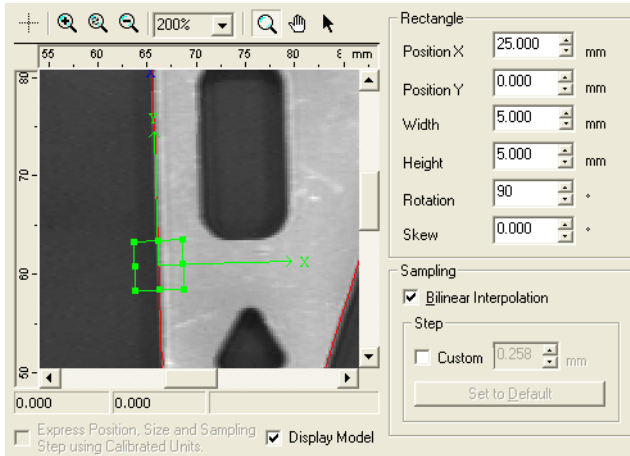


Figure 77 Adjusting the Size, Position and Orientation of the Edge Locator Rectangle

Note: The Edge Locator must always be as perpendicular as possible to the edges to be measured. For the current example, the tool had to be rotated to 90 degrees and the skew was left to its default value of 0 degree.

9. Go to the **Transform** panel. The image shown is that of the current instance of the object with the tool transposed to the correct position on this instance. To see the tool applied to other instances, leave the Edge Locator window open and click **Execute Sequence** in the Process Manager. Yellow lines indicate where the Edge Locator has detected potential edges. Edge selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

Configuring the Edge Locator

The Edge Locator detects potential edges. You will set up constraints that will allow the Edge Locator to locate only valid edges, that is, those that meet the criteria you have determined.

The graphic display in the Configuration panel represents the image information contained in the Edge Locator Rectangle:

- Yellow lines indicate detected edges.

- The blue curve shows a graphical representation of a projection that averages each column of pixels in the Rectangle to create a one-dimensional signal. Sharp changes on this curve indicate potential edges.
- The green magnitude curve is the first derivative of the projection curve. High and low peaks in this curve identify potential edges.
- Horizontal red lines indicate the magnitude threshold for the detection of potential edges.

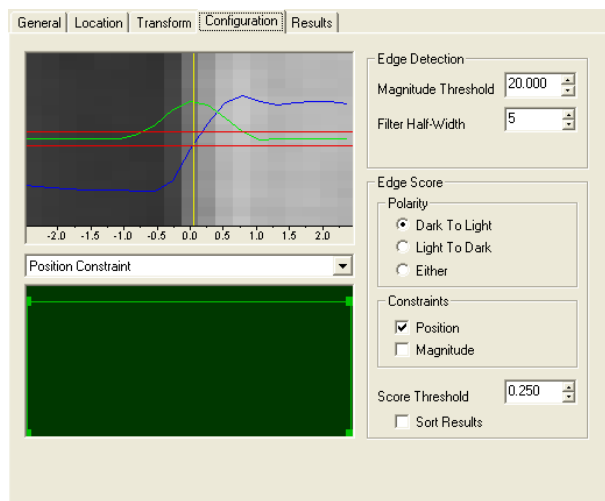


Figure 78 Edge Locator — Configuration Panel

In this example, the constraints used to score edges will be polarity and position constraints.

Polarity indicates changes in light values, from left to right, within the Rectangle.

Position constraints score each edge according to its position as defined by the Position Constraint histogram. **Magnitude** constraints, not used in this tutorial, score each edge according to its magnitude as defined by the Magnitude Constraint histogram.

The Edge Locator attributes a score to each individual edge. Scores range from 0 to 1 with the minimum acceptable score being determined by the **Score Threshold**, set by default at 0.250. Histogram functions for each constraint determine how the Edge Locator scores edges.

1. In the **Edge Score** box, select **Polarity** as **Dark to Light** and enable the **Position** check box.

Next you must set up the Position Constraint histogram for the edge, in the bottom left window of the Configuration panel.

2. Use the mouse to drag the handles in the histogram editor of the selected constraint. Set the Edge Position Constraint histogram as illustrated in [Figure 79](#).
3. Click **Execute**.

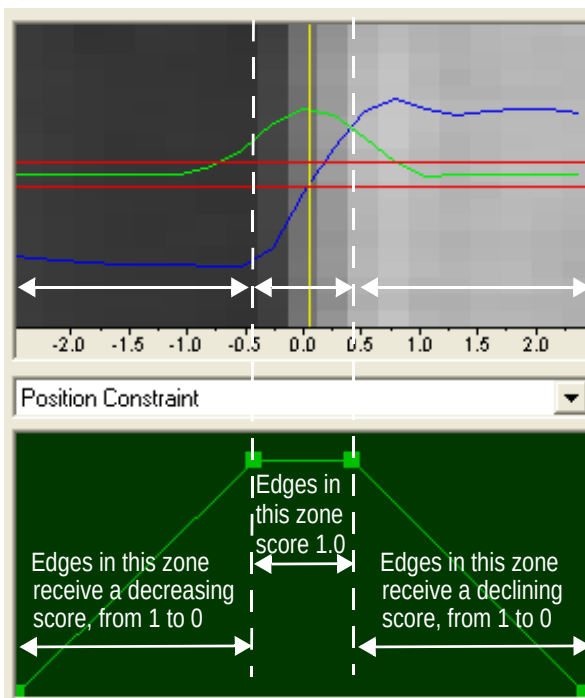


Figure 79 Position Constraint Histogram

Edge Locator Results

You will now observe the Edge Locator's results for various instances of the object. Access the Edge Locator **Results** panel to observe the following:

- Valid edges are displayed as vertical yellow lines.
- The Edge Count box indicates that the Edge Locator has found one edge.
- The Coordinate System combo box indicates that the positions are expressed with respect to the Tool coordinate system (the origin centered on the tool's Rectangle).

The grid of results indicates:

- The score of the found edges.
- The position of the edges, expressed with respect to the selected coordinate system.
- The rotation of the edges.
- The scores of each individual constraint.
- Use the scrollbar to view the entire grid of results.

Now that the Edge Locator is configured to locate the left edge of the part, you will want to test and observe the Edge Locator on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the next step.

Placing the Second Edge Locator

Using the Process Manager, add a second HSEdgeLocator process. Set the tool's **General** panel as follow:

1. Enter *PartRightEdge* as the **Process Name** (instead of HSEdgeLocator7).
2. Select Grey-Scale Image from Acquisition View as the **Input Image**.

3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select Hook from Model View as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select Instance Scene from Locator View as the **Input Instance Scene**.
7. Click **Execute**. Open the **Location** panel.
8. Place the Rectangle on the part as shown in [Figure 80](#). Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees. Click the center of the Rectangle and drag it into position. Since the edge to be located is not orthogonal to the tool's X-axis, adjust the skew by clicking and dragging the arrow of the Y-axis to make it parallel to the right edge of the part. Fine-tune the position with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

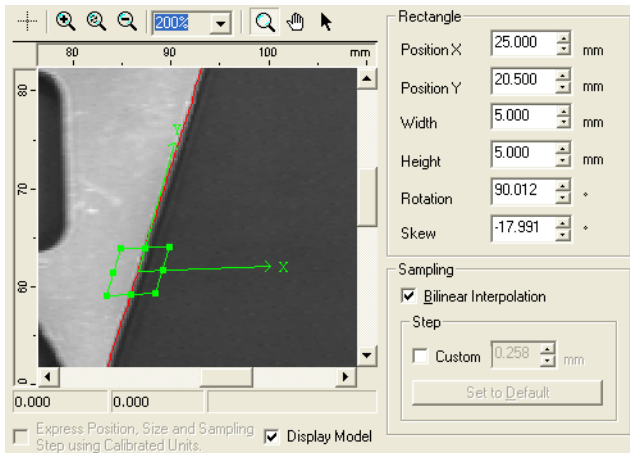


Figure 80 Adjusting Rectangle for the Hook's Right Edge

9. In the **Edge Score** box of the Configuration panel, select **Polarity** as **Light to Dark** and enable the **Position** check box.

10. Set up the Position Constraint using the graphical histogram editor, just as that of the PartLeftEdge Edge Locator.
11. Save your work and move on to the final step.

Adding Code for the Edge Locator

You will finish up by adding code to your Visual C++ class that will display the width of the part computed from the results of the two Edge Locator tools. You will also add a line marker on the display to show the measure on the part.

1. Add this include file following the others in the HookInspectionDlg.cpp file:

```
#include "HSEdgeLocator.h"
```

2. Add this wrapper (.h and .cpp) in your project: HSEdgeLocator.

3. In the **OnButtonInspect** function, insert the lines of code shown in bold:

```
HSPatternLocator lLabelLocator;
HSEdgeLocator    lPartLeftEdge;
HSEdgeLocator    lPartRightEdge;
...
double lHoleOffset;
double lPartWidth;
...
lLabelLocator = lManager.GetProcess( COleVariant(
"LabelLocator" ) );
lPartLeftEdge = lManager.GetProcess( COleVariant(
"PartLeftEdge" ) );
lPartRightEdge = lManager.GetProcess( COleVariant(
"PartRightEdge" ) );
```

This will create a reference that will be used for programmatic access to the two Edge Locator processes instantiated in the HexSight Application Control.

4. Modify the existing line, shown in *italic*, for the one below it:

```

lManager.Execute( COleVariant( "Acquisition" ),
                  COleVariant( "LabelLocator" ) );
lManager.Execute( COleVariant( "Acquisition" ),
                  COleVariant( "PartRightEdge" ) );

```

This will execute the entire sequence of processes.

- Now to display the measurement in the edit box and to draw the marker on the application display, type the lines of code, shown in bold:

```

if ( lLabelLocator.GetMatchCount() > 0 )
{
    ...
}

// Display part edge left and part edge right properties
and marker

if ( ( lPartLeftEdge.GetEdgeCount() > 0 )
    && ( lPartRightEdge.GetEdgeCount() > 0 ) )
{
    lPartLeftEdge.SetCoordinateSystem(
HSEdgeLocator::hsCoordinateSystemObject );
    lPartRightEdge.SetCoordinateSystem(
HSEdgeLocator::hsCoordinateSystemObject );
    lPartWidth = lPartRightEdge.GetEdgePositionY(0) -
        lPartLeftEdge.GetEdgePositionY(0);
    lBuffer.Format( "%.2f", lPartWidth );
    mPartWidthText.SetWindowText( lBuffer );

    lPartLeftEdge.SetCoordinateSystem(
HSEdgeLocator::hsCoordinateSystemWorld );
    lPartRightEdge.SetCoordinateSystem(
HSEdgeLocator::hsCoordinateSystemWorld );
    lMarkerName = "Part Width";
    mApplicationDisplay.AddLineMarker( lMarkerName,

```



```

        lPartLeftEdge.GetEdgePositionX( 0 ),
        lPartLeftEdge.GetEdgePositionY( 0 ),
        lPartRightEdge.GetEdgePositionX( 0 ),
        lPartRightEdge.GetEdgePositionY( 0 ), TRUE );
mApplicationDisplay.SetMarkerAnchorStyle( lMarkerName,
        HSDisplay::hsCross );
mApplicationDisplay.SetLineMarkerConstraints(
    lMarkerName,
        HSDisplay::hsLineNoEdit );
mApplicationDisplay.SetMarkerColor( lMarkerName,
        HSDisplay::hsGreen );
mApplicationDisplay.SetMarkerDisplayName( lMarkerName,
        TRUE );
}

```

6. In case no instance is found, add the line of code shown in bold:

```

mLabelText.SetWindowText( "" );
mPartWidthText.SetWindowText( "" );

```

7. Save your work and test the application. You should see the part width in the Part Width text box of your application. You should also see the line marker you added on the Display highlighting the measurement.

Visual C++ Tutorial 7: Calibration

In this tutorial, you will learn how to calibrate the system to perform acquisitions of live images. Calibration is an important step that can correct certain perspective and lens distortions. It also allows you to specify the basic length units you will use throughout your entire application.

Camera Calibration

The calibration is performed in the Acquisition Device. In this example, the calibration is performed with the largest calibration target provided with your HexSight package. This target has a dot pitch of 10 mm.

1. Open the Acquisition Device Properties. Select the **Configuration** panel. Select **Copy Configuration** and then select **Paste Configuration**.
2. In the displayed window, enter *Camera* as the new configuration name.
3. Select the newly created *Camera* configuration in the list.
4. In the **Input** sub-panel, select **Generic Frame Grabber** and click **Apply**.
5. Go to the **Calibration** sub-panel. Select **Distortion Model**. Click **Show Calibration Dialog**. The **Calibration - Distortion Model** dialog is displayed (see [Figure 81](#)).

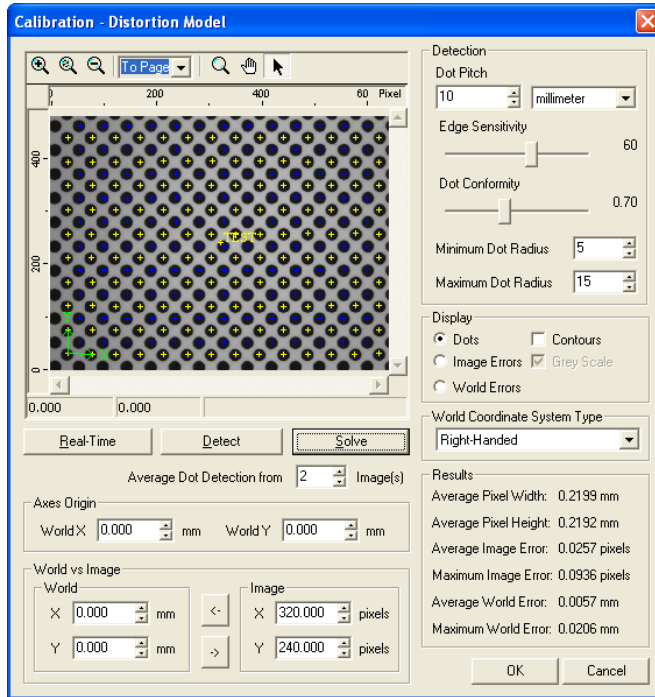


Figure 81 Distortion Model Calibration Dialog

6. Click **Real-Time** to display the image from the camera. Position the grid of dots to cover the whole area. Set the focus and other settings of your camera to obtain the best image possible. Then click **Real-Time** again to stop real time acquisitions.
7. Click **Detect** to detect the dots on the target.
8. The dot pitch is 10.0 mm. Enter the dot pitch in the **Dot Pitch** box.
9. Zoom in at the lower left corner of the display. The green coordinate system marker in the bottom left of the target displayed, as in [Figure 82](#). Note that this coordinate system is displaceable programmatically or with the mouse; leave it as is for this tutorial.

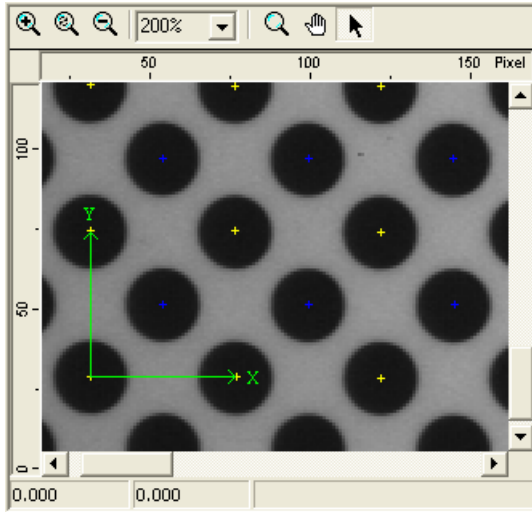


Figure 82 Zoom-in on the Coordinate Marker in the Calibration Dialog

10. Click **Solve**. The statistics appear in the **Results** frame. Click **OK**.
11. The calibration procedure is completed. The computed calibration parameters appear in the **Parameters** grid. Click **Apply** to apply the calibration to the *Camera* configuration.
12. Click **OK** to close the Acquisition Device's properties window.
13. You are now ready to run the application with live images using the Paper Hook provided with your HexSight package.

C# Tutorials

The tutorials in this chapter will guide you through the development of a HexSight application in C#. As you follow the steps for each tutorial you will build an object location application to which you will add and configure a full range of inspection tools.

Note: This tutorial assumes you have a basic knowledge of C#.

Chapter Outline

[C# Tutorial 1: Locator](#)

[C# Tutorial 2: Display](#)

[C# Tutorial 3: Caliper](#)

[C# Tutorial 4: Blob Analyzer](#)

[C# Tutorial 5: Pattern Locator](#)

[C# Tutorial 6: Edge Locator](#)

[C# Tutorial 7: Calibration](#)

C# Tutorial 1: Locator

This first tutorial shows you how to build a basic HexSight application that will locate a model-defined object at whatever angle and displacement it appears.

Creating the Main Form

In this section, you will create the Main Form of the application, add basic lines of code to interact with the interface, and add a HexSight Application Control, which you will edit to build the application.

Constructing the Main Form Interface

Here you will build the interface that will allow you to interact with your application and visualize results.

1. Start Microsoft Visual Studio. Open a new **Visual C# Windows Application** project called **HookInspection**. From **Explorer** View, change the name of the form to *HookInspectionMainform.cs*.
2. Make the toolbox and the properties window visible, using the View menu.
3. Set the form *Text* property to **Hook Inspection**. Set the *FormBorderStyle* property to **FixedSingle** and the *MaximizeBox* property to **False**. Set the *Width* to approximately 355 and the *Height* to 275.
4. On the form, add a group box named **InstanceFrame**. Add 5 text boxes respectively named **TypeText**, **ScaleText**, **RotationText**, **TranslationXText** and **TranslationYText**. Also, set the frame caption, the text box dimensions and add labels. The text property for the text boxes should be empty. Your form should look like the form in [Figure 83](#).

The image shows a Windows form titled "Hook Inspection". Inside the form, there is a group box labeled "Object Instance". Within this group box, there are five text input fields arranged as follows: "Type" is at the top left; "Scale Factor" and "Rotation" are in the middle row; "Translation X" and "Translation Y" are in the bottom row.

Figure 83 C# Form with Object Pose Information

5. At the bottom of the form, add three command buttons named **InspectButton**, **ConfigureButton** and **QuitButton**. Set the caption of these buttons respectively to: **Inspect**, **Configure** and **Quit**.
6. Add a check box named **ContinuousCheck**, and set its caption to **Continuous Mode**. Finally, add an text box named **TimeText** and an appropriate label. Place everything to resemble [Figure 84](#).
7. The interface is almost complete. Save your work.

The image shows the same "Hook Inspection" form as in Figure 83, but with additional controls. Below the "Object Instance" group box, there is a checkbox labeled "Continuous Mode" and a text box labeled "Time (ms)". At the bottom of the form, there are three command buttons labeled "Inspect", "Configure", and "Quit".

Figure 84 C# Form with Command Buttons

Adding a HexSight Application Control

You will now add a new component to the Microsoft Visual Studio toolbox: the HexSight Application Control, which is the core of each HexSight application.

1. From the toolbox context menu, select: *Choose Items...* Move to the *COM Components* tab and scroll down to find the **HSApplication Control**. Enable its check box and click **OK**. The HexSight Application Control appears as a lens shutter icon in the toolbox.
2. Click the **HSApplication Control** icon and drag a small box on the form with your cursor. The default name of this control is `axHSApplication1`. In the Properties window change its name to *ApplicationControl*. The interface of the project is now complete. Save your work. You will configure the HexSight Application Control in subsequent tutorials.

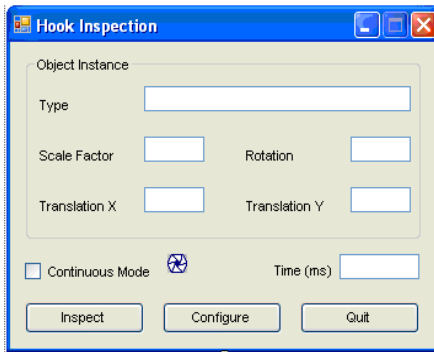


Figure 85 C# Form with HexSight Application Control

Adding Basic Code to the Main Form

You need to add basic code to be able to interact with your application. As you will add new features throughout the other tutorial sections, you will also add lines to this code.

1. Double-click **Quit** and insert the line of code shown in bold:

```
private void QuitButton_Click(...)
{
```



```

        Application.Exit();
    }

```

This will terminate the application when the **Quit** button is clicked.

2. Double-click **Configure** and add the line of code shown in bold:

```

private void ConfigureButton_Click(...)
{
    ApplicationControl.ShowInterface(false);
}

```

This will show the HexSight Application Control's interface in a non-modal window when the **Configure** button is clicked.

3. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find the **HexSight Process Manager Control**, select it and click **OK**.

The HexSight Process Manager is now added to the project's references and can thus be accessed programmatically. Note that in order to work properly, the Process Manager should only be set through the *Project -> Add Reference...* menu and not through the toolbar *Choose Items...* context menu item.

4. Double-click **Inspect** and add the lines of code shown in bold:

```

private void InspectButton_Click(object sender,
    EventArgs e)
{
    HSPROCESSMANAGERLib.HSProcessManager lManager;

    InspectButton.Enabled = false;

    lManager = ApplicationControl.ProcessManager;

    // Inspection loop
}

```

```

do
{
    Application.DoEvents();
}
while (ContinuousCheck.CheckState == CheckState.Checked);

InspectButton.Enabled = true;
}

```

This is the main loop of the inspection application. It loops until **Continuous Mode** is disabled in the interface.

5. Add **Load** and **FormClosed** event handler, as shown below.

```

private void HookInspectionMainForm_Load(...)
{
}

private void HookInspectionMainForm_FormClosed(...)
{
    QuitButton_Click(sender, e);
}

```

You will later place initialization code for the empty *Form_Load* method. Clicking the **Close** button calls the *Form_FormClosed* method and ends the application.

You have completed the code needed to use your application. Run and test the application. When you are sure that everything works fine, save your work and go on to the next section.

Adding an Acquisition Device Tool

In this section, you will learn how to instantiate a process in the HexSight Application Control. You will also learn how to configure an Acquisition Device tool. This tool will acquire an Image from an existing database file to simulate real image acquisition.

Instantiating the Acquisition Device

In order to build a vision application, HexSight processes are instantiated using the Application Control's Process Manager.

1. On the main form, right-click the **HexSight Application Control**. Select **ActiveX-Edit** from the menu. The HexSight Application Control's interface opens and the active tab is the Process Manager. In the **Available Processes** list of the Process Manager, select **HSAcquisitionDevice** and click **Add Process**. The Acquisition Device Properties window appears.

At the end of this tutorial, the HexSight Application Control should look like the illustration in [Figure 86](#).

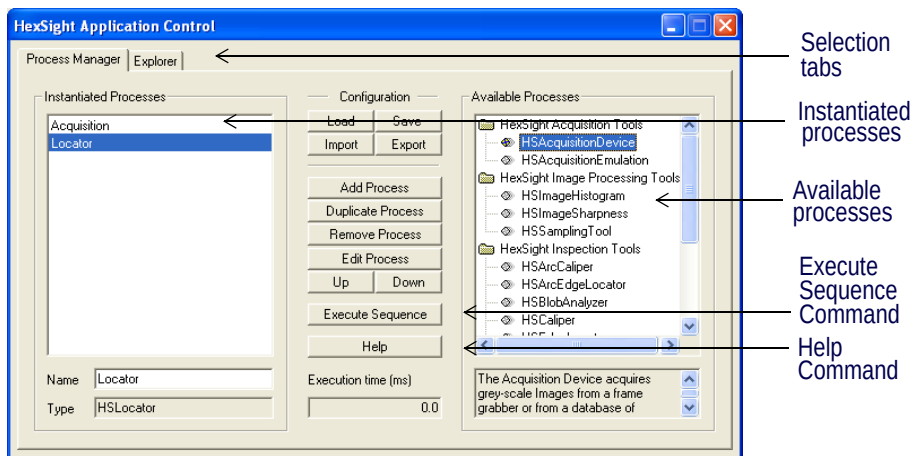


Figure 86 HexSight Application Control — Process Manager

Configuring the Acquisition Device

You will use an existing database of images to simulate a real image acquisition. The database used for this tutorial is constituted of images of the same objects, rotated at different angles.

1. In the Acquisition Device **General** panel, enter *Acquisition* as the **Process Name** (instead of HSAcquisitionDevice0).

2. In the **Configuration** panel, select the **Input** tab. Select **File Emulation** from the list.
3. Click **Load** then load the following file:
InstallationPath\Tutorial\C#\HookInspection\Data\Hook.hdb.
4. Disable all the images of the emulation database except the first one: *Image0*. This ensures that this image will be used to create the model later in this tutorial.
5. Leave the default settings for the Calibration and Output sub-panels. Click **OK** to return to the Process Manager of the HexSight Application Control

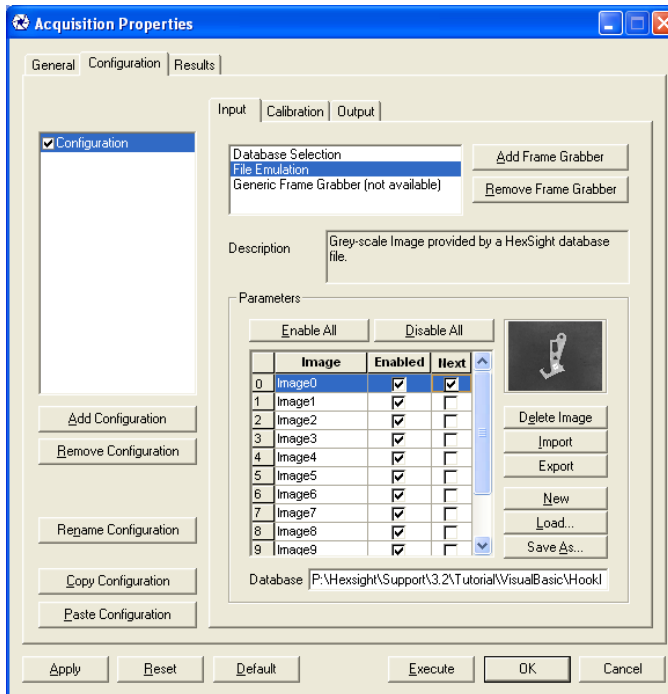


Figure 87 Acquisition Device — Configuration Panel

To sequentially execute the processes instantiated in the Process Manager, click **Execute Sequence**. At this stage, each time you click **Execute**

Sequence, the Acquisition Device enables and outputs a new emulation image to the applications runtime database.

Viewing Scenes and Images

The Explorer panel in the HexSight Application Control shows the application's runtime database as a hierarchical list.

1. Select the **Explorer** panel in the **HexSight Application Control**.
2. Click on + symbols to expand **RunTime** and **Acquisition**.
3. Click **Grey-Scale Image** to display the current Image output by the acquisition process.

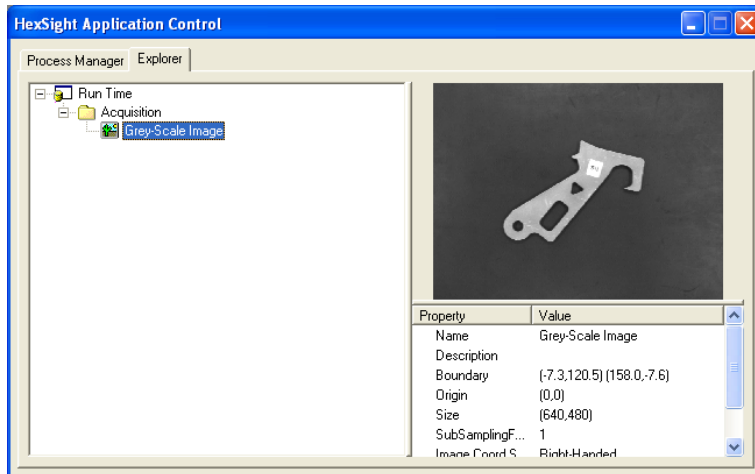


Figure 88 HexSight Application Control — Explorer

Adding a Locator Tool

The Locator locates model-defined objects within the input Image, using the geometry of object contours.

For this tutorial you will use only the basic Locator properties, therefore the **Advanced Parameters** check box should be left unchecked.

Instantiating the Locator

Here you add a Locator process the same way you added the Acquisition Device.

1. In the HexSight Application Control Process Manager, select **HSLocator** from the **HexSight Locator Tools** folder. Click **Add Process**.
2. In the properties window of the newly added Locator process, select the **General** tab.
3. Enter the name *Locator* instead of HSLocator1.
4. In the **Input** frame, click on the selection icon at right to select **Image** from **Acquisition** view.

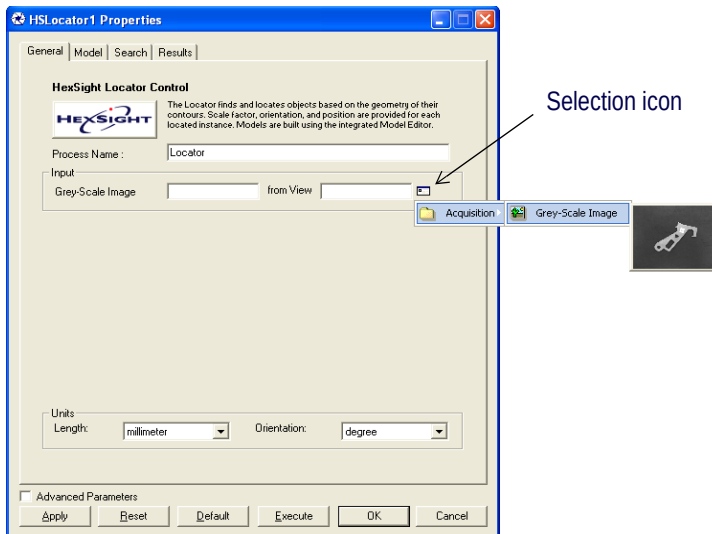


Figure 89 Database Selection Icon

5. Click **Execute**.

Creating a Model

In this section, you will learn how to create a Model of an object using the Model Editor. A Model is a reference or “ideal” representation of an object’s contours that will be used by the Locator’s search process to recognize and

locate instances of the object. Models can be stored in database files for later use.

1. Select the **Model** tab, then click **Add** in the **Models** frame. The **Create New Model** prompt appears.

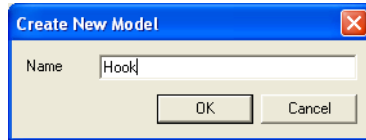


Figure 90 Create New Model Dialog

2. Name the Model *Hook*.
3. Click **OK** to open the Model Editor.

Using the Model Editor

The Model Editor is used to create new models and to modify existing models. Once you have created a model of the hook for this application, you will save it for later use with the rest of the tutorial.

Creating the Source Contours and Selecting Model Features

The Locator uses two coarseness levels to recognize and locate the object: the **Outline Level** and the **Detail Level**. The outline coarseness level is used to rapidly identify potential instances of the object. The detail coarseness level is used to confirm that it is the correct object and to refine its location within the image. Source contours will be created at the selected **Outline Level** and **Detail Level**. Features will be selected from these source contours to create the Model.

When the **Automatic Levels** check box is enabled, the two contour detection parameters (**Outline Level** and **Detail Level**) are automatically optimized by analyzing the Model's bounding area. Automatic contour detection parameters should usually be used when building a Model.

Contrast Threshold is used to set the sensitivity to contrast when generating the source contours. **Contrast Threshold** can be set manually, by selecting **Fixed Value** with the combo box, or automatically by selecting

Adaptive Low Sensitivity, Adaptive Normal Sensitivity or Adaptive High Sensitivity.

Once the source contours are detected, features must be selected to build the Model. Features can be selected manually, by setting the **Feature Selection** slider to **none**, or automatically by selecting **less**, **normal**, **more** or **all**. Setting the slider to **normal** will select the optimal amount of features at both the Outline and Detail Levels. Increasing the setting will add more features, which may slow the search process. Decreasing the setting may cause recognition problems because the Model may lack essential features.

In this tutorial, we will use the default contour detection and feature selection parameters to build the model, as shown in [Figure 91](#).

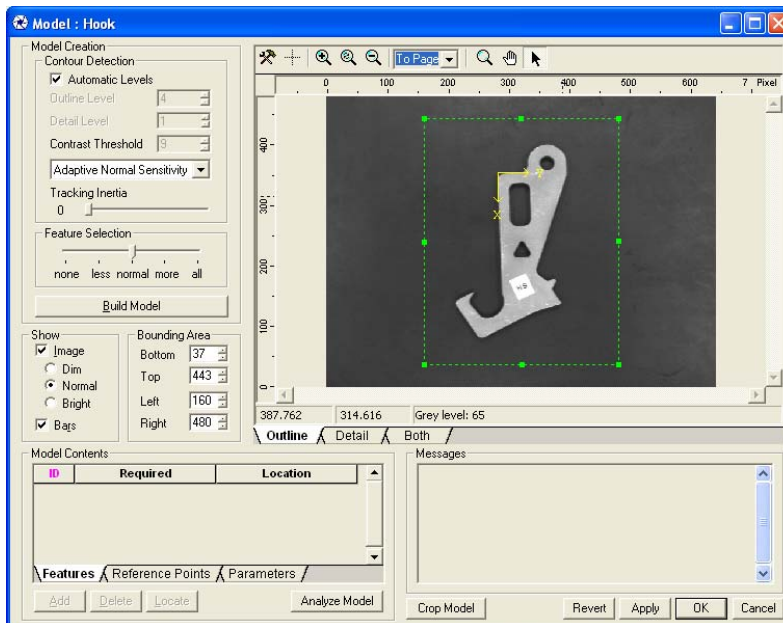


Figure 91 Model Editor Interface

1. In the **Contour Detection** frame, leave the **Automatic Levels** check box enabled, and **Contrast Threshold** combo box to **Adaptive Normal Sensitivity**.

2. In the **Feature Selection** frame, leave the **slider** in the default **normal** position.
3. In the display, drag and resize the green dotted rectangle (the Model's bounding box) to enclose the entire object.
4. Click **Build Model**.

The new Model is now created. Contour detection parameters have been optimized and features were selected at both the Outline and Detail levels. To verify that no problem was encountered while building a model, check the **Messages** text box.

You can view and edit the Model in the display. Select the **Outline** tab at the bottom of the display to view the Model at the Outline Level. Outline features are shown as bold magenta lines. Unused contours are shown as thin blue lines. Select the **Detail** tab at the bottom of the display to view Detail Level features. Detail features are shown as bold green lines. As with the Outline Level, unused contours are shown as thin blue lines. The **Both** tab displays both Outline and Detail Levels as superimposed.

Editing Models

You can now use the Model to locate instances of the object. We will, however, edit it to show how the Model Editor can be used to customize a model for a specific application.

Removing Features from a Model

The application we are building in this tutorial will be used to inspect features on the hook with respect to its outer contour. The steps will be to locate the part using its outer contour and apply tools to inspect the rounded rectangle, the hole and the label. These features need to be inspected because they might be missing on actual parts, be the wrong size, or be mislocated with respect to the outer contour. Because these features are unstable, they should be removed from the model to ensure the best possible accuracy for locating the outer contour of the object.

The first step will be to remove unwanted features at the Outline Level.

1. Click the **Outline** tab at the bottom of the display. It will enable you to edit the Model at the Outline Level. The grid in the **Model Contents** frame displays the **Features** sub-panel for the Outline level of the Model.
2. Click the **Features** tab in the **Model Contents** frame to activate the features grid.
3. Select the rounded rectangle feature by either clicking it in the display or by selecting it in the **Features** grid.
4. Click **Delete** to remove this feature from the Model.

The next step will be to remove unwanted features at the Detail Level.

5. Click the **Detail** tab at the bottom of the display. It will enable you to edit the Model at the Detail Level. The grid in the **Model Contents** frame displays the **Features** sub-panel for the Detail level of the Model.
6. Select all the features, except the outer contour, by clicking them while holding the **Ctrl** key either in the display or in the **Features** grid.
7. Click **Delete** to remove these features from the Model.

The unstable features are now removed from the model. To view the outline and Detail Levels superimposed, click the **Both** tab at the bottom of the display. In this viewing mode, the model is not editable.

Setting the Object Coordinate System for a Model

The part inspection process requires that the measurements of the features be expressed with respect to the outer contour of the part. To fix a reference frame to express these inspection results, you will define an Object coordinate system for the part.

1. Click the **Outline** or the **Detail** tab at the bottom of the display. It will enable you to edit the Object coordinate system.
2. Click the **Coordinate System** tab in the **Model Contents** frame to activate the coordinate system panel.

3. Use the mouse to position the coordinate system marker (yellow axes) over the object, as shown in [Figure 92](#).

To rotate the object coordinate system, click on the arrow of the X or the Y-axis and drag with the mouse.

To move the coordinate system, click the intersection of the X- and Y-axes and drag with the mouse.

You can also drag the arrow of the axes to stretch the marker. Doing so has no effect on the Object coordinate system itself, but can help you align the marker over long features. Fine-tune the final position with the text boxes in the **Coordinate System** panel.

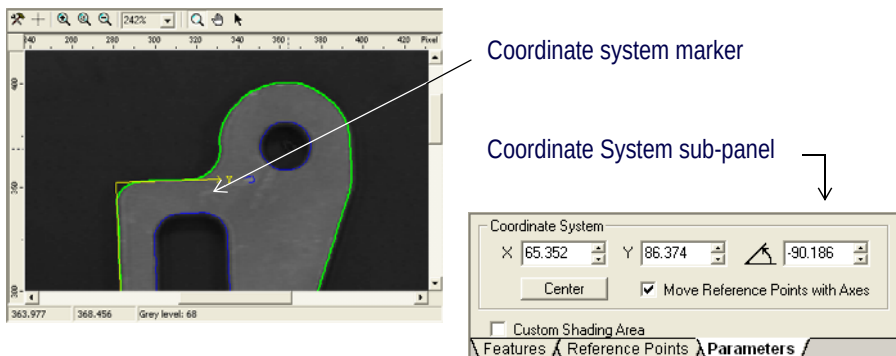


Figure 92 Adjusting the Position and Orientation of the Model Coordinate System

Adding Reference Points to a Model

For some applications, it is sometimes useful to have the position of specific points on the Model transposed in the World coordinate system for each object instance located. To do so without having to compute coordinate transforms manually, reference points can be defined on a Model. In this application, we will add a reference point above the circular hole.

1. Click the **Outline** or the **Detail** tab at the bottom of the display. It will enable you to edit the reference points.

2. Click the **Reference Points** tab in the **Model Contents** frame to activate the reference points panel.
3. Click **Add** to create a reference point in the display.
4. Click and drag the target marker to position the reference point above the circular hole in the part. Zoom in for more precision. Fine-tune the position using the text boxes in the **Reference Points** panel.

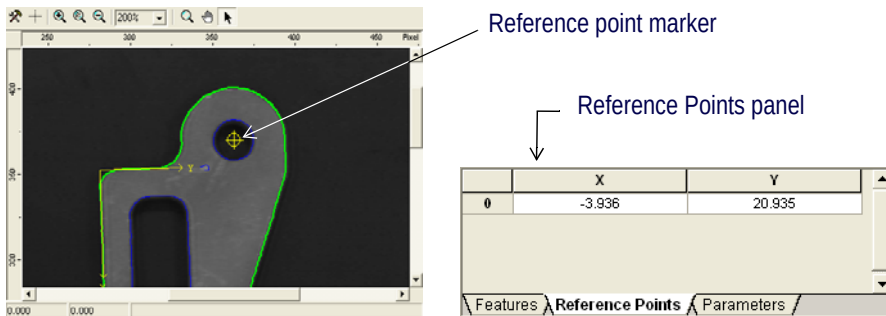


Figure 93 Adding a Reference Point with the Model Editor

This reference point will not be used to locate the part, only contour features are used for locating objects. However, for each instance of the object that will be located, the reference point's position will be returned with respect to the scale, rotation and translation of this instance. You can add as many reference points as you need on a Model.

Manually Adding Features to a Model

Sometimes, it may be useful to manually add a feature to a Model. In Model Editor, you can select features from the unused contours at the Outline Level as well as the Detail Level. To add a feature at the Outline Level, follow the procedure below.

1. Select the **Outline** tab at the bottom of the display to edit the Model at the Outline Level.
2. Click the **Features** tab in the **Model Contents** frame to activate the features grid.

3. Disable the **Image** check box in the **Show** frame to show only the contours. This will make the blue unused contours more visible.
4. In the display, click one of the unused contours, the rounded rectangle for example. The contour turns light blue.
5. Click a starting point on the contour. Hold the **Ctrl** key and click another point on the contour. The selected feature appears in red.
6. To invert your selection, hold the **Shift** key while clicking on the contour. Click **Add** in the **Model Contents** frame. The new feature appears in the features grid.
7. Select the newly added feature in the grid and click **Delete** to remove it from the Model.
8. To add the entire rounded rectangle as a feature, double-click it in the display so the whole contour is selected. Click **Add** to add this feature to the model.
9. Select the newly added feature in the grid and click **Delete** to remove it from the Model, since it is considered unstable for this application.

Analyzing a Model

When a Model has been edited manually, it is important to verify that it will work correctly with the Locator. To do so, click **Analyze Model** in the **Model Contents** frame. Any problem will be displayed in the **Messages** frame.

Saving a Model Database

When a Model is completed, it must be saved in a model database, otherwise it will be lost when you exit the application. To save the Model, follow the procedure below.

1. Click **OK** at the bottom right of the Model Editor. This will transfer the newly created Model to the active model database of the Locator.
2. In the Locator's properties window, select the **Model** tab.
3. Click **Save As...** in the **Database** frame. Save your Model as *HookModel* in your working directory. You will find it convenient to store

Models close to other HexSight files used in a same application, such as image emulation files, the Process Manager configuration file, etc.

4. You can return edit your Model at any time by clicking **Edit** in the **Models** frame.

Configuring Search Parameters

Search parameters are used to constraint the search and configure object recognition and positioning algorithms.

1. Select the **Search** tab.
2. Leave the **Nominal Scale** check box enabled. Its value in the text box should be 1.0.
3. Leave the **Nominal Rotation** check box disabled, with the range between -180.0 to 180.0.
4. Leave all the other parameters to their default setting also, as shown in [Figure 94](#).

Search Constraints		Nominal	Minimum	Maximum
Scale	☒	1.000	0.500	2.000
Rotation	☐	0.000	-180.000	180.000
Instances to Find	☒	1		

Recognition	
Minimum Model Percentage	50

Figure 94 Search Panel of the Locator Tool

Executing the Locator and Verifying the Results

Once the Model is created and the search is configured, the first step is to verify that the Locator finds the object on the image that was used to create the model.

1. Select the **Results** tab.
2. Click **Execute** at the bottom of the Locator's properties window.

3. Verify in the grid that the instance was located correctly.
4. Click **Show Results Viewer** to view the located instance superimposed on the Image, as shown in [Figure 95](#).
5. Move the Locator properties window and the **Results Viewer** to a corner of your screen.

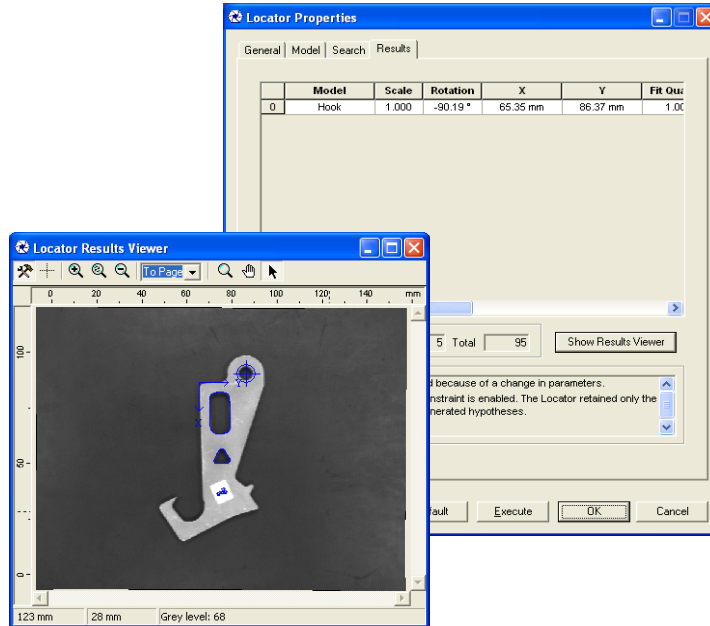


Figure 95 Results Viewer and Results Panel of the Locator Tool

The next step is to verify that the object is located in all the images in the emulation database.

1. In the Process Manager, open the Acquisition Device properties window. Select the **Configuration** panel, then click **Enable All** to enable all the images in the emulation database. Click **OK** to close the Acquisition Device's properties window.
2. Return to the Process Manager and click **Execute Sequence**.
3. In the **Results** panel of the Locator properties window, check if the instance was located correctly. Return to the Process Manager and click

Execute Sequence several times to apply the Locator to each of the images.

4. Close the Results Viewer window, return to the Locator Properties window and click **OK**.
5. Now is a good time to return to the C# form and save your work. Close the window to exit the Process Manager. To return to the application, you will have to right-click the HexSight icon and select **Edit** (or **Show Application Interface** if the icon is still selected).

Adding Code for the Locator

In this section, you will add code to output the properties of the instance found by the Locator in your application interface.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Locator Control**, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
HSPROCESSMANAGERLib.HSProcessManager lManager;  
HSLOCATORLib.HSLocator lLocator;
```

```
InspectButton.Enabled = false;
```

```
lManager = ApplicationControl.ProcessManager;
```

```
lLocator = (HSLOCATORLib.HSLocator)lManager.  
get_Process("Locator");
```

This code creates a reference to the Locator process contained in the HexSight Application Control. This reference will be used to access the tool programmatically.

2. In the existing "Do Loop" of the InspectButton_Click procedure, add the lines of code shown in bold:

do


```

{
    // Execute processes

    lManager.Execute( "Acquisition", "Locator" );

    if (lLocator.InstanceCount > 0)
    {
        // An instance of the object is found

        TypeText.Text = lLocator.get_InstanceModelName(0);

        // Output the properties of the located instance
        // in text boxes

        ScaleText.Text = string.Format("{0:F2}",
            lLocator.get_InstanceScaleFactor(0));
        RotationText.Text = string.Format("{0:F2}",
            lLocator.get_InstanceRotation(0));
        TranslationXText.Text = string.Format("{0:F2}",
            lLocator.get_InstanceTranslationX(0));
        TranslationYText.Text = string.Format("{0:F2}",
            lLocator.get_InstanceTranslationY(0));
    }
    else
    {
        // No object is found

        TypeText.Text = "";
        ScaleText.Text = "";
        RotationText.Text = "";
        TranslationXText.Text = "";
        TranslationYText.Text = "";
    }
}

```

```

// Compute elapsed time

TimeText.Text = string.Format("{0:F2}",
    lManager.ElapsedTime);

Application.DoEvents();
}
while (ContinuousCheck.CheckState == CheckState.Checked);

```

Each time the loop is executed, this code executes the Application Control's sequence of process from "Acquisition" to "Locator". If no parameter is specified to the Execute method of the Process Manager, the whole sequence of processes will be executed, which in this case will yield the same result. After HexSight processes are executed, the pose (scale, rotation and translation) and type (model name) of the located instance are displayed in the interface.

3. Save your work and move on to the next step.

Testing the Application

You are ready to test your application.

1. To start the running Mode, press the **F5** key. Click **Inspect** a few times.

The properties of the found instances displayed in the text boxes should be updated every time you press **Inspect**.

2. Enable the **Continuous Mode** check box and click **Inspect**. The application should run in continuous mode. Exit the Continuous Mode by disabling the check box. Exit the Run mode by closing the window or by clicking **Quit**.

This concludes this tutorial. After debugging, save your work and move on to the next tutorial where you will add a display to your application.

C# Tutorial 2: Display

This second tutorial shows you how to add a Display to your application interface. A Display allows you to view Images and Scenes processed by the HexSight Application Control.

Adding the Display to the Interface

First, you will modify the interface of your application by adding the Display.

1. Start by extending the form. Set the width to approximately 735 and the height to approximately 420. Reorganize the text boxes and buttons to look like the form in [Figure 96](#).

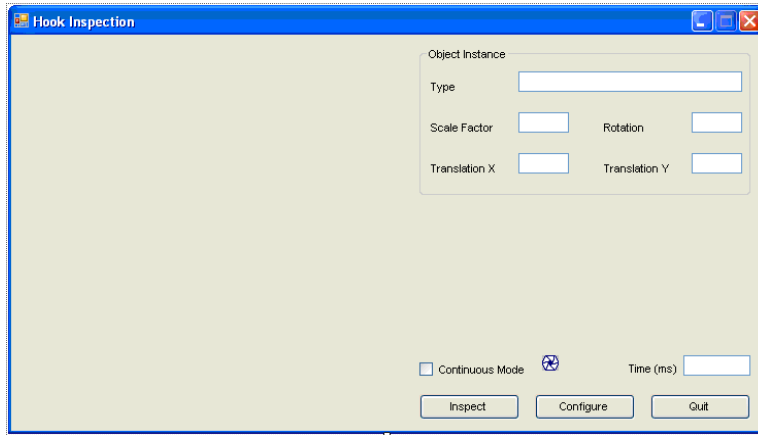


Figure 96 Resized C# Form for Tutorial 2

2. From the toolbox context menu, select: *Choose Items...* Move to *COM Components* tab and scroll down to find **HSDisplay Control**. Enable its check box and click **OK**. The HexSight Display appears as a new icon in the toolbox.
3. Click the **HSDisplay Control** icon and drag a small box on the form. Set the width to 378 and height to 365. The default name is `axHSDisplay1`. Rename it `ApplicationDisplay`. The interface of the project is now complete and should look like the form in [Figure 97](#).
4. Save your work.

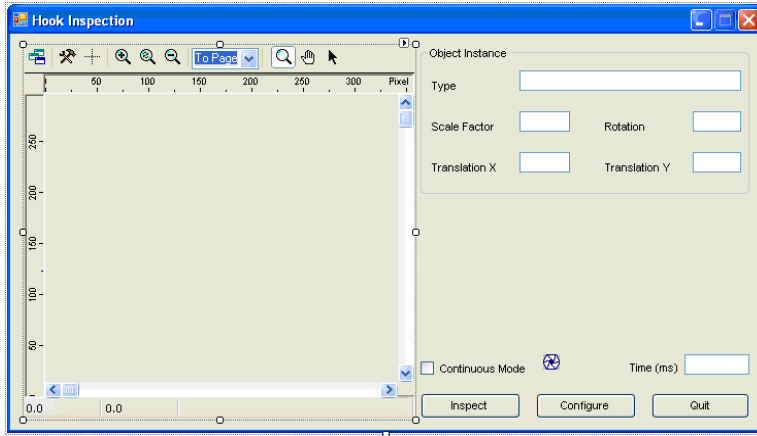


Figure 97 C# Form for Tutorial 2

Adding Code for the Display

Code must now be added to update the Display output every time the inspection loop is executed.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Acquisition Device Control**, select it and click **OK**.
2. Insert the lines of code shown in bold into the existing `Form_Load` procedure:

```
private void HookInspectionMainForm_Load(...)
{
    HSACQUISITIONDEVICELib.HSAcquisitionDevice lAcquisition;

    lAcquisition = (HSACQUISITIONDEVICELib.
    HSAcquisitionDevice)ApplicationControl.ProcessManager.
    get_Process("Acquisition");

    // Inits the Application Display
}
```

```

ApplicationDisplay.set_ImageDatabase(0,
    ApplicationControl.Database.Handle);

if (!(lAcquisition == null))
{
    ApplicationDisplay.set_ImageViewName(0,
        lAcquisition.get_OutputView(null));
    ApplicationDisplay.set_ImageName(0,
        lAcquisition.get_OutputImage(null));
}

ApplicationDisplay.Zoom = -1;

// To ensure releasing of local references to Hexsight
// ActiveX objects

lAcquisition = null;

GC.Collect();
GC.WaitForPendingFinalizers();
}

```

This initialization code configures the Display Control to display the Image provided by the Acquisition Device output. It also sets the zoom to page mode, which automatically adjusts the zoom factor to view the entire image.

3. In the InspectButton_Click procedure, insert the lines of code shown in bold:

```

...
lManager.Execute( "Acquisition", "Locator" );

ApplicationDisplay.RemoveAllMarker();

if (lLocator.InstanceCount > 0)

```

```
ApplicationDisplay.RefreshDisplay();
```

```
Application.DoEvents();
```

```
...
```

The `RemoveAllMarker` call removes all overlay graphic markers that could have been added to the previous image. The refresh call updates the display to show the new Image.

4. Save your work and test the application. Now along with the properties of the found instance, you should see an image of the object for the current instance.

C# Tutorial 3: Caliper

In this tutorial, you will learn how to set up, configure and use a Caliper tool to precisely measure the distance between parallel edges on an object.

Modifying your Application Form

First you need to add two edit boxes to your application for the interface to display the measurement output by the Caliper.

1. On your application form, add a group box named **InspectionFrame** and add two text boxes, the first text box named **WidthText** and the second **HeightText**. Set up your form as illustrated in [Figure 98](#).

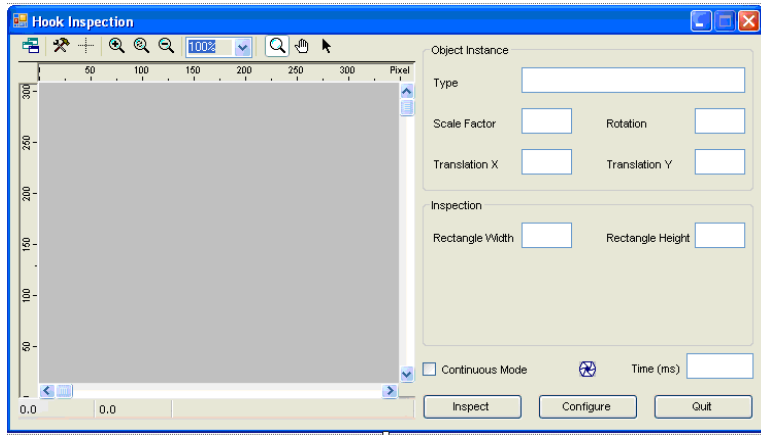


Figure 98 C# Form for Tutorial 3

Placing the Caliper

Right-click the HexSight Application Control icon on the form and select **ActiveX -Edit** from the context menu. Click **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add an **HSCaliper** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *WidthCaliper* as the **Process Name** (instead of HSCaliper2).

2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Leave the **Lock Location Transform on a Selected Instance** check box enabled.
6. For the **Input Instance Scene**, select **Instance Scene** from **Locator View**.

The above settings will make your application fully model-based (automatic model-based mode). The Caliper tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Caliper is applied to an area on the object bounded by the Caliper's Rectangle, which appears in green. The Caliper will be used to measure the width of the rectangle-shaped hole on the part.

8. Place the Rectangle on the part as shown in [Figure 99](#).
9. Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees (the Y-axis is used to adjust the skew).
10. Click the center of the Rectangle and drag it into position.
11. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position and the orientation with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

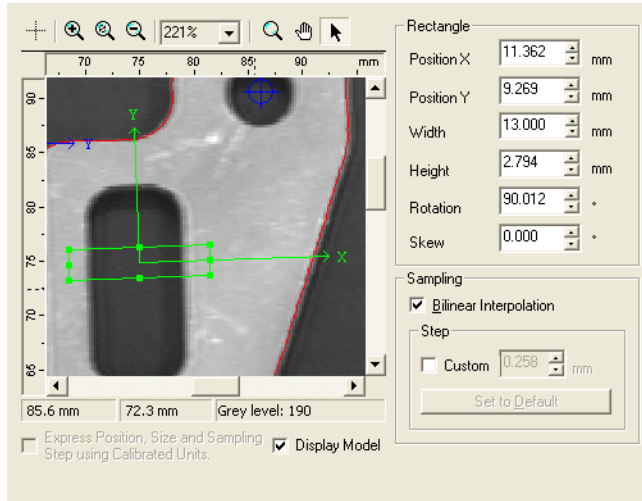


Figure 99 Adjusting the Size, Position and Orientation of the Caliper Rectangle

Note: The Caliper must always be as perpendicular as possible to the edges to be measured. For the current example, the tool had to be rotated to 90 degrees and the skew was left at its default value of 0 degrees.

- Go to the **Transform** panel. The image shown is that of the current instance of the object with the tool transposed to the correct position on this instance. To see the tool applied to other instances, leave the Caliper window open and click **Execute Sequence** in the Process Manager. Yellow lines indicate where the Caliper has detected potential edges. Edge selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

Configuring the Caliper

The Caliper locates potential edge pairs. You will set up constraints that will allow the Caliper to measure only valid edge pairs, that is, those that meet the criteria you have determined.

The display in the Configuration panel represents the part of the Image bounded by the Rectangle. The Caliper processes information within the Image along pixel lines, called projection paths, parallel to the Rectangle's Y-axis. In the display you can observe the following:

- Dotted yellow lines indicate potential edges.
- The blue curve shows a graphical representation of a projection that averages each column of pixels in the Rectangle to create a one-dimensional signal. Sharp changes on this curve indicate potential edges.
- The green magnitude curve is the first derivative of the projection curve. High and low peaks in this curve identify potential edges.
- Horizontal red lines indicate the magnitude threshold for the detection of potential edges.

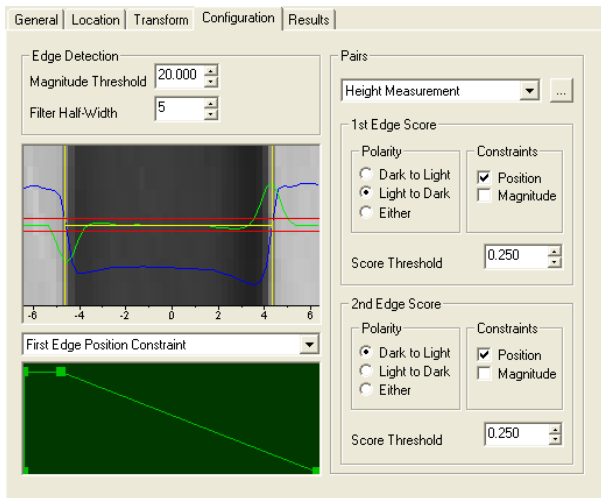


Figure 100 Caliper Tool — Configuration Panel

In this example, only one measurement pair is needed so the default pair named *Pair0* will be used. The constraints used for both edges will be polarity and position constraints. These constraints will be used to score each edge.

Polarity indicates changes in light values, from left to right, within the Rectangle.

Position constraints score each edge according to its position as defined by the Position Constraint histogram. **Magnitude** constraints, not used in this tutorial, score each edge according to its magnitude as defined by the Magnitude Constraint histogram.

The Caliper attributes a score to each edge pair and to each individual edge. Scores range from 0 to 1 with the minimum acceptable score being determined by the **Score Threshold**, set by default at 0.250. Histogram functions for each constraint determine how the Caliper scores edges.

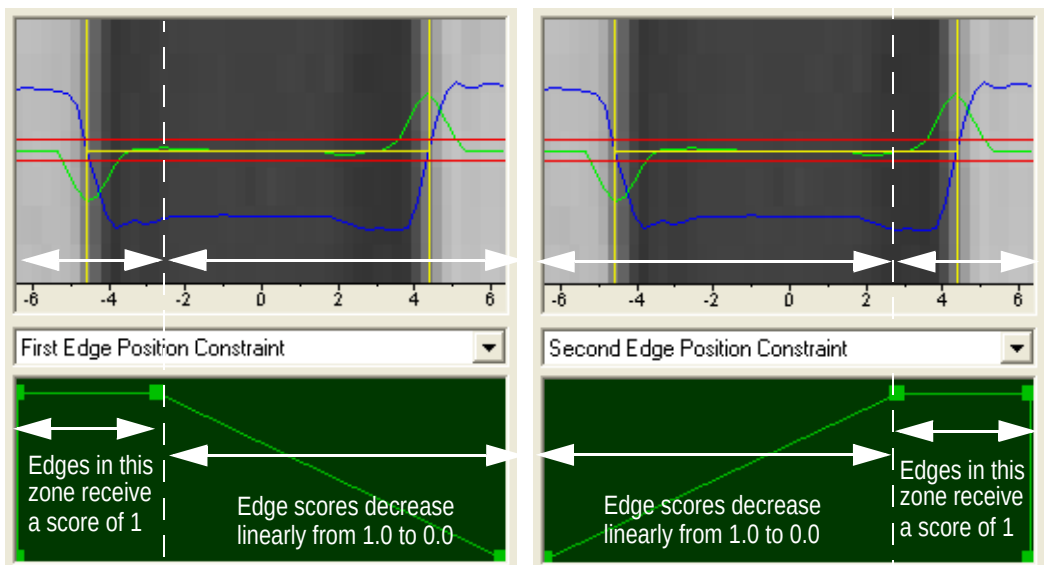


Figure 101 Adjusting Position Constraint Histograms

1. In the **1st Edge Score** frame, select **Polarity** as **Light to Dark** and enable the **Position** check box.
2. In the **2nd Edge Score** frame, select **Polarity** as **Dark to Light** and enable the **Position** check box.

Next you must set up the Position constraint histogram for each edge, in the bottom left window of the **Configuration** panel.

3. Use the combo box above the histogram window to select a constraint.
4. Use the mouse to drag the handles in the histogram window. Set the 1st and the 2nd edge position constraints as illustrated in [Figure 101](#).
5. Click **Execute**. The yellow dotted lines become solid and are joined by a solid horizontal line to indicate that the pair has been detected and measured.

Caliper Results

You will now observe the Caliper's results for various instances of the object.

Access the Caliper Results panel to observe the following:

- Valid edges are displayed as parallel vertical yellow lines.
- The Caliper measure is displayed as a horizontal yellow line, perpendicular to the edges.
- The Pair Count box indicates that the Caliper was configured to measure one pair.
- The Coordinate System combo box indicates that the positions are expressed with respect to the World coordinate system.

The grid of results indicates:

- The Caliper measure, named Size, indicated in calibrated units.
- The score for the edge pair as well as for each edge.
- Use the scrollbar to view the entire grid of results.

Now that the Caliper is properly configured to measure the rectangular hole feature, you will want to test and observe the Caliper on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the next step.

Adding Code for the Caliper

You will finish up by adding code to your application form that will display the measurement of the Caliper in the interface. You will also add a line marker on the display to show the measurement.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Caliper Control**, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
...
HSLOCATORLib.HSLocator lLocator;
HSCALIPERLib.HSCaliper lWidthCaliper;

string lMarkerName;

InspectButton.Enabled = false;

lManager = ApplicationControl.ProcessManager;

lLocator = (HSLOCATORLib.HSLocator)lManager.
get_Process("Locator");
lWidthCaliper = (HSCALIPERLib.HSCaliper)lManager.
get_Process("WidthCaliper");
...
```

This will create a reference that will be used for programmatic access to the WidthCaliper process instantiated in the HexSight Application Control.

2. Modify the existing line, shown in *italic*, for the one below it:

```
lManager.Execute( "Acquisition", "Locator" );
lManager.Execute( "Acquisition", "WidthCaliper" );
```

This will execute the entire sequence of processes.

3. Now, to display the Caliper's measurement in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```
...
TranslationYText.Text = string.Format("{0:F2}",
    lLocator.get_InstanceTranslationY(0));

// Display width caliper properties and marker
if (lWidthCaliper.get_PairScore(0) > 0)
{
    lWidthCaliper.CoordinateSystem =
    HSCALIPERLib.hsCoordinateSystem.hsCoordinateSystemWorld;

    WidthText.Text = string.Format("{0:F2}",
    lWidthCaliper.get_PairSize(0));

    lMarkerName = "Width";

    ApplicationDisplay.AddLineMarker(lMarkerName,
        lWidthCaliper.get_Edge1PositionX(0),
        lWidthCaliper.get_Edge1PositionY(0),
        lWidthCaliper.get_Edge2PositionX(0),
        lWidthCaliper.get_Edge2PositionY(0), true);

    ApplicationDisplay.set_MarkerAnchorStyle(lMarkerName,
        HSDISPLAYLib.hsMarkerAnchorStyle.hsCross);

    ApplicationDisplay.set_LineMarkerConstraints(lMarkerName,
        HSDISPLAYLib.hsLineMarkerConstraints.hsLineNoEdit);

    ApplicationDisplay.set_MarkerColor(lMarkerName,
        HSDISPLAYLib.hsColor.hsRed);

    ApplicationDisplay.set_MarkerDisplayName(lMarkerName,
        true);
}
...
```

If the first pair of the Caliper (Index = 0) is found (Score > 0.250), its measurement is displayed in the WidthText text box. Using the World coordinate system, the calibrated X-Y position of the first and second edge of the pair is used to draw a non-editable line marker on the application display.

4. In case no instance is found, add lines of code shown in bold:

```
...
TranslationYText.Text = "";
WidthText.Text = "";
}
```

5. Save your work and test the application. You should see the measurement in the RectangleWidth text box of your application. You should also see the line marker you added on the Display highlighting the measurement of WidthCaliper.

Adding a Second Caliper

Using the previous steps, try adding a second Caliper that will measure the height of the rectangular hole in the hook. Add the appropriate code to display the result in the remaining text box (Rectangle Height) on your application form. Also add the code to draw another line marker on the Display.

Refer to the Application Example in the following folder:

InstallationPath\Tutorial\C#\HookInspection\Part3.

C# Tutorial 4: Blob Analyzer

In this tutorial, you will learn how to set up, configure and use a Blob Analyzer tool to find, label and analyze irregular shaped features on the part.

Modifying your Application Form

First you need to add two text boxes to your application so you can observe the measurement output by the Blob Analyzer.

1. On your application form, add two text boxes. Name the first one **DiameterText** and the second **OffsetText** and add appropriate labels. Set up your form as illustrated in [Figure 102](#).

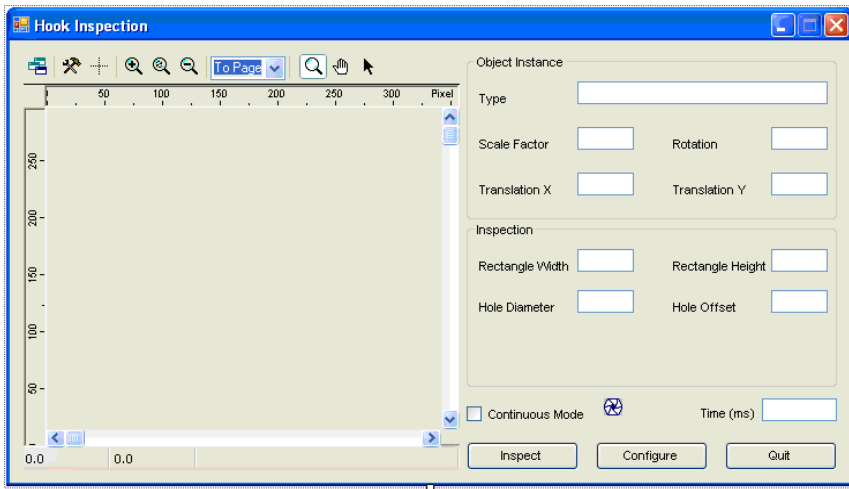


Figure 102 C# Form for Tutorial 4

Placing the Blob Analyzer

Return to the Process Manager (right-click the HexSight Application Control icon, select **ActiveX -Edit**). Click **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add a **HSBlobAnalyzer** process from the **HexSight Inspection Tools** folder of the Process Manager. Set the tool's General panel as follows:

1. Enter *HoleBlob* as the **Process Name** (instead of HSBlobAnalyzer4).
2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Leave the **Create Blob Image** check box disabled.
4. Enable the **Apply Transform to a Model-defined Location** check box.
5. Select **Hook** from **Model** view as the **Input Model Scene**.
6. Enable the **Lock Location Transform on a Selected Instance** check box.
7. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Blob Analyzer tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

8. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Blob Analyzer will be applied to an area on the object bounded by the Blob Analyzer's Rectangle, which appears in green. The Blob Analyzer will be used to inspect the diameter and offset of the circular hole on the part.

9. Place the Rectangle on the part as shown in the illustration. Click the center of the Rectangle and drag it into position. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position with the text boxes if needed. Once the Rectangle is placed, click **Execute**.
10. Go to the **Transform** panel. The image shown is that of the current instance of the object as located in the second image of the emulation database (if you clicked twice on **Execute Sequence** before creating the Blob Analyzer tool). The red target marker indicates the center of the blobs detected by the Blob Analyzer. Blob selection criteria will be

set up in the next step. Open the **Configuration** panel and move on to the next step.

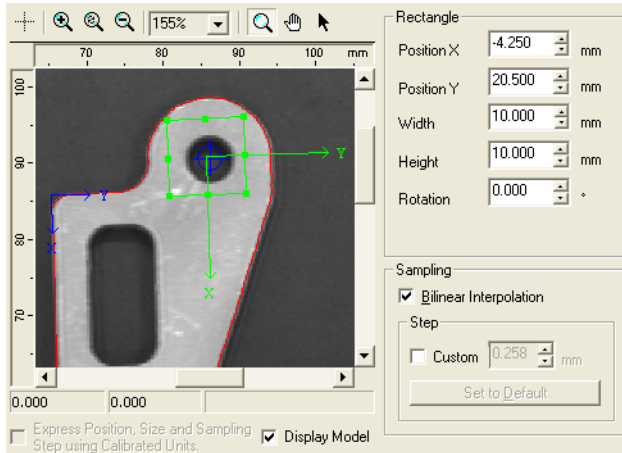


Figure 103 Adjusting the Size, Position and Orientation of the Blob Analyzer Rectangle

Configuring the Blob Analyzer

You will set up constraints that will allow the Blob Analyzer to find only valid blobs, that is, those that meet the criteria you have determined.

The display in the Configuration panel represents the part of the Image bounded by the Rectangle. The Blob Analyzer processes information within this Rectangle. In the display you can observe the following:

- The blob, as defined by the Image Segmentation Mode, is represented in green.
- A blob marker appears in red in the center of each blob.
- When displaying the Pixel Weight Image, you can see the weight value of each pixel as defined by the graphical function of the Image Segmentation box. White pixels have a weight of 1 while black pixels have a weight of 0. Pixels having a weight between 0 and 1 are displayed with a proportional greylevel.

1. First, set the Blob Constraints: **Minimum Area** to 0 and **Maximum Area** to 100, which corresponds to the area defined by the Rectangle.

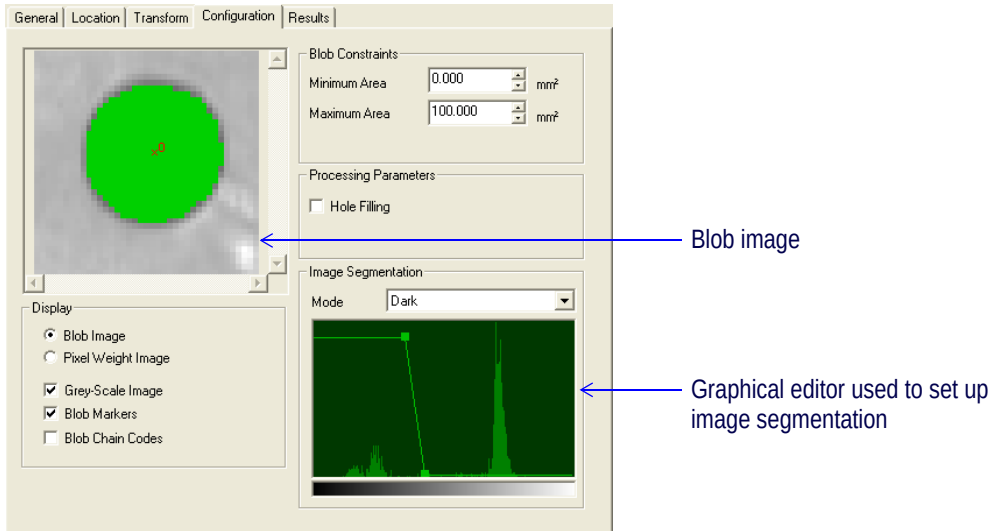


Figure 104 Blob Analyzer — Configuration Panel

Now you must set the segmentation function using the graphical editor in the Image Segmentation box. The graphic shows a histogram of the distribution of pixels inside the Blob Analyzer's Rectangle. Black (0) is at the far left and white (255) at the far right. In this example, there is a small peak in the black portion (around 58) representing the blob, and a higher peak (around 186) representing the part color. You want to include some pixels in the transition, with a weight function proportional to the grey-scale value.

2. Since we are looking for a dark blob on a white background, set the **Mode** combo box to **Dark**.
3. Use the mouse to drag the handles in the graphic function editor. Set the Image Segmentation weight function as illustrated in [Figure 105](#).
4. Click **Execute**. The segmented blob now corresponds to the hole in the part.

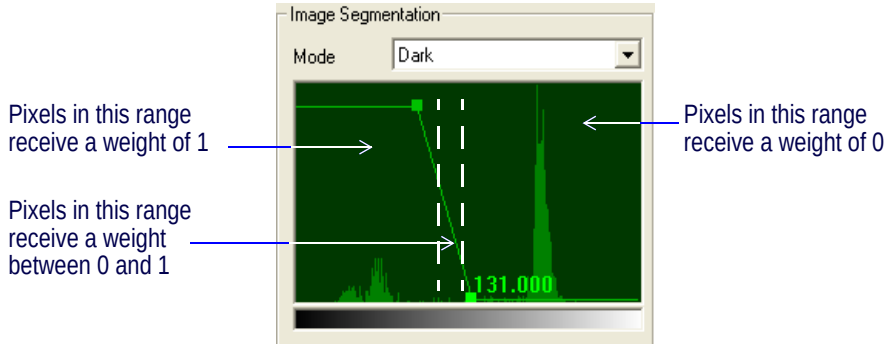


Figure 105 Image Segmentation Function Editor

Blob Analyzer Results

You will now observe the Blob Analyzer's results for various instances of the object.

1. In the **Results** tab, enable the **Sort Results** check box and select **Area** with the **Sort by** combo box.
2. Click **Execute**.

Use the Blob Analyzer Results panel to observe the following:

- The Blob Count box indicates that the Blob Analyzer has found one blob.
- An Image is displayed with a blob marker indicating the center of each blob found for this instance of the object.
- The Coordinate System combo box selection indicates that the positions are returned with respect to the World coordinate system.
- A list of the selected types of blob properties that are computed: General, Perimeter, Intrinsic Inertia, etc.

The grid of results shows:

- The position of the blob according to the selected coordinate system.
- The blob area indicated in calibrated units.

- The properties computed for the blob. To view the complete set of properties, use the grid's tabs and scrollbar.

Now that the Blob Analyzer is properly configured to inspect the circular hole feature, you will want to test and observe the Blob Analyzer on other instances of the object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the final step.

Adding Code for the Blob Analyzer

You will finish up by adding code to your application form that will display the measurements of the Blob Analyzer in the interface. You will also add a target marker on the display to show the position and diameter of the hole.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Blob Analyzer Control**, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
...
HSCALIPERLib.HSCaliper lHeightCaliper;
HSBLOBANALYZERLib.HSBlobAnalyzer lHoleBlob;

string lMarkerName;
float lHoleDiameter;
float lHoleOffset;
...
lHeightCaliper = (HSCALIPERLib.HSCaliper)lManager.
get_Process("HeightCaliper");
lHoleBlob = (HSBLOBANALYZERLib.HSBlobAnalyzer)lManager.
get_Process("HoleBlob");
...
```

This will create a reference that will be used for programmatic access to the hole blob process instantiated in the HexSight Application Control.

2. Modify the existing line, shown in italic, for the one below it:

```
lManager.Execute( "Acquisition", "HeightCaliper" );
```

```
lManager.Execute( "Acquisition", "HoleBlob" );
```

This will execute the entire sequence of processes.

- Now to display the Blob Analyzer's measurements in the text boxes and to draw the marker on the application display, insert the lines of code shown in bold:

```
// Display height caliper properties and marker

if ( lHeightCaliper.get_PairScore( 0 ) > 0 )
{
    ...
}

// Display hole blob properties and marker

if ( lHoleBlob.BlobCount > 0 )
{
    lHoleDiameter = System.Convert.ToSingle(
        System.Math.Sqrt( 4 * lHoleBlob.get_BlobArea( 0 ) /
            System.Math.PI ) );

    DiameterText.Text = string.Format( "{0:F2}",
        lHoleDiameter );

    lHoleBlob.CoordinateSystem = HSBLOBANALYZERLib.
hsCoordinateSystem.hsCoordinateSystemObject;

    lHoleOffset = System.Convert.ToSingle( System.Math.Sqrt(
        Math.Pow( lHoleBlob.get_BlobPositionX( 0 ), 2 ) +
        Math.Pow( lHoleBlob.get_BlobPositionY( 0 ), 2 ) ) );

    OffsetText.Text = string.Format( "{0:F2}", lHoleOffset );

    lHoleBlob.CoordinateSystem = HSBLOBANALYZERLib.
hsCoordinateSystem.hsCoordinateSystemWorld;
```

```

lMarkerName = "Hole";

ApplicationDisplay.AddTargetMarker( lMarkerName,
    lHoleBlob.get_BlobPositionX( 0 ),
    lHoleBlob.get_BlobPositionY( 0 ), lHoleDiameter / 2,
    true );
ApplicationDisplay.set_TargetMarkerConstraints(
    lMarkerName, HSDISPLAYLib.hsTargetMarkerConstraints.
    hsTargetNoEdit );
ApplicationDisplay.set_MarkerColor( lMarkerName,
    HSDISPLAYLib.hsColor.hsRed );
}

```

4. In case no instance is found, add the lines of code shown in bold:

```

...
HeightText.Text = "";
DiameterText.Text = "";
OffsetText.Text = "";
}

```

If the blob is found, the offset of the hole from the origin of the Object coordinate system and the diameter of the hole are computed and displayed in the OffsetText and DiameterText text boxes. Using the World coordinate system, the calibrated X-Y position of the blob is used to draw a non-editable target marker on the application display.

5. Save your work and test the application. You should see the computed diameter and offset of the hole in the text boxes of your application. You should also see the target marker you added on the Display, highlighting the position and the diameter of the hole.

C# Tutorial 5: Pattern Locator

In this tutorial, you will learn how to set up, configure and use a Pattern Locator tool to find and locate instances of the “HS” printed on the part.

Modifying your Application Form

First you need to add one text box to your application so you can display the result output by the Pattern Locator.

1. On your application form, add one text box. Name it **LabelText** and add an appropriate label. Set up your form as illustrated in [Figure 106](#).

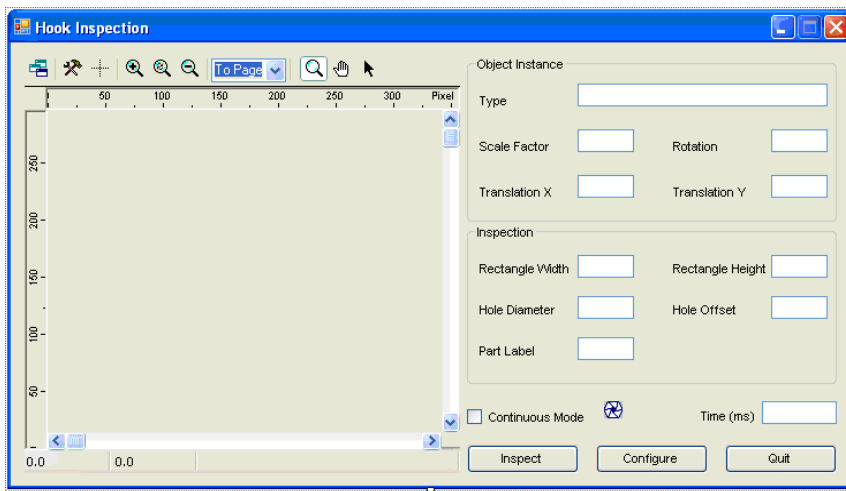


Figure 106 C# Form for Tutorial 5

Placing the Pattern Locator

Select the HexSight Application Control on the form. With the right-click menu, select **ActiveX -Edit**. Click on **Execute Sequence** twice to locate the object in the second image of the emulation database. Using the Process Manager, add the **HSPatternLocator** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *LabelLocator* as the **Process Name** (instead of HSPatternLocator5).

2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Pattern Locator tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the **Location** panel.

The Location display shows the Model View. The Pattern Locator will be applied to an area on the object bounded by the tool's Rectangle. The Pattern Locator will be used to verify the presence of the "HS" label printed on the part.

8. Place the Rectangle on the part as shown in the illustration. Click the center of the Rectangle and drag it to the center of the "HS" label. Use the corner and side handles to adjust the width and height of the search rectangle to fit the label. Zoom in on the label and by clicking and dragging the arrow on the tool's X-axis, adjust the rotation to fit the orientation of the label. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.

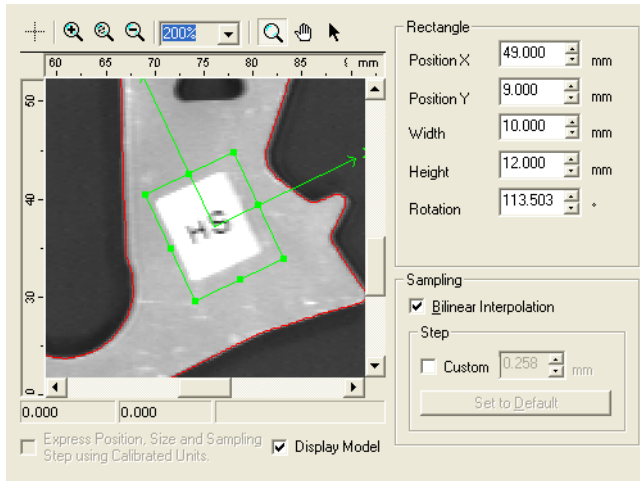


Figure 107 Adjusting the Size, Position and Orientation of the Location Rectangle

9. Go to the **Transform** panel. The image shown is that of the current instance of the object as located in the second image of the emulation database (if you clicked twice on **Execute Sequence** before creating the Pattern Locator tool). To see the tool applied to the other instances, leave the Pattern Locator open and click **Execute Sequence** in the Process Manager. The Pattern Locator will be configured to check for the presence of the label in the next step. Open the **Pattern** panel and move on to the next step.

Placing the Pattern Rectangle

You will set up the rectangle for defining the pattern that you want the Pattern Locator find within the Image. In this tutorial, the Pattern Locator will be used to verify the presence of the "HS" label printed on the part.

1. Place the Rectangle on the part as shown in the illustration.
2. Click the center of the rectangle and drag it to the center of the "HS" label. Use the corner and side handles to adjust the width and height of the Pattern Rectangle to fit the label.

3. Zoom in on the label and by clicking and dragging the arrow on the tool's X-axis, adjust the rotation to fit the orientation of the label.
4. Readjust the size of the Pattern Rectangle to include only the "HS" characters in the pattern. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.
5. Open **Configuration** panel and move on to the next step.

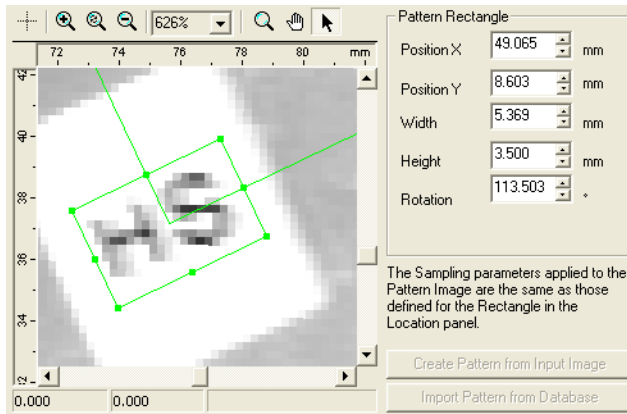


Figure 108 Adjusting the Size, Position and Orientation of the Pattern Rectangle

Configuring the Pattern Locator

You will set up the parameters of the tool in order to detect the grey-scale pattern on the part. Select the Configuration tab and set the following parameters:

1. Set the **Instance To Find** value to 1. This sets the maximum number of pattern instances to be found.
2. Set the **Match Threshold** value to 0.500. This sets the minimum strength score required to accept a valid match
3. Leave the **Automatic Coarseness Selection** check box enabled. The Pattern Locator will automatically determines the **Search Coarseness** and **Positioning Coarseness** levels.

4. Click **Execute**.

Pattern Locator Results

You will now observe the Pattern Locator's results for various instances of the object. Select the **Results** tab to observe the following:

- Each valid pattern is displayed with a blue rectangle marker indicating the position of the pattern detected in the Search Rectangle.
- The left display shows the Image of the Search Rectangle.
- The right display shows the Image of the reference pattern.

The Coordinate system combo indicates that the positions are returned with respect to the World coordinate system.

The grid of results indicates:

- The strength of the match.
- The position of the pattern according to the selected coordinate system.
- The rotation of the pattern in calibrated rotation units.

Use the scrollbar to view the entire grid of results.

Now that the Pattern Locator is configured properly to check the presence of the “HS” label, you will want to test and observe the Pattern Locator on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the final step.

Adding Code for the Pattern Locator

You will finish up by adding code to your application form that will display the result of the pattern search in the interface. You will also add a point marker on the display to show the position of the label located by the Pattern Locator.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Pattern Locator Control**, select it and click **OK**. Double-click **Inspect**.

2. Insert the lines of code shown in bold:

```
...
HSBLOBANALYZERLib.HSBlobAnalyzer lHoleBlob;
HSPATTERNLOCATORLib.HSPatternLocator lLabelLocator;
...
lHoleBlob = (HSBLOBANALYZERLib.HSBlobAnalyzer)
lManager.get_Process("HoleBlob");
lLabelLocator = (HSPATTERNLOCATORLib.HSPatternLocator)
lManager.get_Process("LabelLocator");
```

This will create a reference that will be used for programmatic access to the LabelLocator process instantiated in the HexSight Application Control.

3. Modify the existing line, shown in italic, for the one below it:

```
lManager.Execute( "Acquisition", "HoleBlob" );
lManager.Execute( "Acquisition", "LabelLocator" );
```

This will execute the entire sequence of processes.

4. Now to display the result of the search in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```
if (lHoleBlob.BlobCount > 0)
{
    ...
}

// Display label locator properties and marker

if (lLabelLocator.MatchCount > 0)
{
    LabelText.Text = "Present";

    lLabelLocator.CoordinateSystem = HSPATTERNLOCATORLib.
hsCoordinateSystem.hsCoordinateSystemWorld;
```

```

lMarkerName = "HSLabel";

ApplicationDisplay.AddPointMarker(lMarkerName,
    lLabelLocator.get_MatchPositionX(0),
    lLabelLocator.get_MatchPositionY(0), true);

ApplicationDisplay.set_MarkerDisplayName(lMarkerName,
    true);
ApplicationDisplay.set_MarkerColor(lMarkerName,
    HSDISPLAYLib.hsColor.hsBlue);
}
else
{
    LabelText.Text = "Absent";
}

```

5. In case no instance is found, add the line of code shown in bold:

```

...
OffsetText.Text = "";
LabelText.Text = "";
}

```

6. Save your work and test the application. You should see the presence/absence of the "HS" label in the text box of your application. You should also see the point marker you added on the Display, highlighting the position of the label.

C# Tutorial 6: Edge Locator

In this tutorial, you will learn how to set up, configure and use two Edge Locator tools to measure non-parallel edges on the part.

Modifying your Application Form

First you need to add one text box to your application so you can observe the result output by the Edge Locator processes.

1. On your application form, add one text box. Name it **PartWidthText** and add appropriate label. Set up your form as illustrated in [Figure 109](#).

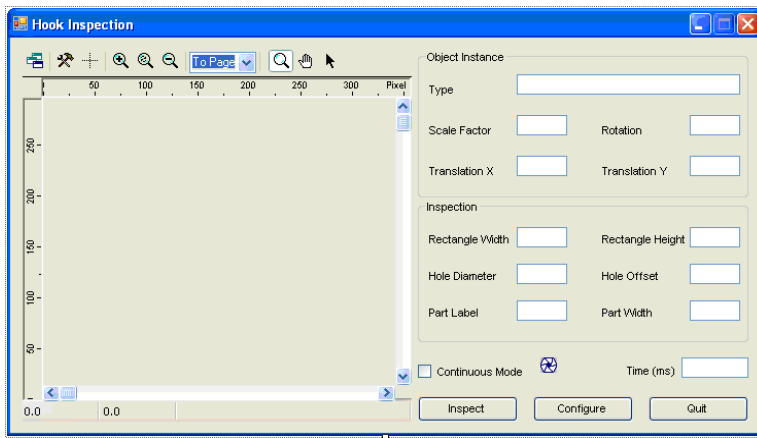


Figure 109 C# Form for Tutorial 6

Placing the Edge Locator

Select the HexSight Application Control on the form. With the right-click menu select **ActiveX -Edit**. Click **Execute Sequence** twice to locate the object in the second Image of the emulation database. Using the Process Manager, add the **HSEdgeLocator** process from the **HexSight Inspection Tools** folder. Set the tool's General panel as follows:

1. Enter *PartLeftEdge* as the **Process Name** (instead of HSEdgeLocator6).

2. Select **Image** from **Acquisition** view as the **Input Image**.
3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.

The above settings will make your application fully model-based (automatic model-based mode). The Edge Locator tool is defined on the model and will be automatically transposed to fit the located instance, whatever its scale, rotation or displacement.

7. Click **Execute**. Open the Location panel.

The Location display shows the Model View. The Edge Locator will be applied to an area on the object bounded by the tool's Rectangle, which appears in green. The Edge Locator will be used to measure the position of the left edge of the part.

8. Place the Rectangle on the part as shown in the illustration. Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees (the Y-axis is used to adjust the skew). Click the center of the Rectangle and drag it into position. Use the corner and side handles to adjust the width and height of the Rectangle. Fine-tune the position with the text boxes if needed. Once it is placed, click **Execute**.

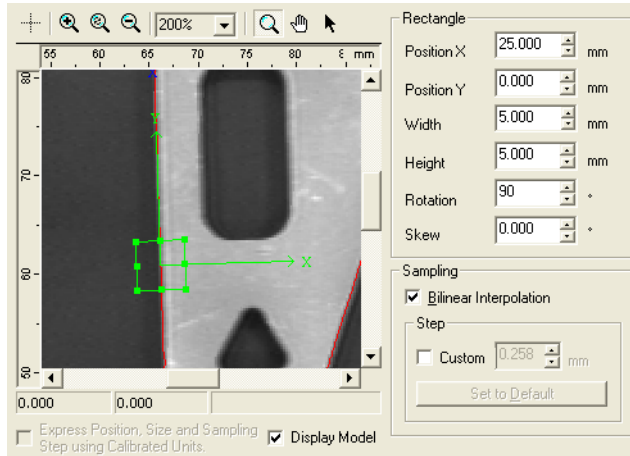


Figure 110 Adjusting the Size, Position and Orientation of the Edge Locator Rectangle

Note: The Edge Locator must always be as perpendicular as possible to the edges to be measured. For the current example, the tool had to be rotated to 90. Here the skew was left at its default value of 0 degrees.

9. Go to the **Transform** panel. The Image shown is that of the current instance of the object, as located in the second Image of the emulation database (if you clicked on **Execute Sequence** twice before creating the Edge Locator tool, as indicated previously). To see the tool applied to other instances, leave the Edge Locator window open and click **Execute Sequence** in the Process Manager. Yellow lines indicate where the Edge Locator has detected potential edges. Edge selection criteria will be set up in the next step. Open the **Configuration** panel and move on to the next step.

Configuring the Edge Locator

The Edge Locator detects potential edges. You will set up constraints that will allow the Edge Locator to locate only valid edges, that is, those that meet the criteria you have determined.

The graphic display in the Configuration panel represents the image information contained in the Edge Locator Rectangle:

- Yellow lines indicate detected edges.
- The blue curve shows a graphical representation of a projection that averages each column of pixels in the Rectangle to create a one-dimensional signal. Sharp changes on this curve indicate potential edges.
- The green magnitude curve is the first derivative of the projection curve. High and low peaks in this curve identify potential edges.
- Horizontal red lines indicate the magnitude threshold for the detection of potential edges.

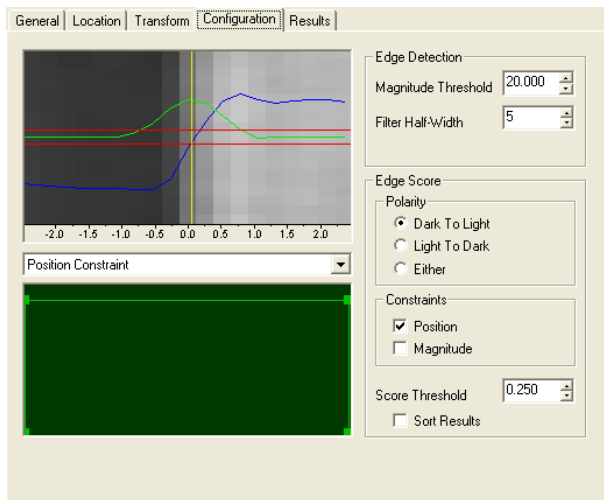


Figure 111 Edge Locator — Configuration Panel

In this example, the constraints used to score edges will be polarity and position constraints.

Polarity indicates changes in light values, from left to right, within the Rectangle.

Position constraints score each edge according to its position as defined by the Position constraint histogram. **Magnitude** constraints, not used in this

tutorial, score each edge according to its magnitude as defined by the Magnitude Constraint histogram.

The Edge Locator attributes a score to each individual edge. Scores range from 0 to 1 with the minimum acceptable score being determined by the **Score Threshold**, set by default at 0.250. Histogram functions for each constraint determine how the Edge Locator scores edges.

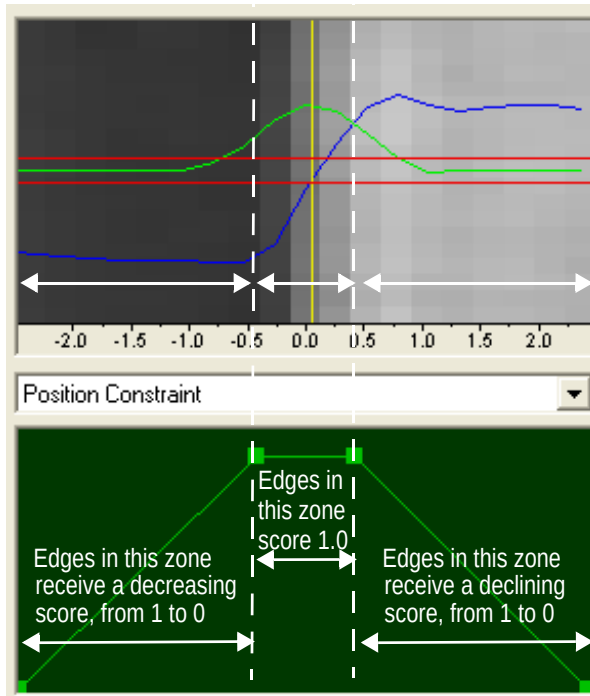


Figure 112 Position Constraint Histogram

1. In the **Edge Score** box, select **Polarity** as **Dark to Light** and enable the **Position** check box.

Next you must set up the Position Constraint histogram for the edge, in the bottom left window of the Configuration panel.

2. Use the mouse to drag the handles in the histogram editor of the selected constraint. Set the Edge Position Constraint histogram as illustrated in [Figure 112](#).

3. Click **Execute**.

Edge Locator Results

You will now observe the Edge Locator's results for various instances of the object. Access the Edge Locator Results panel to observe the following:

- Valid edges are displayed as vertical yellow lines.
- The Edge Count box indicates that the Edge Locator has found an edge.
- The Coordinate System combo box indicates that the positions are expressed with respect to the World coordinate system.

The grid of results indicates:

- The score of the found edges.
- The position of the edges, expressed with respect to the selected coordinate system.
- The rotation of the edges.
- The scores of each individual constraint.
- Use the scrollbar to view the entire grid of results.

Now that the Edge Locator is configured to locate the left edge of the part, you will want to test and observe the Edge Locator on other instances of the Object by clicking **Execute Sequence** in the Process Manager. Click **OK** and save your work before moving on to the next step.

Placing the Second Edge Locator

Using the Process Manager, add a second HSEdgeLocator process. Set the tool's General panel as follow:

1. Enter *PartRightEdge* as the **Process Name** (instead of HSEdgeLocator7).
2. Select **Image** from **Acquisition** view as the **Input Image**.

3. Enable the **Apply Transform to a Model-defined Location** check box.
4. Select **Hook** from **Model** view as the **Input Model Scene**.
5. Enable the **Lock Location Transform on a Selected Instance** check box.
6. Select **Instance Scene** from **Locator** view as the **Input Instance Scene**.
7. Click **Execute**. Open the **Location** panel.
8. Place the Rectangle on the part as shown in [Figure 113](#). Click the arrow of the X-axis and drag it to adjust the rotation to 90 degrees. Click the center of the Rectangle and drag it into position. Since the edge to be located is not orthogonal to the tool's X-axis, adjust the skew by clicking and dragging the arrow of the Y-axis to make it parallel to the right edge of the part. Fine-tune the position with the text boxes if needed. Once the Rectangle is placed, click **Execute**.

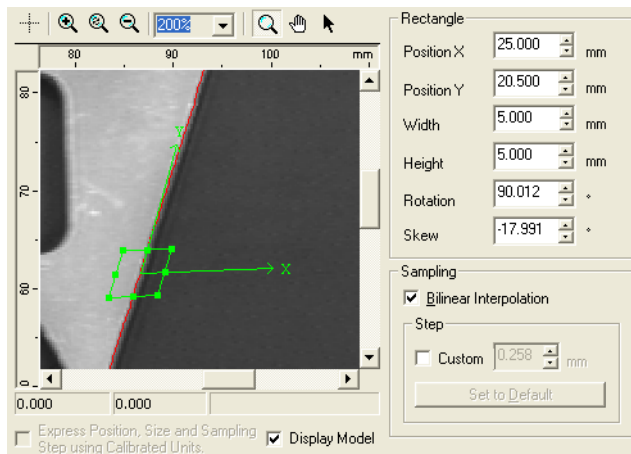


Figure 113 Adjusting Rectangle for the Hook's Right Edge

9. In the **Edge Score** box of the Configuration panel, select **Polarity** as **Light to Dark** and enable the **Position** check box.

10. Set up the Position Constraint using the histogram editor, just as that of the PartLeftEdge Edge Locator.
11. Save your work and move on to the final step.

Adding Code for the Edge Locator

You will finish up by adding code to your application form that will display the width of the part computed from the results of the two Edge Locator tools. You will also add a line marker on the display to show the measure on the part.

1. From the Microsoft Visual Studio menu bar, select *Project -> Add Reference...* Move to the COM tab and scroll down to find **HexSight Edge Locator Control**, select it and click **OK**. Double-click **Inspect** and insert the lines of code shown in bold:

```
...
HSPATTERNLOCATORLib.HSPatternLocator lLabelLocator;
HSEGELOCATORLib.HSEdgeLocator lPartLeftEdge;
HSEGELOCATORLib.HSEdgeLocator lPartRightEdge;
...
float lHoleOffset;
float lPartWidth;
...
lLabelLocator = (HSPATTERNLOCATORLib.HSPatternLocator)
lManager.get_Process("LabelLocator");
lPartLeftEdge = (HSEGELOCATORLib.HSEdgeLocator)
lManager.get_Process("PartLeftEdge");
lPartRightEdge = (HSEGELOCATORLib.HSEdgeLocator)
lManager.get_Process("PartRightEdge");
```

This will create a reference that will be used for programmatic access to the two Edge Locator processes instantiated in the HexSight Application Control.

2. Modify the existing line, shown in italic, for the one below it:

```
lManager.Execute( "Acquisition", "LabelLocator" );
lManager.Execute("Acquisition", "PartRightEdge");
```

This will execute the entire sequence of processes.

3. Now to display the measurement in the text box and to draw the marker on the application display, add the lines of code shown in bold:

```
...
else
{
    LabelText.Text = "Absent";
}

// Display part edge left and part edge right properties
// and marker

if (lPartLeftEdge.EdgeCount > 0 &&
    lPartRightEdge.EdgeCount > 0)
{
    lPartLeftEdge.CoordinateSystem = HSEdgeLocatorLib.
    hsCoordinateSystem.hsCoordinateSystemObject;
    lPartRightEdge.CoordinateSystem = HSEdgeLocatorLib.
    hsCoordinateSystem.hsCoordinateSystemObject;

    lPartWidth = lPartRightEdge.get_EdgePositionY(0) -
        lPartLeftEdge.get_EdgePositionY(0);

    PartWidthText.Text = string.Format("{0:F2}", lPartWidth);

    lPartLeftEdge.CoordinateSystem = HSEdgeLocatorLib.
    hsCoordinateSystem.hsCoordinateSystemWorld;

    lPartRightEdge.CoordinateSystem = HSEdgeLocatorLib.
    hsCoordinateSystem.hsCoordinateSystemWorld;

    lMarkerName = "Part Width";
```

```

ApplicationDisplay.AddLineMarker(lMarkerName,
    lPartLeftEdge.get_EdgePositionX(0),
    lPartLeftEdge.get_EdgePositionY(0),
    lPartRightEdge.get_EdgePositionX(0),
    lPartRightEdge.get_EdgePositionY(0), true);

ApplicationDisplay.set_MarkerAnchorStyle(lMarkerName,
    HSDISPLAYLib.hsMarkerAnchorStyle.hsCross);

ApplicationDisplay.set_LineMarkerConstraints(lMarkerName,
    HSDISPLAYLib.hsLineMarkerConstraints.hsLineNoEdit);
ApplicationDisplay.set_MarkerColor(lMarkerName,
    HSDISPLAYLib.hsColor.hsGreen);

ApplicationDisplay.set_MarkerDisplayName(lMarkerName,
    true);
}

```

4. In case no instance is found, add the line of code shown in bold:

```

...
OffsetText.Text = "";
LabelText.Text = "";
}

```

5. Save your work and test the application. You should see the part width in the Part Width text box of your application. You should also see the line marker you added on the Display highlighting the measurement.

C# Tutorial 7: Calibration

In this tutorial, you will learn how to calibrate the system to perform acquisitions of live images. Calibration is an important step that can correct certain perspective and lens distortions. It also allows you to specify the basic length units you will use throughout your entire application.

Camera Calibration

The calibration is performed in the Acquisition Device. In this example, the calibration is performed with the largest calibration target provided with your HexSight package. This target has a dot pitch of 10 mm.

1. Open the Acquisition Device Properties. Select the **Configuration** panel. On the Input sub-panel, select **Copy Configuration** and then select **Paste Configuration**.
2. In the displayed window, enter *Camera* as the new configuration name.
3. Select the newly created *Camera* configuration in the list.
4. In the **Input** sub-panel, select **Generic Frame Grabber** and click **Apply**.
5. Go to the **Calibration** sub-panel. Select **Distortion Model**. Click **Show Calibration Dialog**. The **Distortion Model** dialog is displayed (see [Figure 114](#)).

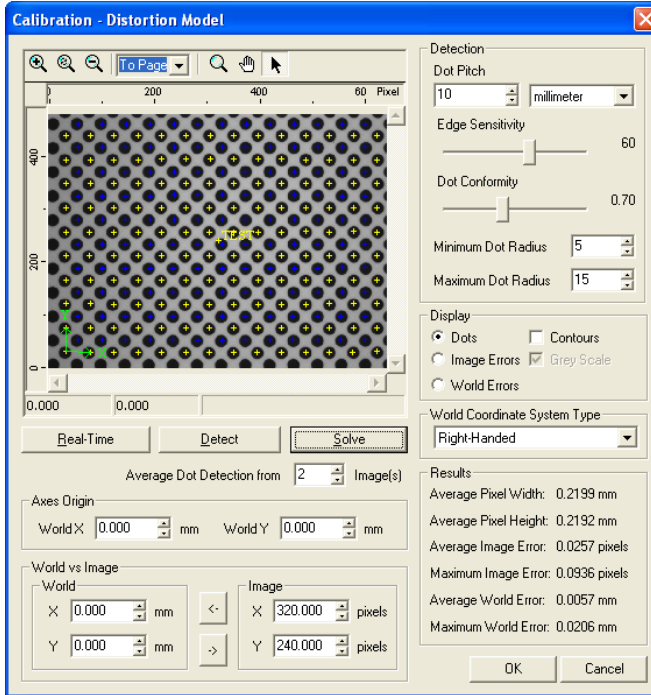


Figure 114 Distortion Model Calibration Dialog

6. Click **Real-Time** to display the image from the camera. Position the grid of dots to cover the whole area. Set the focus and other settings of your camera to obtain the best image possible. Then click **Real-Time** again to stop real time acquisitions.
7. Click **Detect** to detect the dots on the target.
8. The dot pitch is 10.0 mm. Enter the dot pitch in the **Dot Pitch** box.
9. Zoom in at the lower left corner of the display. A green coordinate system marker should be displayed in the bottom left, as in [Figure 115](#). Note that this coordinate system is displaceable programmatically or with the mouse; leave it as is for this tutorial.

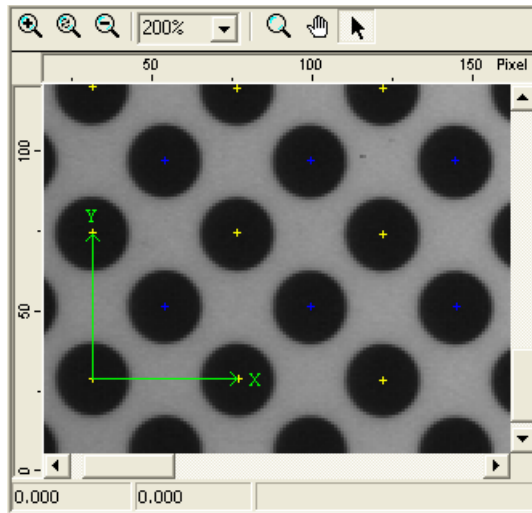


Figure 115 Zoom-in on the Coordinate Marker in the Calibration Dialog

10. Click **Solve**. The statistics appear in the **Results** frame. Click **OK**.
11. The calibration procedure is completed. The computed calibration parameters appear in the **Parameters** grid. Click **Apply** to apply the calibration to the *Camera* configuration.
12. Click **OK** to close the Acquisition Device's properties window.
13. You are now ready to run the application with live images using the Paper Hook provided with your HexSight package.

Using HexSight Displays

This Chapter describes the use and the functionalities shared by HexSight displays. Besides the HSDisplay Control, displays appear under different forms throughout the HexSight interface.

Chapter Outline

[Introduction to HexSight Displays](#)

[Elements of HexSight Displays](#)

[Toolbar](#)

[Rulers](#)

[Viewport](#)

[Status Bar](#)

Introduction to HexSight Displays

This section explains the basic use of all display screens in HexSight software. The examples are based on a display added to a project by adding a HSDisplay control. Mastering the few basic controls for display screen will enable you to work efficiently with this type of component.

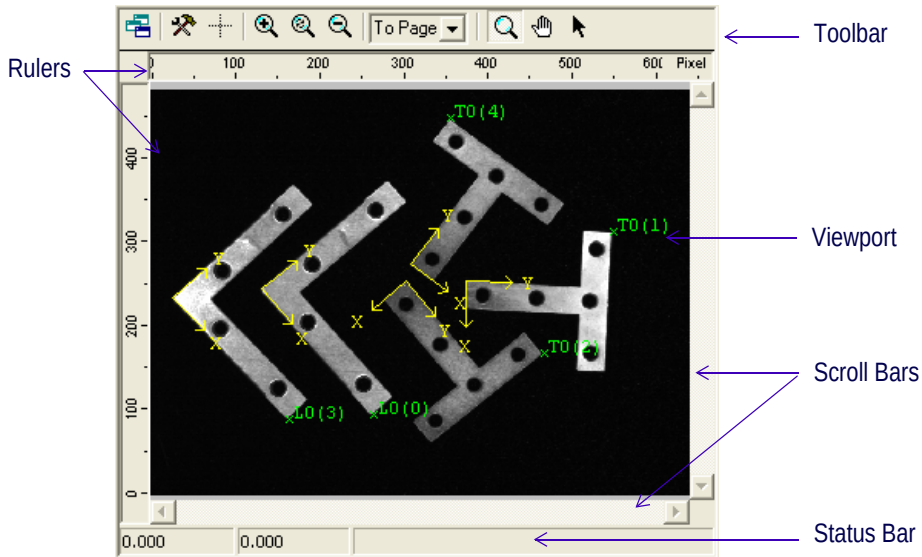


Figure 116 Typical HexSight Display Screen

- The user-interface of all HexSight tools contain one or more display screens which can serve to configure the tool or visualize results.
- Display screens can be added to a Visual Basic form in order to visualize different Images and/or Scenes generated by the HexSight application. This can be accomplished by pasting a Display control over a VB form. See the **Getting Started** chapter and the **Tutorials** for more information on this topic.
- [Figure 116](#) shows a typical HexSight display screen. The **Elements of HexSight Displays** section presents the different elements that constitute such a screen.

Elements of HexSight Displays

Toolbar

The display toolbar contains three distinct areas: The tool area, the zoom area and the mode area.

Tool Area

This area is composed of three buttons as shown in the next figure.

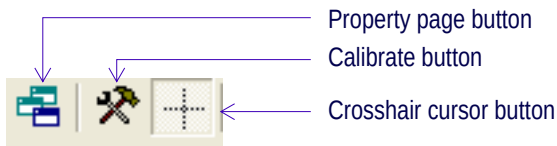


Figure 117 Tool Area Components

The **Property Page** button displays the property page of the display control.

- The parameters that appear in this property page can be modified programmatically only if the display screen originates from a Display control pasted on a Visual Basic form.
- In other words, you cannot programmatically modify the properties of a display screen that is in the a HexSight interface.

The **Calibrate** button specifies if the Acquisition Device calibration is applied on the image prior to displaying it in the viewport. For more information on the calibration process, consult the **Acquisition Device** chapter.



Figure 118 Cross Hair Cursor

The **Crosshair Cursor** button displays the mouse pointer as a crosshair cursor. This is practical for viewing a very precise location in the image.

Zoom area

The zoom area controls let you change the zoom factor of the image in the display.



Figure 119 Zoom Area Components

- The **Zoom In** the zoom factor is multiplies the zoom factor by 2, therefore enlarging the displayed image twice, while the viewport remains the same size.
- The **Zoom Back** button restores the view displayed prior to the last zoom change.
- The **Zoom Out** button divides the zoom factor by two.
- Use the **Zoom Select** Combo Box button to select among eight different zoom factor values to apply, including a special selection that makes the Image and/or Scene fit the viewport size.

Mode Area

The mode area buttons are used to specify the effect of a left mouse button click when the mouse pointer is positioned over the viewport.

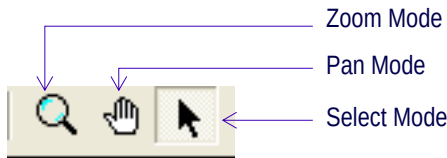


Figure 120 Mode Area Buttons

- If your mouse has a wheel, its action results in a zoom in or zoom out no matter which mode is selected.
- The **Zoom Mode** changes the appearance of your cursor to a magnifying glass. In this case, when the left mouse button is clicked, the zoom factor is doubled. If the left mouse button is clicked while the **Ctrl** key is down, the zoom factor is divided by two.
- The **Pan Mode** changes the appearance of your cursor to a hand. Use this mode to is used to shift the displayed Image to a new location in the viewport.
- The **Select Mode** changes the appearance of your cursor to an arrow. In this mode you can use the cursor to modify the position as well as some of the parameters of different markers that may be displayed in the viewport. You can also use it to select scene objects, for example in the Model Editor interface. a

Rulers

The rulers show the dimensions of the displayed image and the unit lengths of the Image or Scene.

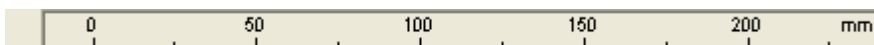


Figure 121 Display Screen Ruler

Viewport

The viewport is the drawing zone of the display screen. This is where the image is displayed. With the mouse pointer positioned over the viewport, right-click to obtain the context menu illustrated in [Figure 122](#). The selections available in this menu perform exactly the same operations as the buttons in the tool, zoom and mode areas (see previous sections).

The contents of the context menu can differ in the displays contained in the HexSight user-interface.

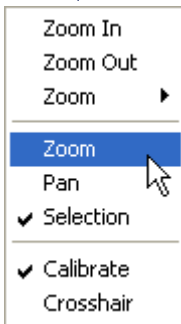


Figure 122 Viewport Context Menu

Status Bar

The status bar provides information on the X and Y position of the mouse pointer in the image. It gives also information, such as the greylevel value of the pixel on which the cursor is positioned.

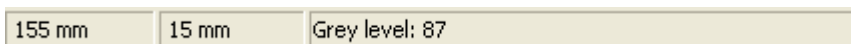


Figure 123 Status Bar

Coordinate Systems

Location and inspection tools can return results with respect to one of four coordinate systems. This allows you to use the set of references best adapted to the requirements of your applications, by selecting the coordinate system in which results will be displayed and returned.

The four coordinate systems are the World coordinate system, the Image coordinate system, the Object coordinate system, and the Tool coordinate system.

This chapter describes the four coordinate systems and the advantages of each for building applications.

Coordinate Systems Chapter Outline

[Right-Handed vs. Left-Handed Coordinate Systems](#)

[Relationship between Coordinate Systems](#)

[World Coordinate System](#)

[Image Coordinate System](#)

[Object Coordinate System](#)

[Tool Coordinate System](#)

[Unit and Angle Conversions](#)

Right-Handed vs. Left-Handed Coordinate Systems

Coordinate systems can be of two types: right-handed or left-handed. [Figure 124](#) illustrates the difference between both types.

HexSight supports both left- and right-handed coordinate systems, for both the calibrated (World) coordinate system and the Image coordinate system. The type of other HexSight coordinate systems, Object and Tool, follow the type of their associated Image or World coordinate system.

The type of coordinate system selected for a calibrated image (World) does not have to be the same as that of the acquired Image. i.e a right-handed image can be calibrated into either a left-handed or a right-handed coordinate system.

Origin

By default, the origin of a right-handed coordinate system is at bottom-left of an image whereas the origin of a left-handed coordinate system is at top-left.

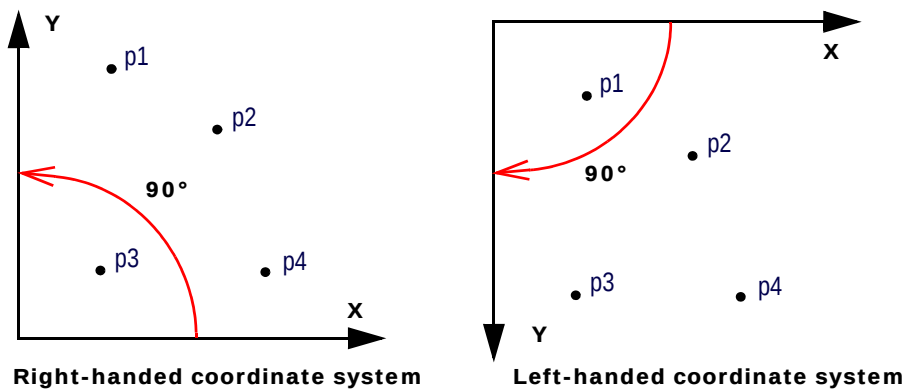


Figure 124 Right-Handed vs. Left-Handed Coordinate Systems

Rotation

Angles are measured from the X-axis towards the Y-axis. In a right-handed coordinate system, positive angles are counterclockwise; in a left-handed coordinate system, positive angles are clockwise.

Note: Angles are expressed in a range of 180° to -180° . For example, an angle of 210° , in HexSight tools, is expressed as an angle of -150° .

Relationship between Coordinate Systems

The coordinate systems can be described as layered in physical space. The World coordinate system (WCS) occupies the highest layer of level since it physically contains the other coordinate systems. There can only be one WCS,

containing more than one instance of the other coordinate systems. The relationship between the World coordinate and the Image coordinate system (ICS) is defined by the calibration procedure. See the **Acquisition Device** chapter.

- Though there is usually only one ICS, more than one instance of an ICS can occur if two independent input devices, cameras on different frame grabber channels for example, are used for an application.
- The Object coordinate system (OCS) is unique for each Model defined. The OCS is defined with respect to the WCS of the Image from which the Model is built.

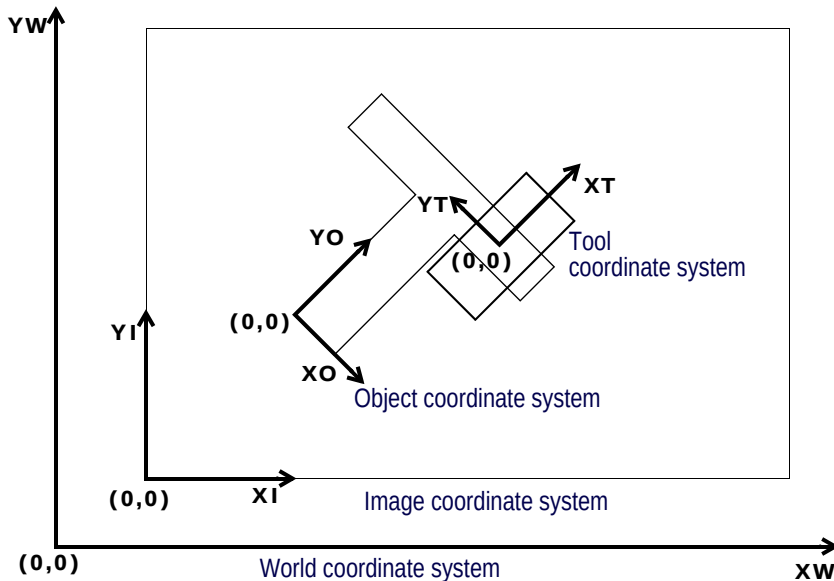


Figure 125 Relationship Between HexSight Coordinate Systems

World Coordinate System

The World coordinate system is especially useful for guidance applications. In such a case, the origin can be defined with respect to the robot's reference

coordinate system. This can also be useful for applications where more than one camera is used: objects found by each camera are thus located and defined within a same coordinate system.

- The World coordinate system is also described as the calibrated coordinate system.

Characteristics

Type: Can be right- or left-handed. The type is set at calibration.

Units: User-selected calibrated units.

Boundaries: The defined workspace or workcell in which the vision application is carried out. See Figure 124.

Image Coordinate System

This coordinate system is useful for Image processing applications.

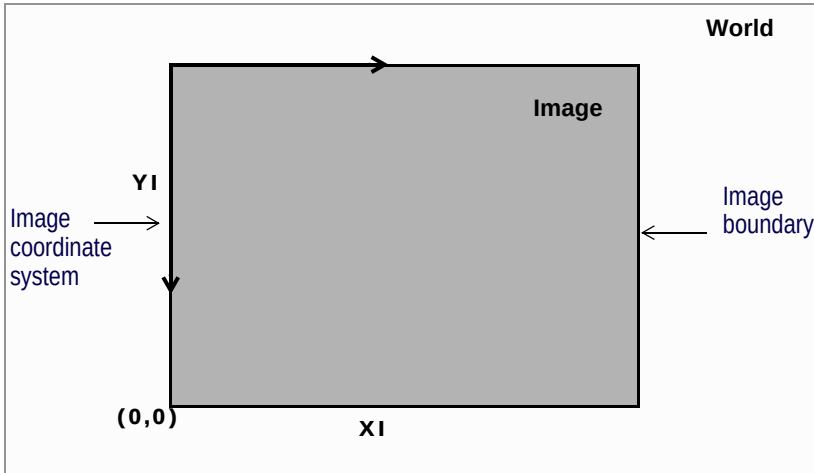


Figure 126 Image Coordinate System

Characteristics

Type: Can be right- or left-handed. The type is set at acquisition.

Units: Pixels.

Boundaries: The physical boundaries of the Image.

Object Coordinate System

The origin of the Object coordinate system is user-defined and is set up within the Model Editor.

- In an application, all instances of the same type of object use the same origin for defining any reference points that have been created in the Model as well as the position of model-based inspection tools. This is useful when you must determine a feature's location in relation to other features in the object, or in relation to a reference point on the object.
- This coordinate system is useful for quality control applications in which features must be inspected at a specific point on the object.

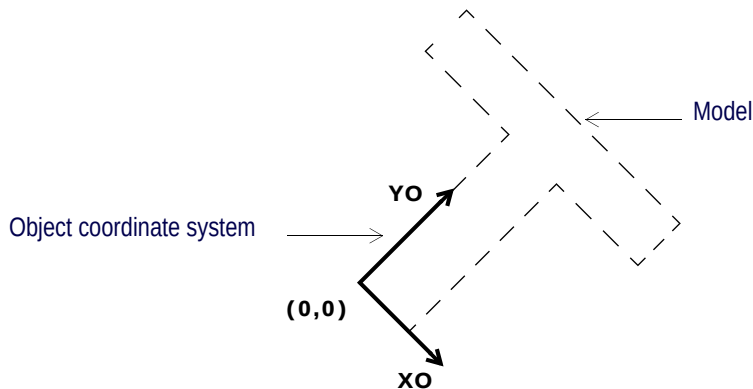


Figure 127 Object Coordinate System

Characteristics

Type: Always the same type as the World coordinate system.

Origin: Variable, defined by the user within each Model.

Units: User-selected calibrated units.

Boundaries: The physical boundaries of the Image.

Tool Coordinate System

Inspection tools carry out their action within an area typically bounded by a Rectangle or Sector region of interest. When the tool area is bounded by a Rectangle, the origin of any Tool coordinate system is the center point of the Rectangle. If the tool area is bounded by a Sector (arc tools) the origin of the coordinate system is the origin of the Sector.

- The Tool coordinate system is useful for measuring a feature that does not need to be defined or located with respect to a specific location on the object. For example, a single caliper measure across two edges on an object.

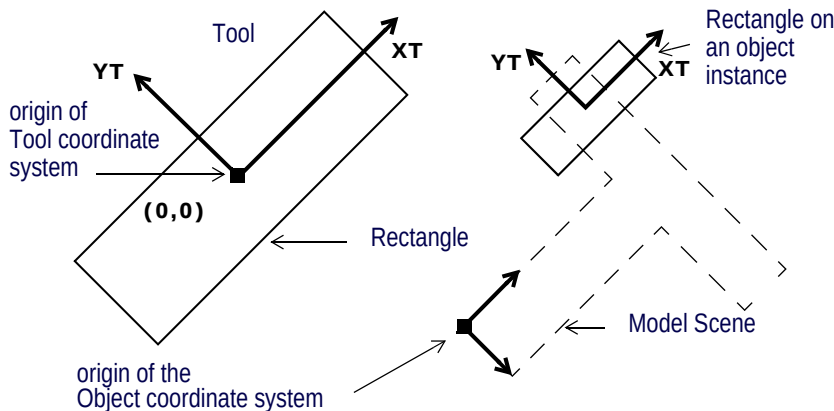


Figure 128 Tool Coordinate System

Characteristics

Type: In calibrated applications, always the same type as the system. In uncalibrated applications, always the same type as the Image coordinate system.

Origin: Fixed: at the center of an inspection tool's Rectangle region of interest or at the origin of its Sector region of interest.

Units: Calibrated units or pixels depending on the tool's configuration.

Boundaries: The physical boundaries of the Image.

Unit and Angle Conversions

All HexSight tools let you select the unit system of your choice, in either the Metric or British System.

The choices of units are:

Table 3 Available Units and Their Descriptions

Unit	Abbreviation
meter	m
centimeter	cm
millimeter	mm
micron	μm
yard	yd
foot	ft
inch	in
mil	mil
calibration units see below	c.u.

Calibration Units

Calibration units are generic length units that can be used instead of the predefined unit systems provided. A calibration unit has no fixed dimension and is defined by the user when calibrating the system.

These units can be used when no absolute measurements are required, or when they are not possible. Calibration units are dimensionless and the conversion factor is 1:1 for any other of the length units provided.

Angles

Angles of skew and rotation can be specified in either degrees or radians.

Application Control

The Application control provides the platform in which a HexSight application is built and configured. It manages the sequence of tools that make up the application and provides a runtime database of Images and Scenes.

Application Control Chapter Outline

[Accessing the Application Control](#)

[Visual Basic](#)

[Visual C++](#)

[Visual C#](#)

[Using the Application Control](#)

[Process Manager](#)

[Explorer](#)

[Application Control Properties](#)

[General Application Control Properties](#)

[Process Manager Properties](#)

[Explorer Properties](#)

Accessing the Application Control

The Application control is basically an ActiveX control that can be used in Visual Basic, Visual C++ or Visual C#.

Visual Basic

Once you add the Application Control to a Visual Basic form:

- Right-click on the Application Control icon.

- Select **Edit** to open the HexSight Application Control interface used to build an application.
- Select **Properties** to open the properties window (advanced users).

Consult **Tutorial 1** in the [VB .NET Tutorials](#) chapter for more details on adding a HexSight control to a Visual Basic form.

Visual C++

Once you have added the Application Control to a Visual C++ dialog:

- Right-click on the Application Control icon.
- Select **Edit** to open the Application Control interface used to build an application.
- Select **Properties** to open the properties window (advanced users).

Consult **Tutorial 1** in the [Visual C++ Tutorials](#) chapter for more details on adding a HexSight control to a C++ dialog.

Visual C#

Once you have added the Application Control to a Visual C# dialog:

- Right-click on the Application Control icon.
- Select **Edit** to open the Application Control interface used to build an application.
- Select **Properties** to open the properties window (advanced users).

Consult **Tutorial 1** in the [C# Tutorials](#) chapter for more details on adding a HexSight control to a C# dialog.

Using the Application Control

The Application Control provides two panels: the Process Manager and the Explorer. This section will explain how to use the Application Control for building a vision application.

Process Manager

The Process Manager manages the list of tools that you have instantiated to build your application. This panel is constituted of four main sections: a list of instantiated processes, a list of available processes, a set of command buttons and the execution time text box.

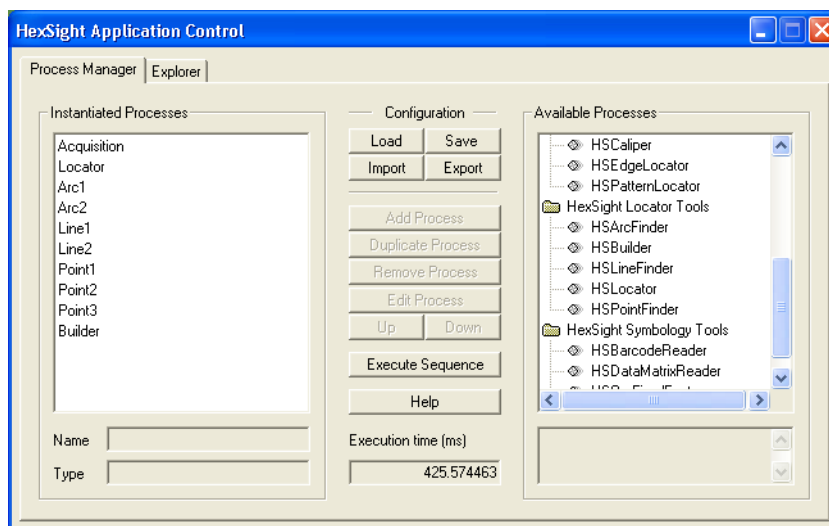


Figure 129 Application Control — Process Manager Interface

Instantiated Processes

The **Instantiated Processes** frame shows the sequence of tools you have added or instantiated to build your vision application.

This set of instantiated processes, including their properties, constitutes a *configuration*. You can save a configuration for latter use in the same

application or in another vision application. Configurations are saved to the HexSight cfg format.

Note: The order of instantiated processes is important. The execution sequence is done in process order.

Name Text Box

This displays the current name of a selected tool in the list of Instantiated Processes.

Use this text box to modify the name of the currently selected tool. The tool's name can also be modified within its properties window.

Type Text Box

This displays the read only type name of a selected tool. For example: if you renamed an instance of the Acquisition Device as *Acquisition*, its Type name remains the same i.e. **HSAcquisitionDevice**.

Command Buttons

The command buttons allow you to manage the list of instantiated processes that constitute application. Instantiated processes (also called a tools) application can be saved to a configuration file that saves both the order of the tools and the configured properties of each tool.

- **Load:** Loads an existing configuration (*.cfg) file that replaces the current list of instantiated processes.
- **Save:** Saves the current list of tools along with their properties to a configuration file (*.cfg).
- **Import:** Imports all the processes from the selected configuration file into the current list of instantiate processes.
- **Export:** Exports selected processes from the current list by saving them to a new configuration file.
- **Add Process:** Adds a process selected from the **Available Processes** list to **Instantiated Processes** list and opens the properties window for the selected process.

- **Duplicate Process:** Replicates a process selected from the **Instantiated Processes** list and appends it to the current list. The new tool instance is configured identically, except for its name.
- **Remove Process:** Removes a selected process from the application's list of **Instantiated Processes**.
- **Edit Process:** Opens the properties window for a process selected from the **Instantiated Processes** list.
- Each click of the **Up** button moves a selected process up one level higher in the **Instantiated Processes** list.
- Each click of the **Down** button moves a selected process down one level lower in the **Instantiated Processes** list.

Execute Sequence

When you first open an application either in design mode or runtime mode the runtime database of Images and Scenes is empty. To create the runtime database you must first execute the sequence of processes that make up the application.

When you click **Execute Sequence**, each tool executes its process in the order that it appears in the list of instantiated processes.

- Outputs by a tool are used as input for tools further down the list. For example: images output by the HSAcquisitionDevice can be used as input by the Locator.
- The list of processes has been executed when a new execution time appears.

HexSight Online Help

The **Help** button opens HexSight online help, as illustrated in [Figure 130](#). HexSight help contains both the **User Guide** and **Reference Manual**.

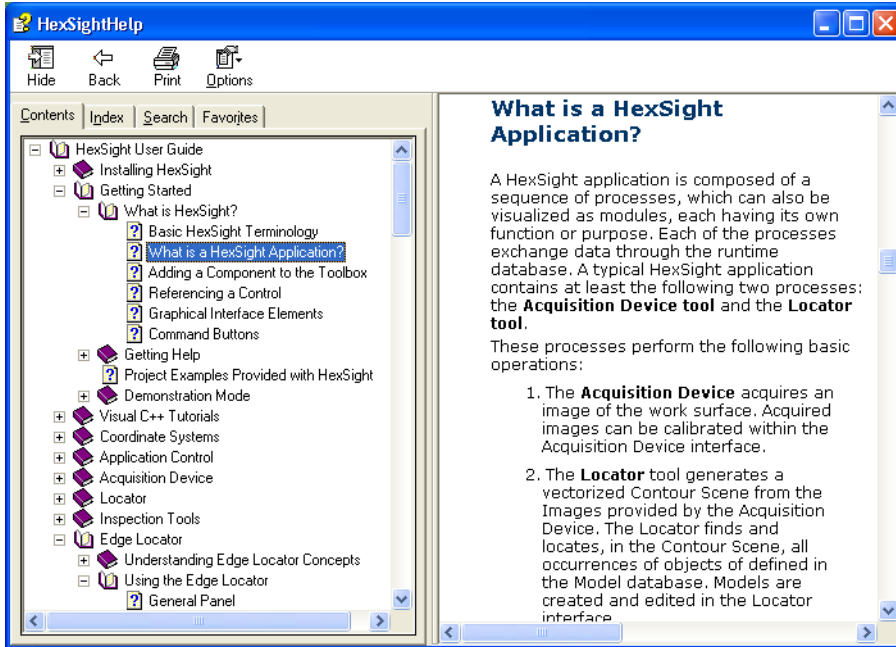


Figure 130 HexSight HTML Help Window

Execution Time Text Box

The read only **Execution Time** text box displays the time for the last execution of instantiated processes.

Available Processes

Available HexSight processes, or tools, are presented as a list of expandable folders regrouping tools by type. In many cases, more than one instance of a type of tool can be added to an application. As each new process is instantiated, or added to the application, its name appears in **Instantiated Processes** list. Below is a brief description of each process, as they appear in the Process manager.

HexSight Acquisition Tools

- **HSAcquisition Device** acquires Images from a frame grabber, a DirectShow Device (IEEE 1394) or from a database of Images.

Acquired images can be calibrated within the Acquisition Device interface.

HexSight Image Processing Tools

- **HSImage Processing Tool:** processes grey-scale images by applying arithmetic, assignment, logical, filtering, morphological or histogram operators. Users can also define and apply custom filtering operators.
- **HSSamplingTool:** samples model-defined area of an image and outputs the sample as a new Image.
- **Image Sharpness Tool:** The Image Sharpness Tool computes the sharpness of preponderant edges in a user-defined region of interest.
- **Image Histogram Tool:** The Image Histogram tool computes greylevel statistics within a user-defined region of interest.

HexSight Inspection Tools

- **HSArcCaliper:** measures distances between edges in arc-shaped areas of an object. Users can select edge pairs of interest by setting Caliper properties such as constraints, thresholds and polarities.
- **HSArcEdge Locator:** finds and locates edges in arc-shaped areas of an object, meeting user-defined criteria such as strength and polarity. The Edge Locator also scores and sorts located edges.
- **HSBlob Analyzer:** finds, labels and analyzes irregular shaped objects. This tool detects and quantifies blobs that meet user-defined criteria.
- **HSCaliper:** measures distances between edges. Users can select edge pairs of interest by setting Caliper properties such as constraints, thresholds and polarities.
- **HSEdge Locator:** finds and locates edges meeting user-defined criteria such as strength and polarity. The Edge Locator also scores and sorts located edges.

- **HSPattern Locator**: finds and locates instances of a pattern occurring within an Image. Patterns are provided by the HexSight Model control.

HexSight Locator Tools

- **HSLocator**: finds and locates objects based on the geometry of their contours. Scale factor, orientation, and position are provided for each located instance. Models used for object location are created, edited and managed through the Locator interface.
- **HSPointFnder**: The Point Finder finds and locates point features on objects and returns the angle as well as the coordinates of the found point.
- **HSLineFinder**: The Line Finder allows the user to access and manipulate Line entities. Entities can be generated by a HexSight Finder tool, the Builder tool or created programmatically within the Class Library.
- **HSArcFinder**: The Arc Finder finds and locates circular features on objects and returns the coordinates of the center of the arc, the start and end angles, and the radius.
- **HSBuilder**: The Builder creates geometric entities such as points, lines or arcs from user-defined parameters or by combining other geometric entities.

Output Tools

- **HSEvaluatorTool**: The Evaluator tool converts any result from any other tool as a Pass or Fail value, in order to trigger an event with an upcoming OutputTool with the proper decision result. Comparisons and mathematical operations are available. Multiple Evaluator tools can be used to obtain a final decision.
- **HSOutputTool**: The Output tool schedules an event to the digital IO of a device depending on a Pass/Fail result from an Evaluator tool. This event can be based on an ecoder value offset or a time delay, from an image acquired.

Symbology Tools

Symbology Tools are an optional set of tools.

- **HSBarcode Reader**: reads and extracts information from bar code symbology.
- **HSData Matrix Reader**: reads and extracts information from Data Matrix symbology.
- **HSOCR Fixed Font**: recognizes alphanumeric fixed-font character strings.

General Steps for Building an Application

1. Instantiate process.
 - If processes are not visible in the Available Processes list, double-click on folder to expand.
 - Select a process from the list and click **Add Process** to add it to the list of instantiated processes.
 - The properties page for the tool added tool appears.

Note:The first process of any application must be a HSAcquisitionDevice process.

2. Configure process.

Set up and configure the tool. Consult **Using** sections in each tool's chapter for details on the setup and configuration of individual tools. Return to the Process Manager after configuring the tool.
3. Execute sequence so that outputs are available for subsequent processes.
 - Click **Execute Sequence** in the Process Manager.
 - Use the Explorer panel to visualize each tools output to the runtime database.

Explorer

The Explorer panel contains two main sections: the Tree View and the Information Area. Basically, the Explorer lets you view the information contained in the application's runtime database.

Most HexSight tools rely on the output provided by a tool further up in the sequence of instantiated tools. The database can thus be used for diagnostic purposes. You can rapidly identify missing outputs that might be disabling the execution of some tools.

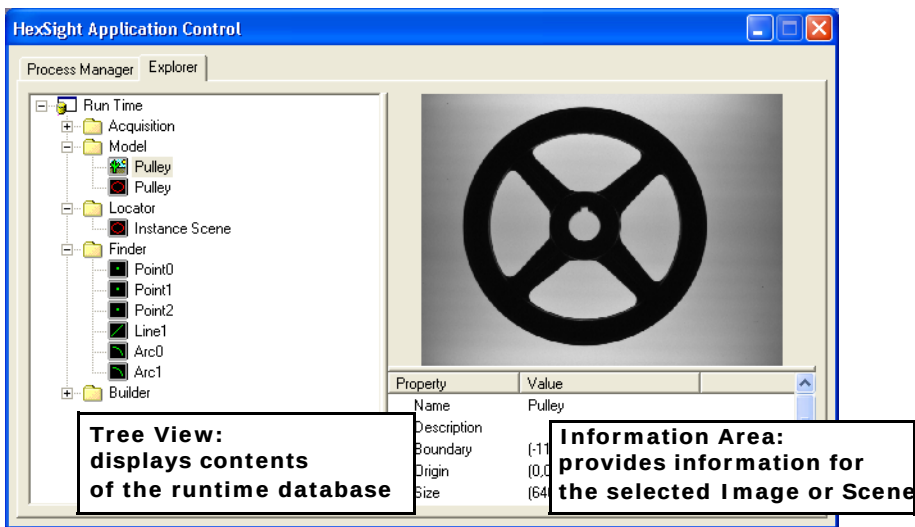


Figure 131 Application Control — Explorer Panel

Tree View

The tree view displays the contents of the application's runtime database as a hierarchical list of folders. These folders contain Views, Scenes and Images generated by the processes in the application.

- Click the +/- signs in the Explorer Database to expand/collapse the list of contained folders.

Note: The tree view appears empty until you have executed at least one process in the Process Manager's list of **Instantiated Processes**.

If you are in **design mode**, i.e. you accessed the application control from the ActiveX control on the form (by Visual Basic or Visual C++):

- Go to the Process Manager and click **Execute Sequence**.
- Return to the Explorer to view the database.

Views

A View is a folder or collection of database objects that are output by a HexSight process. These objects can be entities, images or scenes as shown in Figure 132.

- Typically, you will name one View per process. Model Scenes and Images are by default stored in a separate Model View.
- The name of the View is defined in the tool's property page.
- As you add processes and execute the sequence, each new View, with its associated objects, is displayed in the Explorer.

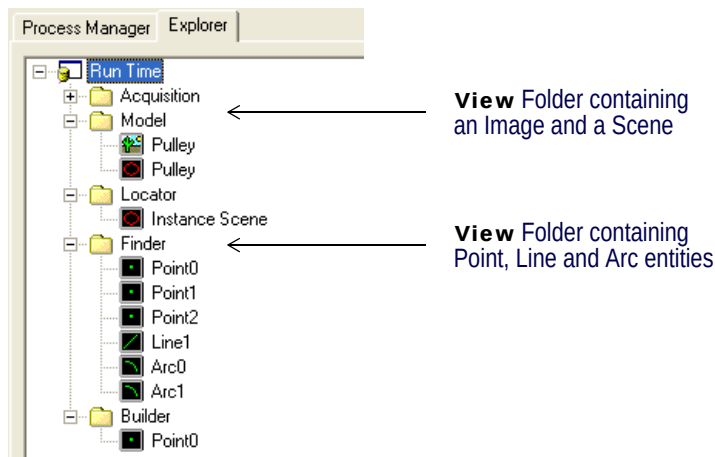


Figure 132 Expanded Views in the Explorer

Images

An image is a matrix of pixels that represents an object's 2D appearance. Images in the tree view, which have been generated by HexSight, can be saved as bitmap (bmp) images.

Scenes

A Scene is a vectorized representation of the geometry in an Image. More precisely, a Scene is a description of line vectors that reproduce the geometric shapes of an object.

Entities

An entity is a geometric object that is created and used by HexSight Finder tools. Basic entities are Points, Lines and Arcs.

Using the Tree View

Upon opening an application the Explorer is empty. A new database is created when you execute an application for the first time. [Figure 132](#) shows a database that has been created after the execution of an application.

Managing Runtime Database Objects

The context menu for the Tree View provides you with the following options for managing the runtime database, as illustrated in Figure 133.

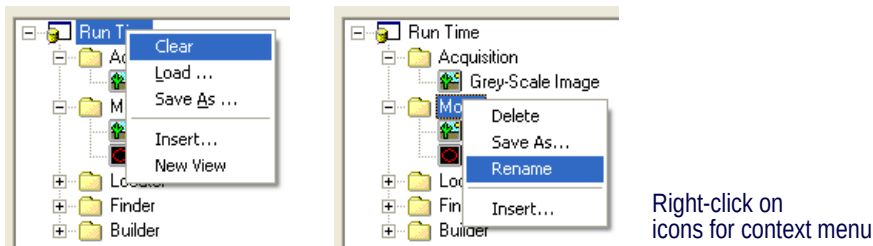


Figure 133 Database Options

- **Clear:** Clears all objects in the current database.
- **Load...:** Allows you to load an existing database file to the runtime database.
- **Delete:** Deletes the selected database object
- **Save As...:** Allows you to save the current database to file.
- **Rename:** Allows you to rename the selected database object.
- **Insert:** Allows you to add one or more objects from a selected file.

- **New View**: Lets you add a new View to the runtime database.

Opening and Displaying an Image, Scene or Entity

Scenes, Images and Entities can be displayed from the Explorer database.

- Use **+** symbol to expand View, Image and/or Scene folders.
- Right-click an icon to open a context menu.

Select **Show in New Window** or **Append To**.

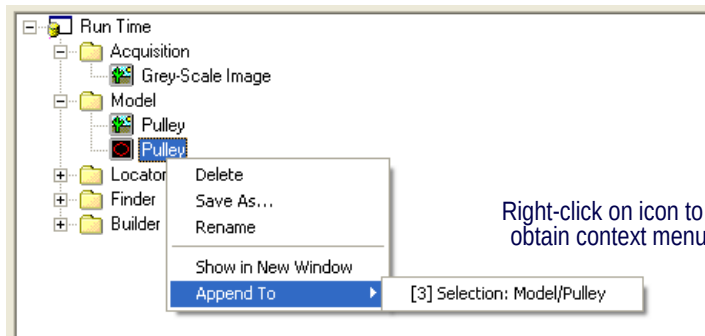


Figure 134 Displaying a Scene from the Explorer Database

- If you select **Append To** you must specify an already opened Image or Scene to be used as foreground. [Figure 135](#) shows a display window with the Contour Scene as foreground and the Image as background.

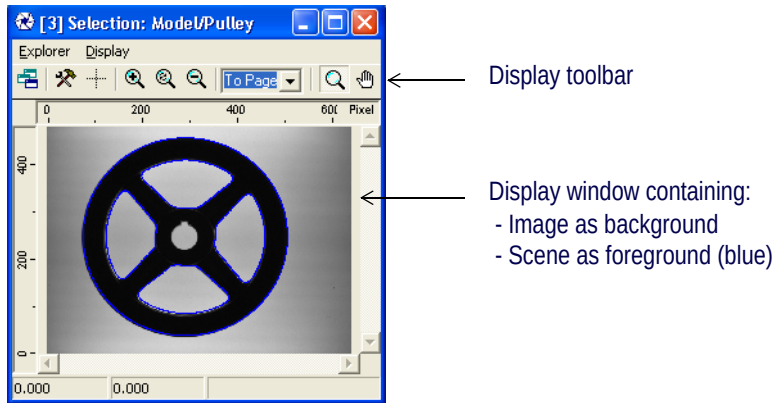


Figure 135 Display Scene and Image from the Explorer Database

Saving Database Objects

- On an icon in the tree view and select **Save As...** from the context menu and specify the path for the new file in the **Save** dialog box.
- Images are, by default, saved to the HexSight Images format (*.hig). You can also save images to the Windows Bitmap (*.bmp) format.
- Scenes are saved to the HexSight Scene format (*.hsc).
- Views are saved to the HexSight View format (*.hvw).
- Point Entities can be saved as HexSight Points format (*.hpt)
- Line Entities can be saved to the HexSight Lines format (*.hln)
- Arc Entities can be saved to the HexSight Arcs format (*.hac)

Information Area

The information area to the right of the tree view displays a thumbnail representation of the selected Image or Scene and information describing the selected item. See Figure 131.

Application Control Properties

General Application Control Properties

To access the General Application Control you must select the properties for the HexSight Application control, either in the Visual Basic or Visual C++ window.

Use this window to set the Application Control properties and to access the Process Manager and Explorer property pages.

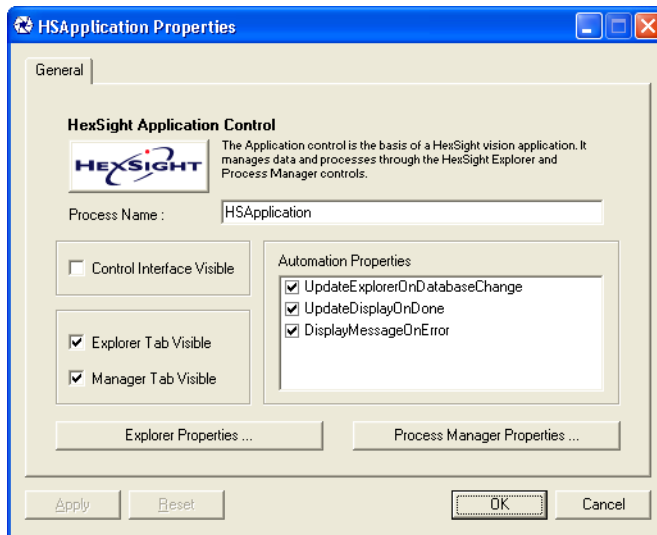


Figure 136 Application Control Properties

- **Process Name** lets you customize the name property for the Application Control.
- If **Control Interface Visible** is enabled, the Application Control interface will be visible in your application interface.

Automation Properties

- **Update Explorer On Database Change**: When enabled, each time the tree structure of the runtime database is changed, the Tree View is refreshed.

- **Update Display On Done:** When enabled, display windows opened with the Tree View are refreshed after each execution of the application.
- **Display Message On Error:** When enabled, message errors are displayed.

Process Manager Properties

Access Process Manager properties through the HSApplication Properties window by clicking the **Process Manager Properties**. Use this window to set the Process Manager properties.

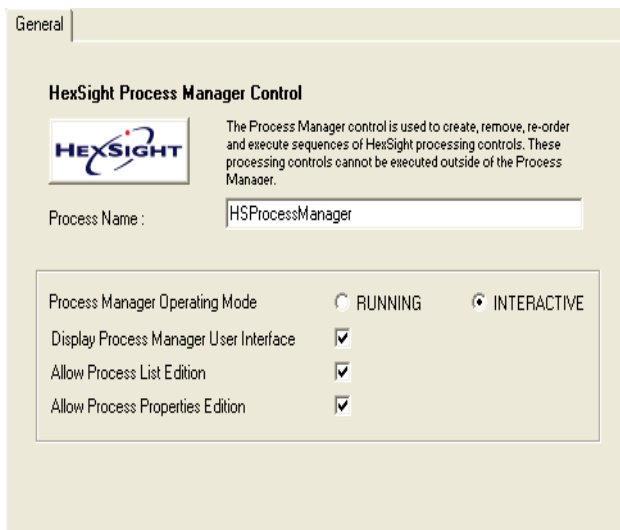


Figure 137 Process Manager Properties

Process Name

This lets you customize the Name property for the Process Manager.

Process Manager Operating Modes

Use the radio buttons to enable one of two modes:

Running: When this mode is enabled, no edition or creation can be carried out in the Process Manager.

Interactive: This mode lets you to control which edition functions can be carried out in the Process Manager:

- **Display Process Manager User:** If disabled, all graphical user elements are removed from the Process Manager window.
- **Allow Process List Edition:** If disabled, the list of Instantiated Processes in the Process Manager cannot be edited.
- **Allow Process Properties Edition:** If disabled, the properties pages of the tools instantiated in the Process Manager cannot be opened for edition.

Explorer Properties

Access this properties window from the Application Control properties window by clicking the **Explorer Properties** command button. Use this window to customize the appearance and content of the Explorer Panel in the Application Control Interface.

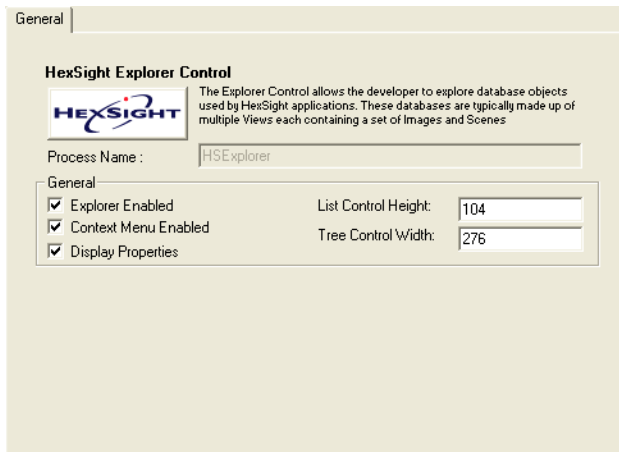


Figure 138 Explorer Control Properties

General Options

- **Explorer Enabled:** If disabled, the Explorer will not appear in the Application Control interface.

- **Context Menu Enabled:** If disabled, right-click option menus in the Tree View are not available.
- **Display Properties:** Enables the display of the information area.
- **List Control Height:** Sets the height of the information frame.
- **Tree Control Width:** Sets the width of the Tree View frame.

Locator

The Locator finds and locates objects based on the geometry of their contours. Scale factor, orientation, and position are provided for each located instance. Models are built using the integrated Model Editor.

Locator Chapter Outline

[Understanding Locator Concepts](#)

[Input and Output](#)

[Models](#)

[Model Creation and Edition](#)

[Model Optimization](#)

[Model-Based Tools](#)

[Search Process](#)

[Learn Process](#)

[Improving Search Performance](#)

[Locator Results](#)

[Troubleshooting](#)

[Using the Locator](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Contour Panel](#)

[Model Panel](#)

[Search Panel](#)

[Results Panel](#)

Understanding Locator Concepts

This section explains the main concepts pertaining to the Locator process. The order of the information is based on the Locator interface.

The Locator finds and locates instances of model-defined objects. Models characterize object types and are created and edited through the Locator's Model Editor.

Input and Output

The Locator owes its robustness to the use of vectorized descriptions. It searches for object instances on the geometric features calculated from the initial input image.

Input

The Locator requires the input of an image, typically provided by another HexSight process. To ensure the positional accuracy of the Locator process, the input Image should be calibrated beforehand, through the Acquisition Device.

It is possible to add one or more model-based Locator tools to an application. A model-based Locator requires the input of a **Model Scene** and an **Instance Scene**, which are provided by another “initial” Locator tool in the application. A model-based Locator can be used to precisely locate features or “sub-features/sub-parts” on a parent object. See the [Model-Based Tools](#) section for more details.

Output

The Locator outputs to the application's runtime database are used by subsequent model-based tools in an application. Outputs consist mainly of Scenes, which are vectorized representations of the geometry in an Image. The Locator can also output a Model View that contains both the image upon which the model was created as well as the Model Scene.

Model View

The **Model View** is an output of the Locator to the runtime database. It contains the image used to build the Model and the vectorized description of the Model, called the Model Scene.

Contour Scenes - Outline and Detail

A **Contour Scene** contains a vectorized description of contours generated from an input image. The Locator generates contours on two coarseness levels: **Outline** and **Detail**.

Instance Scene

The **Instance Scene** contains the contour representation of actual object instances that have been located within the current Image.

Since outputs to the runtime database increase the execution time, only outputs required for subsequent processes should be enabled. To display Locator results, you should preferably use the [Results Viewer](#).

Models

A Model is a representation of an object, based on the description of the geometry of its contours. The basic considerations for creating Models that will permit an accurate and efficient recognition of object instances can be resumed as follows:

- The part used to build the Model should be as "perfect" as possible, meaning it should define what the average shape of that part is. This improves the robustness to normal variations in a part's shape.
- The image used to build the Model, as well as the runtime images used to locate the part, should *always be calibrated*. This improves recognition robustness and location accuracy, especially when the parts to be located can be rotated or significantly translated. [Figure 139](#) illustrates the case of an image in which the pixel ratio is not exactly 1:1, as is typically the case. In the given example, if the image is not calibrated, when a part is rotated to 90 degrees, it will be impossible to locate the part to better than 1 pixel in accuracy.

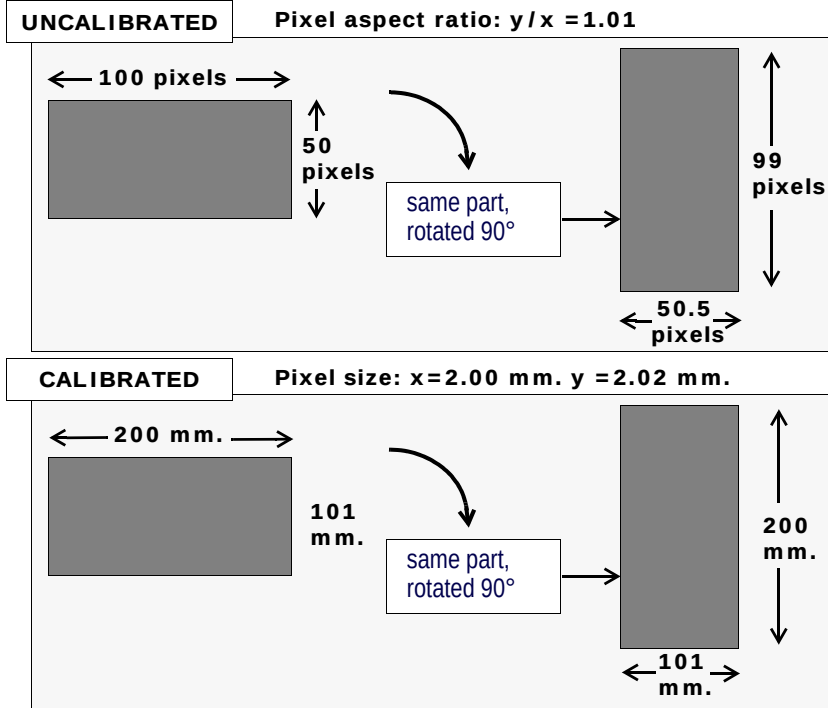


Figure 139 Calibration Required for Accuracy in Results

- The Locator will be more robust if the Model is built using a clean image, in which the geometry of the object is clearly visible. This is especially important when the runtime images are very cluttered and difficult.
- The features that constitute the Model should be stable in shape and relative positioning with respect to each other in the runtime images. Whenever possible, avoid using features created by shadows or specular reflections.

IMPORTANT: MODELS MUST BE CREATED FROM IMAGES CALIBRATED IN THE SAME TYPE OF COORDINATE SYSTEM AS THE SYSTEM IN WHICH THE INPUT IMAGE IS CALIBRATED. FOR EXAMPLE, IF THE MODEL IS CREATED FROM AN IMAGE CALIBRATED IN A RIGHT-HANDED SYSTEM, THE LOCATOR WILL NOT RECOGNIZE INSTANCES OF THE SAME OBJECT IN AN IMAGE THAT HAS BEEN

CALIBRATED IN A LEFT-HANDED COORDINATE SYSTEM. THIS WOULD REQUIRE A FLIP TRANSFORM, WHICH IS NOT SUPPORTED BY THE LOCATOR.

Models Database

The Models database contains the Models used by a given instance of the Locator tool. Such a database can contain any number of Models, each of which is constituted of both a Model Image and Model Scenes (Outline and Detail) that describe the contour geometry of an object.

- A Models database should contain Models that are similar in nature, size and calibration, such as models for parts or objects commonly used within the same application.
- A large number of Models can significantly affect the search speed.
- Cropping Models reduces the size of the models database files and the runtime memory usage can thus improve search time.
- Created Models, as well as modifications made to existing Models, will exist only in memory and will be lost once the application is closed, unless the changes are saved to the current database, or to a new models database.
- Models in a database can be individually enabled or disabled as needed for the current application.
- You can also import selected Models to the database from other HexSight Model databases.
- Model database files are created and saved in the same format as other HexSight database files (*.hdb).

Sharing the Models Database

Multiple instances of the Locator tool can share a same models database in memory. This is useful for applications in which the same models must be searched in different areas and/or with different parameters.

- Instead of having each Locator keep its own copy of the models database, handles can be used to share one Locator's models database with other Locators.
- The models database handle can also be used to directly display the models contained in a Locator's Model database in a Display control.

Disabling Model Disambiguation

The Locator automatically applies disambiguation to resolve both inter-model ambiguities (similar models) and intra-model ambiguities (similar hypotheses of same object). Disambiguation can be disabled when needed, typically in applications in which there is a very large Models database.

- Disabling disambiguation will significantly reduce the time to learn, or re-learn, the models.
- Disabling disambiguation removes the Locators capability to discriminate between very similar models and between similar hypotheses for a same object.

Model Creation and Edition

Models can be taught, built and edited through the Model Editor interface. Models can also be created programmatically through the HSModelEditor COM interface, without having to show the interactive user interface.

Note:The Model Demo, on the HexSight CD provides an example of the use of the HSModelEditor interface.

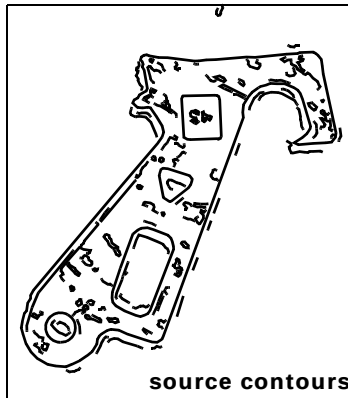
Teaching a Model

A Model is automatically taught by simply dragging the bounding box of the Model Editor interface to the region of interest in the image, then using the **Build** command. The Locator builds the Model, based on **Contour Detection** and **Feature Selection** parameters.

Model building is a three-step process. First, contours are detected in the image. Then, from the detected contours, appropriate features are selected,

as illustrated in [Figure 140](#). Thirdly, if necessary, the Model can be manually edited to add or remove features.

A: Source contours are detected in input image



B: Appropriate features are selected to build the Model

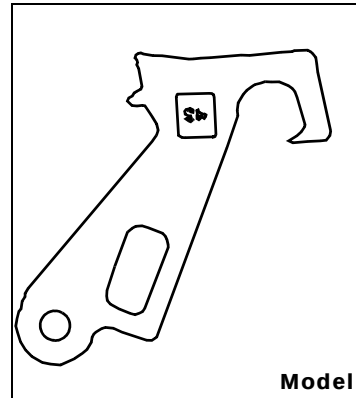


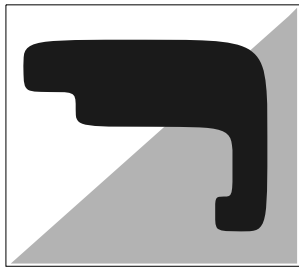
Figure 140 Model Building Process

Model Image Considerations

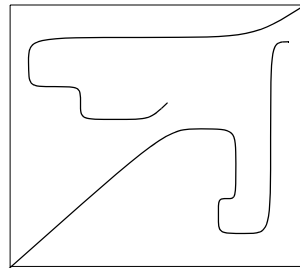
The choice and the quality of the image used to build the Model have an impact on the efficiency and robustness of the location process. The following points should be considered in selecting the Image on which to teach a Model.

- The image used to build the Model should be as ideal as possible.
- Make sure there are no shadows around the part in the image used to build the Model. If a shadow is near the outline of the part, it can distort its shape, which will reduce recognition robustness and positioning accuracy.
- The background should be as uniform as possible. If the background in the runtime images is textured, a uniform background such as a sheet of paper should be placed on the surface to eliminate the geometry of the background from the image that is used to build a Model.
- For example, in the first case of a given object, as shown in [Figure 141](#), if a Model is built from contours of the part on a textured background,

two separate features will be created to describe its geometry at the Outline Level. These features will be partly constituted of geometric information that is part of the background and will not be appropriate features for the Model since they are not stable from one runtime image to another.



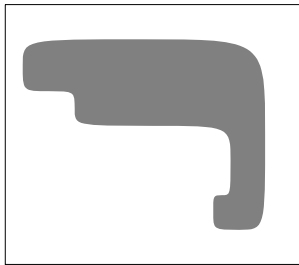
Image



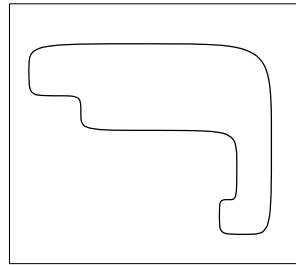
Contours

Figure 141 Broken Outlines Caused by Non-Uniform Background

- Although it is not mandatory, using long continuous features to describe the geometry of a part at the Outline Level will usually improve robustness.
- To improve the robustness of the Locator in the preceding example ([Figure 141](#)) it would be preferable to build the Model on a uniform background such as shown in [Figure 142](#).



Image



Contours

Figure 142 Complete Outline Obtained on a Uniform Background

- This unbroken representation of the part at the Outline Level will significantly improve the robustness of the Locator. In runtime images, the Locator will be able to extract a significant part of the contour geometry even though the contour is broken into smaller sections. See [Figure 143](#).

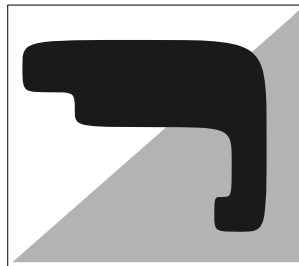
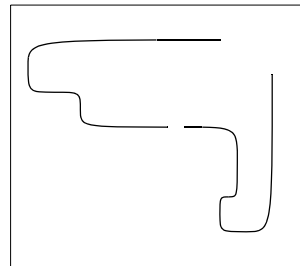
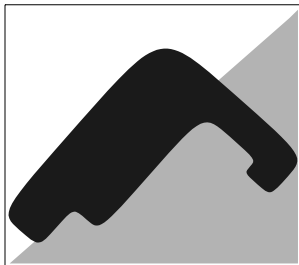
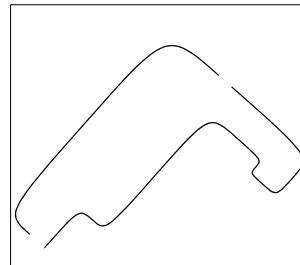
**Image****Matched Model****Image****Matched Model**

Figure 143 Examples of Matched Models on Non-Uniform Background in Runtime Images

Model Contour Detection

The Model Editor provides contour detection parameters that are applied by the Locator to generate the contours used to build a Model. This enables you to create Models having different contour detection settings than the default values. The Locator automatically sets optimized contour coarseness levels when **Automatic Levels** is enabled. To manually configure contour coarseness levels, you must disable the **Automatic Levels** mode.

Outline and Detail Level Values

- The Detail Level is used to refine the position and orientation of the object instance. In Automatic Levels mode, the Locator sets the value for the Detail Level based on the sharpness of the contours. This ensures accuracy and stability of the object position and orientation.
- To ensure positional accuracy: when you have sharp images, use the lowest Detail Level that will provide stable contours. If images are blurry, you may obtain better results by increasing the Detail Level. Blurriness in images is in fact the only reason for setting higher Detail Level values.
- To improve recognition robustness in multi-model applications, the same Outline and Detail coarseness should be used for all models.
- A higher coarseness setting at the Outline Level can improve robustness to high frequency noise and to fine textures.
- With a higher coarseness setting at the Outline Level, the shape of the Model is coarser. This can diminish the efficiency of object recognition and result in missed instances, especially in cluttered Scenes.

Contrast Threshold

- When it is difficult to obtain well-contrasted images for building the Model, reducing the **Contrast Threshold** value, (or selecting **High Adaptive Sensitivity**) can help detect weak-contrast contours. In this case, the amount of noisy contours in the background will also increase.
- If there is too much noise in the image a lower sensitivity setting (**Adaptive Low Sensitivity** or a higher **Contrast Threshold** value) can help reduce the noise.

Tracking Inertia

- Increasing the **Tracking Inertia** can help close small gaps and connect contours that would otherwise be broken into smaller sections.

Feature Selection

A feature is a section of an object's contour geometry. For any given model, the arrangement of features, especially at the Detail Level, should allow the Locator to reliably and accurately measure the scale, rotation and translation of the object.

- Features are added automatically, when the Model is taught, based on the setting of the **Feature Selection** parameter, which ranges from **none** (no features added) to **all** (all contours selected as features).
- The **normal** setting will select the most appropriate features from the detected contours. This setting should be used with simple to moderately complex patterns such as mechanical parts.
- The **all** position will incur the selection of every contour feature detected by the Locator, including small features and features with lower contrasts. This setting should be used only with very complex patterns such as those encountered in electronics and semiconductor applications.
- The **none** position should be selected when you want to manually select the features for the Model from the detected contours. Note that you must use the **Build Model** command to detect the contours.
- Once a Model has been taught, it can be edited to remove features or add manually selected features.

Manual Model Editing

The capacity to manually add features to an automatically generated Model is particularly useful when you know a specific object feature to be particularly important, unstable or required for unambiguous recognition.

- The principal consideration for selecting features to add to the Model is identifying portions of an object's contour that characterize it and set it apart from other somewhat similar objects or background while being stable with respect to lighting.

- More precisely, the user should favor features that are long and rich in their variation of curvature. This allows for more robust recognition and more precise positioning.
- Any number of features can be selected and added to a Model, at either the Outline or the Detail Levels or both.
- The Outline Level should contain at least a minimum amount of features that allow the Locator to reliably generate hypotheses.
- Non-distinctive and unstable features at the Outline Level can negatively impact the ability of the Locator to recognize the Model.
- The Detail Level should contain non-blurry and stable features that can be used to accurately locate the part. Blurry and unstable features at the Detail Level can negatively impact the location and positioning accuracy.

Stable Model Features

A good Model contains features that are stable and discriminating. A feature is said to be **stable** if it shows the following behavior from one image to another:

- The geometry of its contour remains unchanged under normal lighting variations.
- The geometry of its contour remains unchanged when the object is moved and/or rotated in the image.
- The relative position of the feature with respect to the other features of the Model remains constant (rigidity). For example, on a part with a strong 3D aspect, features that are closer to the camera will usually move with respect to more distant features. When working with such parts, it is best to build the Model using features that are at a similar distance to camera.

Unstable Model Features

- Features that are slightly unstable in position can be used in models by deactivating their **Location** property. They will be used to recognize the part but they will be ignored for refining its location.
- Using unstable features caused by shadows or reflections in the Model may reduce robustness and lower the **Fit** and **Match Quality** of recognized instances.

Discriminating Model Features

A Model should be built with discriminating features. A feature is ***discriminating*** if it allows the Locator to identify and locate the part among other objects appearing in the runtime images.

- A good feature is relatively long.
- A good feature is rich in curvature variations. For more robustness to occlusions, the curvature variations should be well distributed along the feature.
- A good feature contains a discriminative geometry that does not appear in other objects in the image.
- When possible, it is preferable to use features that individually define a complete Scale/Rotation/Translation transform. Examples of such features are shown in [Figure 144](#).

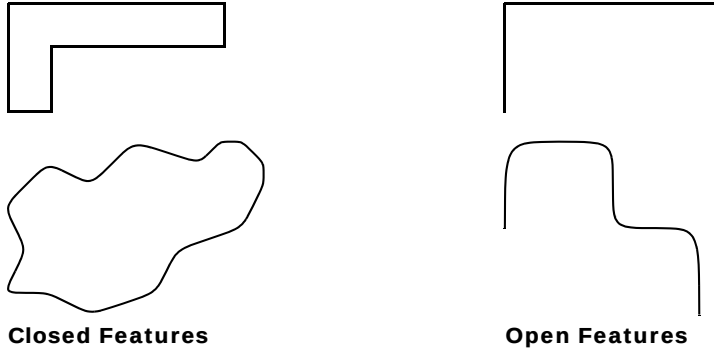


Figure 144 Features Rich in Curvature Variations

- Features that only define a partial Scale/Translation transform (complete circles, circular arcs, lines) can also be used but the arrangement of these features in the Model *must* define a *complete* Scale/Rotation/Translation transform as shown in [Figure 145](#).

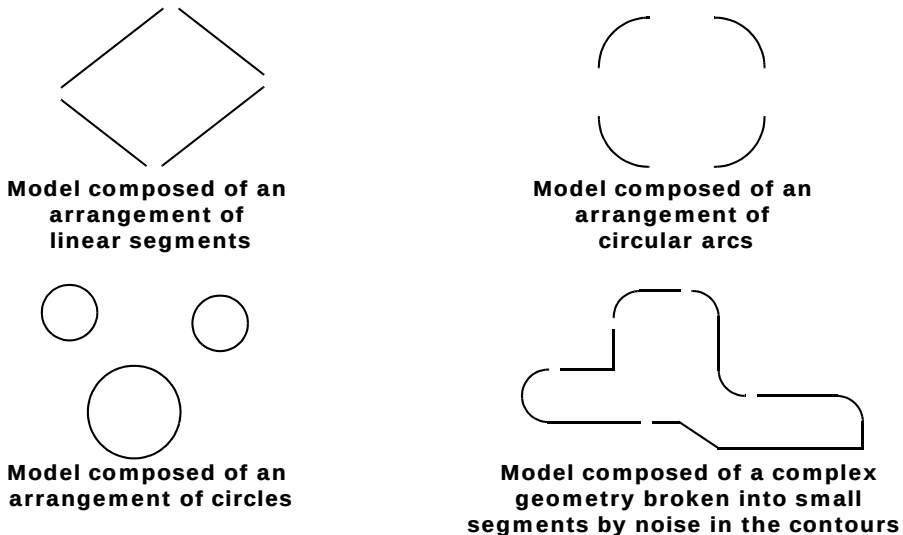


Figure 145 Model Defined with an Arrangement of Partial Transform Features

Symmetries

A good Model should not be symmetrical in rotation; otherwise the orientation of a recognized instance of the part will be ambiguous. As a general rule, include disambiguating features that break the symmetry of the Model and allow the Locator to establish the proper orientation of the object.

- A Model can be built using a single complete circle or an arrangement of concentric circles (see [Figure 146](#)) but the rotation will not be accurate nor repeatable.

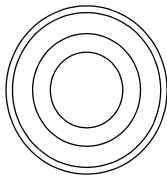
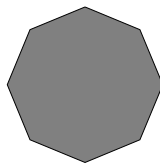
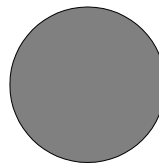


Figure 146 Model Defined with an Arrangement of Concentric Circles

- The term *complete circle* does not necessarily mean an actual smooth circle. It means a circular shape for which the Locator is not able to reliably compute a rotation. Sometimes a symmetric shape can be considered as a circle at high coarseness levels, as shown in [Figure 147](#).



Coarseness level of 1



Coarseness level of 8

Figure 147 Effect of Coarseness on a Symmetrical Shape

- Although each feature may have rotational symmetries, the arrangement of these features should not have symmetries, as shown in the *Bad Models* example in [Figure 148](#).
- Optionally, you can enable the **Required** property for one or more disambiguating features, which ensures that the Locator does not

output an instance of the object if these features are absent. **Required** features can be used at both the Outline and Detail Levels.

- If the disambiguating features are always present but are unstable in position, their **Location** property can be deactivated. When the Location property is deactivated for a given feature, the Locator only checks for its presence near its expected location but does not use the feature to compute the pose of the object. Non-location features can be used at both the Outline and Detail Levels.
- For example, to properly recognize and locate the object illustrated in [Figure 148](#), the feature properties could be set as shown in [Figure 149](#). The small circle will be required to recognize the part. The other features will not be individually required to recognize the part, as long as the **Minimum Model Percentage** and **Minimum Required Features** criteria are met. If the small circle is slightly unstable in position, its **Location** property can be set to false. Therefore, to compute the pose of the part, the Locator will use all of the recognized features except the small circle.

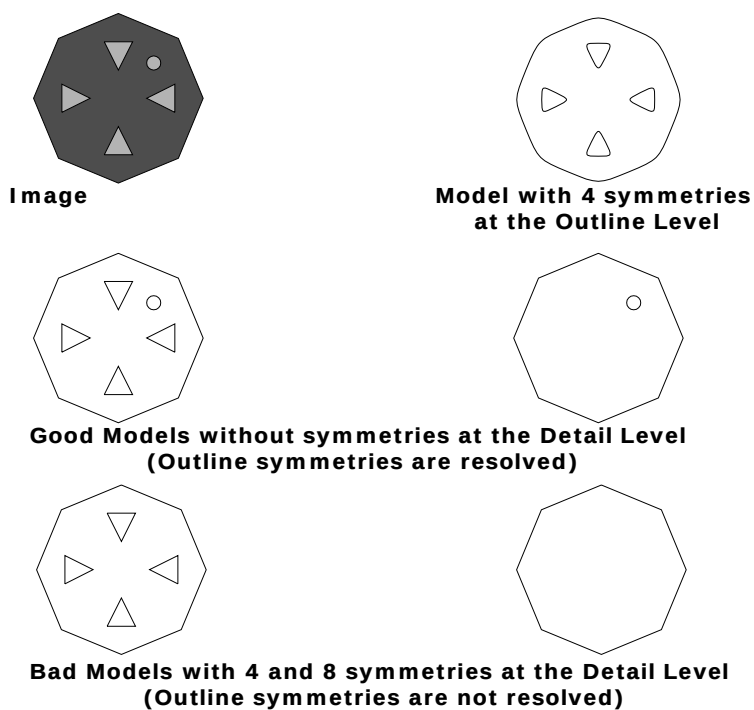


Figure 148 Good/Bad Model Examples built with Symmetrical Features

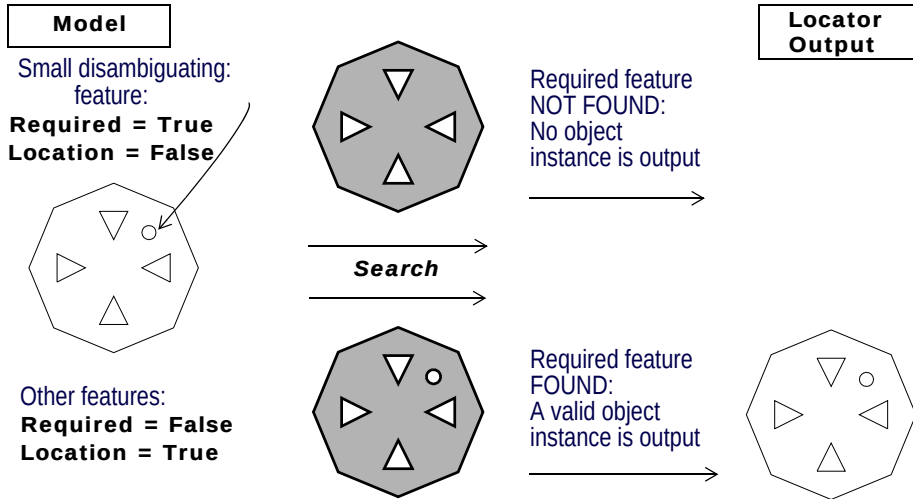


Figure 149 Effect of Assigning the Required Property to a Model Feature

Inter-Model Disambiguation

To ensure that the Locator will properly differentiate and recognize similar objects in multi-model applications each model must contain disambiguating features, which identify and distinguish the model from other similar models.

- The Locator automatically uses disambiguating features to distinguish between several similar Models. The example in [Figure 150](#) shows two “hook” objects that are identical except for the size of the circular hole at each end, which in this example is the *only* discriminating feature.
- If the disambiguating features are always present but are unstable in position, their **Location** property can be deactivated. When a feature deactivated for location, the Locator only checks for its presence near its expected location but does *not* use the feature to compute the pose of the object.

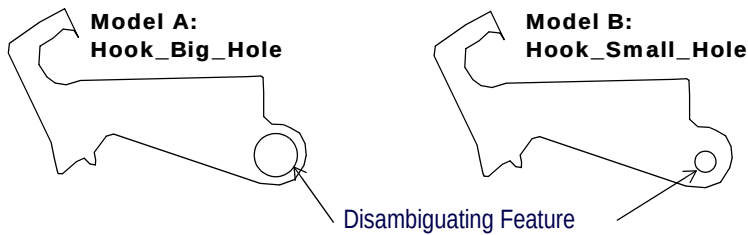


Figure 150 Using Discriminating Features to Distinguish Between Similar Objects

- Optionally you can, set one or more disambiguating features as **Required**. [Figure 151](#) illustrates a case in which the discriminating feature may sometimes be absent or undetected because of an occlusion. This example assumes that there is enough contour found for all three objects to be identified as the Hook model type. Since the Locator cannot find the **Required** feature on object C, this object instance not recognized. Note that **Required** features can be used at both the Outline and Detail Levels. In this example if the disambiguating features were not set as required, instance C would have been recognized as either Model A or Model B, depending on the **Match** and **Fit Quality** of the instance.

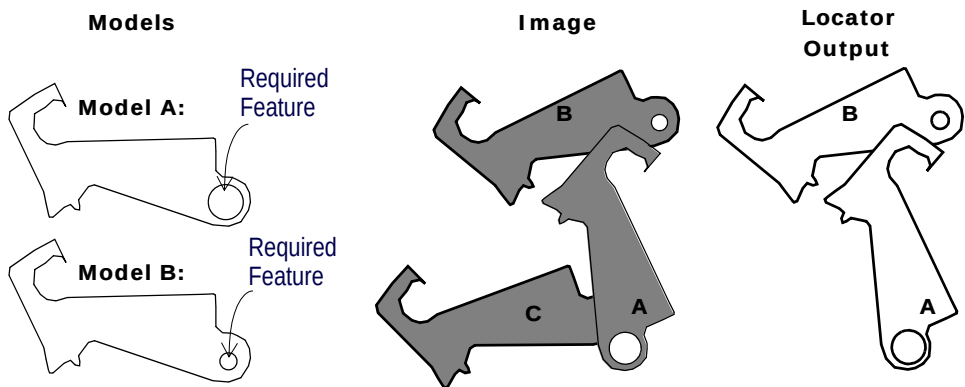
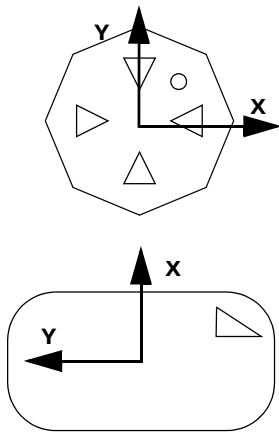


Figure 151 Using the Required Property in a Multi-Model Application

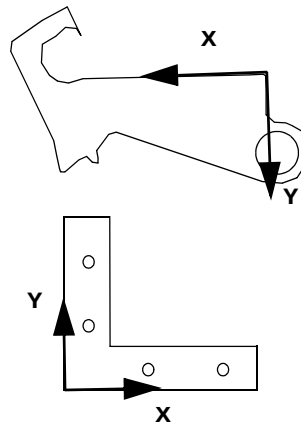
Object Coordinate System

The position and orientation of the Object coordinate system's origin is set by the user: the X-Y axes can be rotated and their origin displaced anywhere within the bounds of the Model Image, not necessarily on or within the object contour.

- At runtime, the position of each located instance will correspond to the location of the Object coordinate system's origin in the search image, while the orientation will correspond to the rotation of the Object coordinate system in the search image.
- The placement of the coordinate system on the Model has no impact on the recognition and location process. It only defines the origin of the part. It is typically placed at its center of mass or aligned with important features on an object, such as illustrated in [Figure 152](#).



Coordinate system positioned at the center of the object



Coordinate system aligned with important object features

Figure 152 User-Determined Placement of the Object Coordinate System According to Characteristics of the Object

Reference Points

A Reference Point identifies a point on an object that is of particular interest for a vision application. For instance, it can indicate a gripping point for a pick-

and-place application. You can place any number of Reference Points in a given Model.

- A Reference Point may or may not refer to a visible feature on an object. It constitutes a specific X-Y coordinate, referenced with respect to the Object coordinate system.
- The placement and number of Reference Points on a Model has no impact on the recognition and location process. They are only model-defined positions that are transposed with respect to located instances of the Model.

Model Editor Messages

The Model Editor provides a brief report on model building as well as troubleshooting information to help resolve model creation or edition problems. Messages are generated when the **Build Model** or **Analyze Model** command is applied. [Table 4](#) presents the possible diagnostic messages with their code numbers.

Table 4 Model Editor Messages

Message number	Message
1059	At least one feature must be checked for Location in the Outline Model
1060	At least one feature must be checked for Location in the Detail Model.
4000	Model Edition OK.
4002	Invalid Model. No contours in the Bounding Area at the selected Outline Level.
4003	Invalid Model. No contours in the Bounding Area at the selected Detail Level.
4004	No features have been selected in the Outline Level. Select features manually or increment the position of the Feature Selection slider.
4005	No features have been selected in the Detail Level. Select features manually or increment the position of the Feature Selection slider.

Table 4 Model Editor Messages

Message number	Message
4006	Contours seem blurry at the selected Detail Level.
4007	Detail Level was set to a level higher than 1. Location results may lack accuracy.
4008	Due to blurriness, the Detail Level was increased for recognition reliability.
4010	Scale and Translation may be unstable or lack accuracy.
4011	Rotation may be unstable or lack accuracy.
4012	Outline Model may contain symmetries. If the search is performed at the Outline Level Only and a stable rotation angle is required, break the symmetries by manually selecting features.
4013	Detail Level may contain symmetries. It might be possible to break the symmetries by manually selecting features.
4014	The Detail Level removes the symmetries existing at the Outline Level.
4020	Due to fuzziness, the Detail Level was increased for recognition reliability.
4021	Due to noise, the Detail Level was increased for recognition reliability.

Model Optimization

The purpose of the Model optimization is to build an improved model based on multiple object instances. Typical situations for optimizing a model include:

- Objects that vary in shape such that it is not possible to build a satisfactory typical model from a single object instance.
- Cases when a single model contains unwanted features resulting from noise or irregularities in the object and manual editing of the model to remove these extra features is not practical such as illustrated in [Figure 153](#).

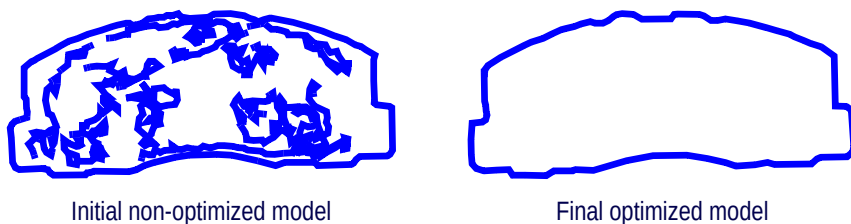


Figure 153 Model Optimization of a Noisy Object

Optimization Process

To build an optimized Model, you must provide an initial Model of the object then execute the Locator process. While optimization is enabled, each new instance of the object found by the Locator is analyzed and compiled into the optimized model that is currently in progress. Strong features, which are those that recur frequently in the analyzed instances, are retained in the optimized model. Once the model is considered satisfactory, the optimized model can be saved.

[Figure 154](#) illustrates an initial Model and the final optimized model for a variable-shaped object. Note that the initial image is retained in the Model; the image is not modified by the optimization process.

Optimization Parameters

Optimization parameters are set for the Detail and Outline levels of the Model. By default, the Outline level parameters are linked to the Detail level parameters in which case the **Link to Detail** property is enabled. This is the recommended setting. If necessary, **Link to Detail** can be disabled to allow you to control the setting of Outline level parameters.

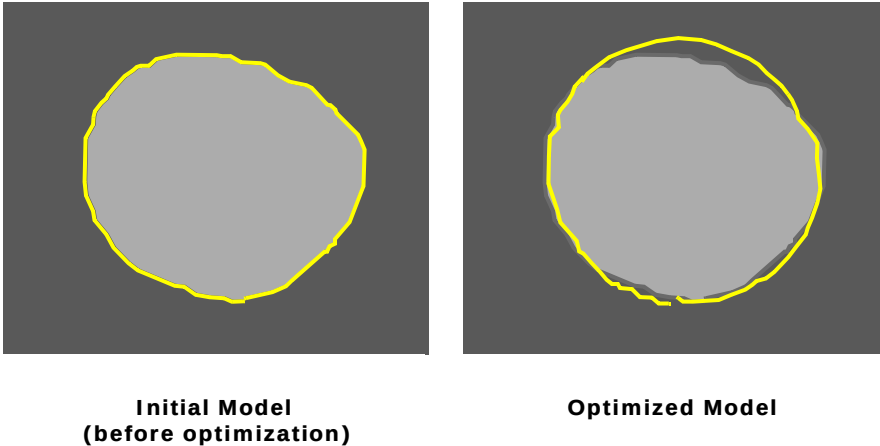


Figure 154 Model Optimization on an Object of Variable-Shape

Percentage of Points to Analyze

The **Percentage Of Points To Analyze** is set from within the Locator. This value sets the percentage of points on a model contour that are actually used for the optimization process. For example, when set to the default 50% value, one out of two points are used. Increasing this value can increase the accuracy of the optimized model but incurs a longer execution time.

Morph to Average Shape

Morphing allows the optimization process to add recurring features to the optimized model even if these features were absent from the initial model. These recurring features are appended to the model only if their frequency of recurrence is equal to or above the threshold set for **Minimum Feature Recurrence**. [Figure 155](#) illustrates the effect of enabling or disabling morphing in a case where strong features do not appear in the initial model.

When the **Link to Detail** property is enabled, enabling **Morph To Average Shape** at the Detail level automatically enables morphing at the Outline level.

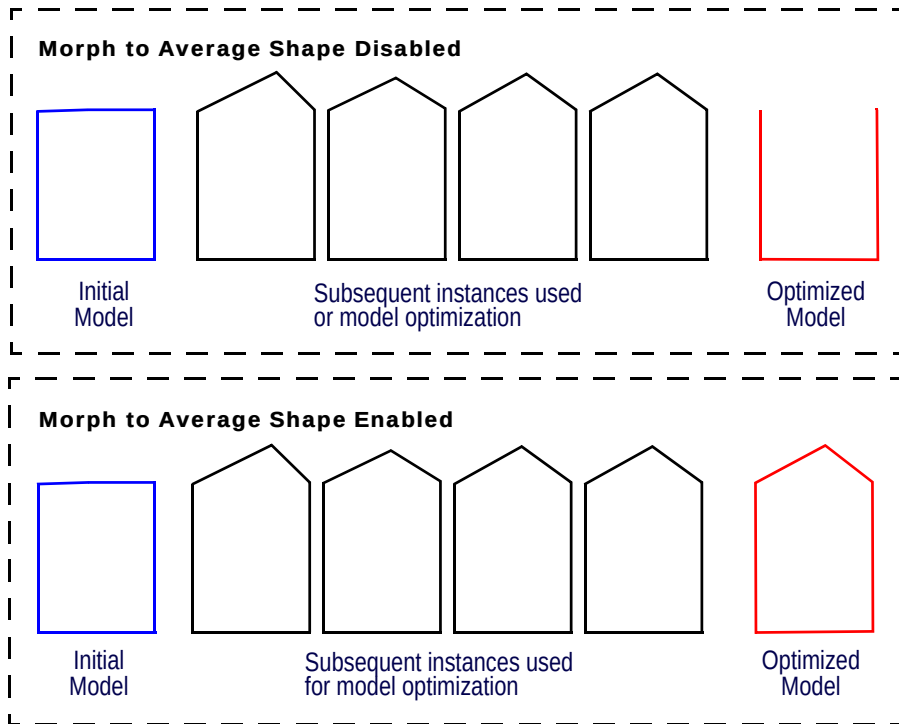


Figure 155 Effect of Morphing to Average Shape on Model Optimization

Minimum Feature Recurrence

Minimum Feature Recurrence sets the frequency of recurrence that is required for a feature to be retained and used for the optimization process. For example, at the default setting of 50, a feature will be retained for the optimized model only if it occurs in at least 50% of all analyzed instances.

When the **Link to Detail** property is enabled, the Outline level recurrence is automatically set to 10% lower than the Detail level feature recurrence.

Model-Based Tools

Most HexSight tools, including the Locator, can be model-based. Model-based tools enable easy inspection of fixtureless and randomly oriented parts. Moreover, implementing model-based tools can greatly reduce the amount of code needed to carry out an inspection task.

Model-Based Positioning Modes

There are two model-based positioning modes, Model-Based Automatic and Model-Based Manual. In both modes, a Locator tool provides a Model on which the tool is initially positioned. Subsequently when the application is run, the model-based tool is applied to modeled objects using transform parameters. In Model-based Automatic positioning, the transform is automatically provided by the Locator's Instance Scene. In Model-Based Manual, the transform must be provided manually.

The [Inspection Tools](#) chapter provides illustrated, in-depth information on HexSight [Tool Positioning Modes](#).

Model-Based Locator

A model-based Locator can be useful for precisely locating a feature or small part on a parent object. An example application on the HexSight CD: Board Location Model Based, shows how to gain advantage of the Locator's Model-based capabilities to simplify the coding of an already existing application.

A model-based Locator requires the input from another Locator tool in the application, called the "initial" Locator. The initial Locator must be configured to output a Model Scene to the database, and also an Instance Scene, if you are implementing the default Model-Based Automatic positioning mode.

Search Process

The Search process is the actual finding and locating of recognized object instances. The Locator first detects contours in the input image and then searches these contours for instances of objects defined within the Model database. The Locator generates hypotheses of potential instances, based on the Outline Level contours. The Locator then, confirms the hypotheses and

refines the location of the instances based on Detail Level contours. User-determined Search constraints and Recognition parameters are used to restrict the search, for example to find objects within a specific range of scale or orientation.

Rectangle

The **Rectangle** defines the region of interest that is processed by the Locator. By default, the **Rectangle** is the entire image but it can be reduced and placed anywhere within the area of the input image (see [Figure 156](#)).

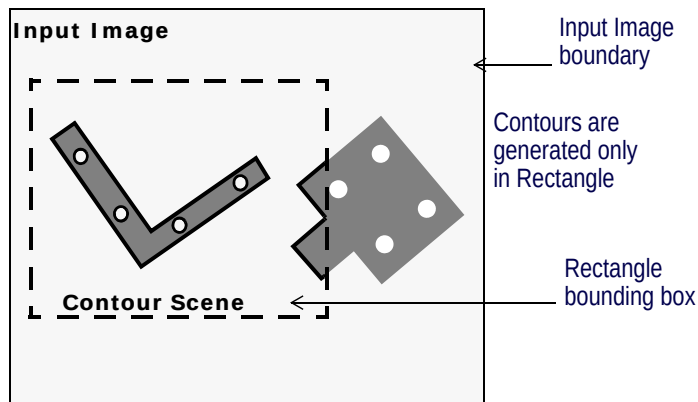


Figure 156 Locator Rectangle is the Locator Region of Interest

- A smaller area constrains the search process to a specific region of interest and can reduce the execution time.
- Since the Locator can be configured to tolerate occlusions, it may be possible, depending on the application's requirements, to locate objects that fall partially out of the bounds of the **Rectangle**.

Note: To improve recognition robustness, the Search should be configured to use same Outline and Detail coarseness levels as those used to create the Models that are used by the application. This is done by setting the **Parameters Based** on property to **All Models**.

Learn Process

A **Learn** process is carried out the first time the Locator tool is executed. A Learn is also carried out in the execution that follows a change to search parameters. Exceptions to this are presented further in this section.

In the Learn process the Locator "learns" the currently activated models with respect to the current Search parameters. The Locator integrates the information gathered during the Learn into the search process. Subsequent executions of the Locator are therefore significantly shorter than an initial execution that includes a Learn. The Locator outputs separate execution times for both the Learn and Search processes.

Note: Learn should not be confused with Teach, which is the term applied to creating a new Model.

Exceptions to the Learn Process

Because a Learn can add a significant amount of time to an execution, exceptions to the Learn have been built in for applications that modify certain parameters during runtime. This is especially helpful for multi-model applications that use a large number of models.

Exceptionally, the following do not trigger a re-learn:

- Changes to the list of enabled models,
- Changes to Scale or Rotation constraints,
- Changes to Conformity Tolerance value or Mode.

Basically, in these cases, the Locator does not need to re-learn the models if new parameters are contained within the range of the initially learned parameters.

Changes to the List of Enabled Models

The Locator initially learns only the models that are currently enabled (activated) in the Models database.

- Changes to the list of enabled models will *not* incur a Learn when you enable or disable models that were initially learned, i.e. that were enabled when the Learn was carried out.
- A Learn *will* be executed however, if new models, that have not been previously learned, are added or if any of the models are edited.

Changes to the Scale Factor Range

After an initial learn phase, you can reduce the range of scale factors *without* incurring a new learn as long as the scale factor values remain within the maximum and minimum values of the initially learned range.

- If you select a range of scale factors outside the learned range, a new learn will occur at the next execution of the Locator and the previously learned scale factors will be discarded.

Changes to the Range of Rotation

After an initial learn, you can reduce the range of rotation *without* incurring a new learn as long as this range is contained within the previously learned range.

- If you select a range of rotation outside the learned range, a new learn will occur at the next execution of the Locator and the previously information learned will be discarded.
- In the example in [Figure 157](#), range “**B**” will be set without a incurring learn. Because range “**C**” lies partially outside the learned range “**A**”, the Locator will automatically learn the new range “**C**” and discard the range “**A**” settings.

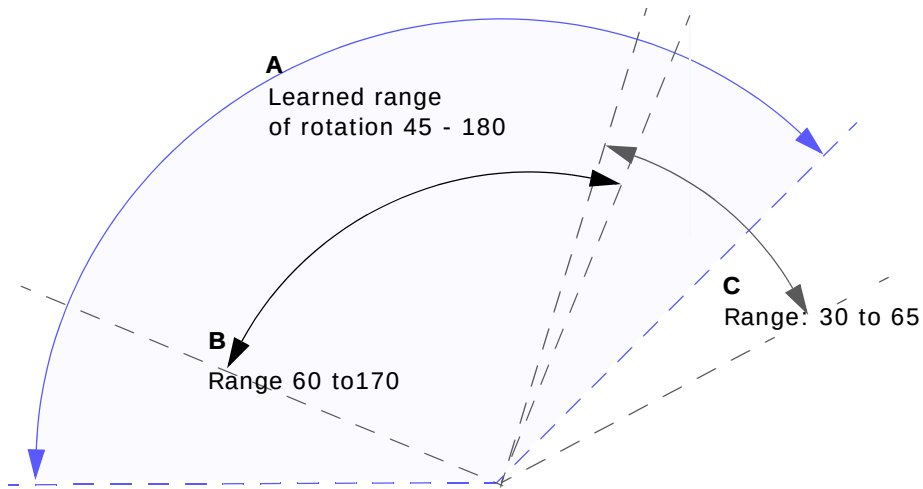


Figure 157 Learned Range of Rotation

Learn and Unlearn Commands

The **Learn** command initiates the Learn process only, without executing the Locator. This Learn includes any modifications made to parameters since the since the last execution. The purpose of is to ensure that the next execution of the Locator will be shorter, because the Learn has already been carried out.

The **Unlearn** command clears all information gathered by the last Learn. The next execution of the Locator that follows the **Unlearn** command, the Locator will re-learn all the active models. This command is useful for freeing up memory.

Contour Detection

The Locator carries out contour detection by processing the input images to generate a vectorized description of the image, called a Contour Scene. The Locator generates source contours on two coarseness levels: Outline and Detail, within a region of interest called the **Rectangle**.

Contour Detection Parameters

Contour detection parameters modify the quality and quantity of contours that are generated by the Locator from the input image. These parameters can be set either automatically, from the application's list of Models, or manually.

- Although each Model in the database has its own set of contour detection parameters, to preserve performance at runtime, contour detection is performed using only one set of parameters.
- In multi-model applications you can specify which Model's detection parameters the Locator should apply to contour detection.
- You can also specify that contour detection parameters be provided by the whole set of Models. In this case, the Locator selects the most restrictive contour detection parameters among all the models in the current models database: the lowest Outline Level, the highest Detail Level, the highest sensitivity Contrast Threshold and the lowest Tracking Inertia value.
- For the above reasons, to obtain higher performance, robustness and accuracy, the Locator should preferably search for objects that are similar in size and detail. A simultaneous search for very different objects, such as a very small object with a large object, can noticeably hinder the Locator's performance.

Outline and Detail Levels

The Locator process generates two levels of contours. The coarser Outline Level is used to generate hypotheses of potential instances. The finer Detail Level is used to confirm the hypotheses and refine the location of the instances.

- For each level, coarseness values range from 1 to 16. At level 8 for example, the resolution is 8 times lower than an image at full resolution (level 1).

- The Detail Level, by definition has a coarseness value less than or equal to that of the Outline Level. Similarly, the Outline Level value is always greater than or equal to that of the Detail Level.
- To improve search speed, set the Outline Level to the highest value at which contour geometry is still distinctive and stable.
- To obtain high-accuracy object location, it is preferable to use images with sharp edges and set the lowest coarseness value for the Detail Level.
- For images that are not in perfect focus, better results will be obtained with a higher value for the Detail Level.

Contrast Threshold

The **Contrast Threshold** sets the level of sensitivity that is applied to the detection of contours in the input image. The contrast threshold can be either Adaptive, Fixed or based on Models.

Adaptive thresholds are so named because the Locator internally sets a sensitivity level based on image content. This provides flexibility to variations in image lighting conditions and contrast during the Search process. There are three possible **Adaptive** thresholds: **Normal**, **Low** and **High**.

- **Adaptive High Sensitivity** provides a Contour Scene that includes a great amount of low-contrast contours and noise.
- **Adaptive Low Sensitivity** setting provides a Contour Scene that retains strongly defined contours and eliminates noise, at the risk of losing significant contour segments.

A **Fixed Value** threshold sets an absolute value for the sensitivity to contrast. A typical situation for the use of a fixed value is a setting in which there is little variance in lighting conditions.

- The numerical fixed value corresponds to the minimum step in intensity values required to detect contours.

- A lower value adds more contours at the expense of adding more noise, which may possibly generate false detections and/or slow down the search process.

Model-based thresholds allow you to base the contrast threshold on active Models.

- **Model Mode** applies the most sensitive mode, selected from the Models specified by the **Parameters Based On** selection.
- **Model Value** applies the most sensitive value, selected from the Models specified by the **Parameters Based On** selection.
- **Model Value and Mode** (default) applies either the most sensitive mode or the most sensitive value, selected from the Models specified by the **Parameters Based On** selection.

Search Constraints

Search constraints provide restrictions for the Locator's Search process.

Scale Factor

The scale of objects to be located can be set at a fixed nominal value, or as a range of scale values.

- The default setting for the scale factor is a nominal value of 1, which applies to most situations.
- If a nominal value is used with objects that present a slight variation in scale, the objects may possibly be recognized and positioned with reduced quality because their true scale factor will not be measured. In such a case it is preferable to configure a narrow scale range, such as +/- 2%, instead of a nominal value.
- Using a wide range in scale can significantly slow down the search process. This range should be configured to include only the scale factors that are actually expected for a given application. The scale factor range is one of the parameters that has the biggest impact on search speed.

Rotation

By default the Locator finds objects within a full range of rotation values: between -180 and 180 degrees. Allowable rotation can be restricted by either decreasing the range of rotation values or by setting a fixed nominal value. The rotation range spans counterclockwise from the specified minimum angle to the maximum specified angle. [Figure 158](#) illustrates impact of selecting a minimum and maximum angle.

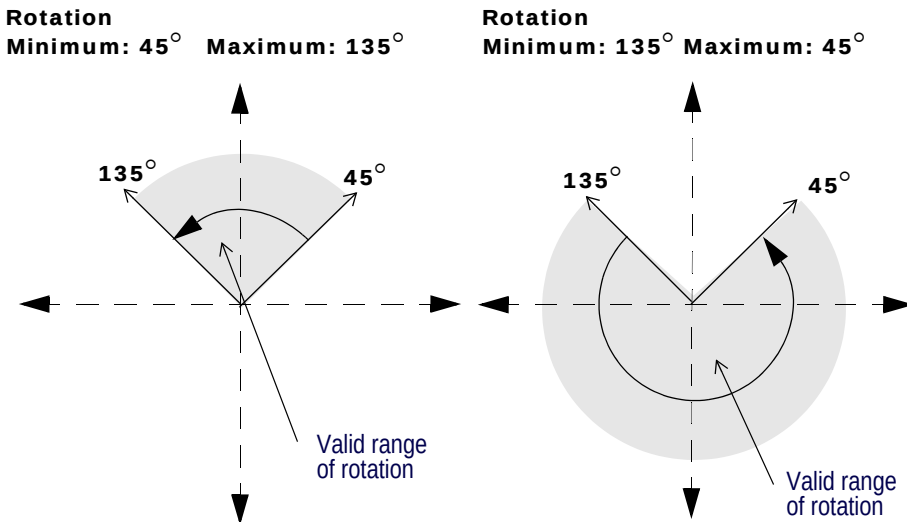


Figure 158 Rotation Range — Minimum to Maximum Angle

- If a nominal value is used with objects that present a slight variation in rotation, the objects may possibly be recognized and positioned with reduced quality because their true rotation will not be measured. In such a case it is preferable to configure a narrow rotation range of +/- 1 degree instead of a nominal value.

Number of Instances

The number of instances that can be recognized in an image is theoretically limited only by physical factors such as the number of objects that can be

contained within an acquired image. However, the Locator can be constrained to identify only a specific number of instances.

- **Instances to Find** limits the total number of objects to be found in the input image, *NOT* the total number of objects of each type.
- If the number of objects present in the input image exceeds the constraint, the Locator retains only the best quality object instances, regardless of their type.
- Quality of instance is a combination of **Fit Quality** and **Match Quality** (see **Results, Quality**).

Timeout

The timeout value sets a time limit indicator after which the Locator terminates the processing of an image. If the Locator has not finished processing an image within the allotted time, the Locator returns all the instances it has located up to the timeout.

- This is particularly useful for time-critical applications in which fast operation is more important than the occasional occurrence of undetected object instances.
- This value is only indicative; the actual processing time may be slightly greater.

Instance Ordering

The **Instance Ordering** parameter sets the order in which object instances are output.

- At the default **Evidence** setting, the instances are ordered according to their hypothesis strength.
- Instances can also be output in the order they appear in the image: **Left to Right, Right to Left, Top to Bottom, and Bottom to Top**. This feature is particularly interesting for pick-and-place applications in which parts that are farther down a conveyor must be picked first. It is also possible to use their **Precise** counterparts for an exact directional ordering.

- The **Quality** setting orders instances according to their Match Quality. Instances having the same Match Quality are subsequently ordered by their Fit Quality. This setting can significantly increase the search time because the Locator cannot output instance results until it has found and compared all instances to determine their order. The time required to output the first instance corresponds to the total time needed to search the image and analyze all the potential instances. The time for additional instances is zero since the search process is already complete.
- **Image Distance** orders instances according to their proximity to a user-defined point in the Image coordinate system. The X,Y coordinates of the point are expressed in pixels.
- **World Distance** orders instances according to their proximity to a user-defined point in the World coordinate system. The X,Y coordinates of the point are expressed in the user-selected length units.
- **Shading Consistency** orders instances according to their shading consistency (or color, depending on the input image) with respect to the Model, beginning with the strongest. This mode is useful for BGA applications, where the shading or color information, in addition to the normal contour information, can assist in discriminating between very similar hypotheses. See the example in [Figure 159](#).

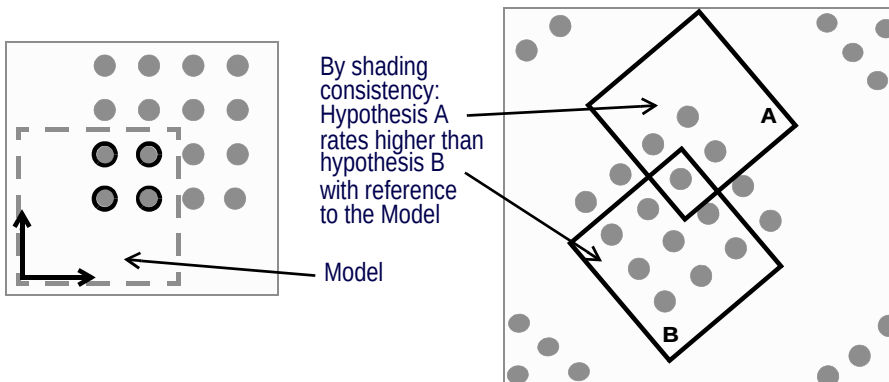


Figure 159 Instance Ordering - Shading Consistency

Contrast Polarity

Contrast Polarity defines the polarity change between an object and its surrounding area. This polarity can be dark to light or dark to light. The reference polarity for an object is defined by its Model.

- If **Normal** is selected, the Locator searches only for object instances having the same object-to-background polarity as the Model. For example, if the model is a dark object on a light background, the Locator searches only for dark objects on a light background.

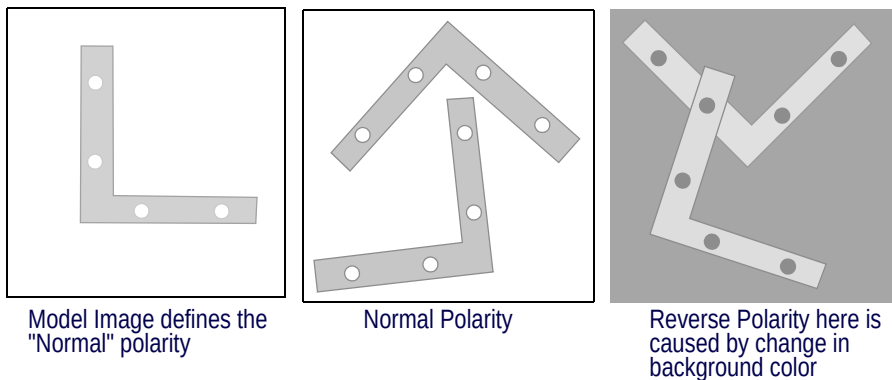


Figure 160 Contrast Polarity - Normal and Reverse

- **Reverse** restricts the Locator to search only for objects having a object-to-background polarity that is inverse to that in the Model. For example, if the model is a dark object on a light background, the Locator searches only for light objects on a dark background (see [Figure 160](#)Figure 160).
- **Normal & Reverse** enables the Locator to search for all cases that present either Normal or Reverse polarity. This *will not* take into account cases where polarity is reversed at various locations along the edges of an Object.
- When **Don't Care** is selected, the Locator will be robust to complete contrast reversal as well as to local contrast reversal, on sections of the object contour, such as shown in the “checkerboard background” example in [Figure 161](#).

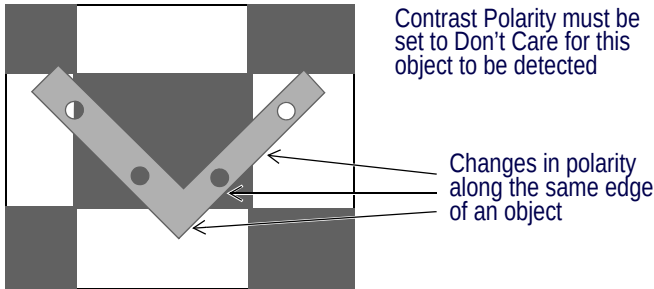


Figure 161 Contrast Polarity - Local Reversals in Polarity

Symmetric Instances

This option constrains the Locator to output not only the most plausible position and orientation for an instance but also *all possible* symmetric poses in the observed Scene. Symmetrical objects can present a problem for location. Since their orientation cannot be deduced from a given Scene, the **Locate Symmetric Instances** setting enables the Locator to output all possible locations of a single object.

- If this option is not enabled for symmetrical objects, then only the estimated best-case instance of the object will appear in the results.

Search Based on Outline Level Only

This constraint restricts the Search to using only the Outline Level source contours and the Outline Model, to search, recognize and position object instances. Detail Level contours are ignored completely for recognition and pose refinement.

- The trade-off for enabling this parameter is reduction in positional accuracy and increased possibility of detecting false instances.
- An Outline-based search is useful for time-critical applications that do not require a high positioning accuracy, or that need only to check for presence/absence of objects.
- To be effective, this type of search requires clean runtime images that provide high contrast contours with little or no noise or clutter.

Recognition Parameters

The Locator searches the contours detected in the input image for Model-defined contours. For a given application context, the user can adjust the Locator's recognition process using the following parameters.

Recognition Effort

To improve flexibility, the Locator provides a parameter for adjusting the recognition effort, which ranges from **Quick** to **Exhaustive**.

- The default center position has been optimized to suit most cases.
- A quicker setting will improve speed for easily recognizable objects on clean backgrounds
- Contours in the image that fall within the conformity tolerance of their associated model contours are considered matched and increase the **Match Quality** of the instance.
- Hard to recognize models, consisting of arrangements of small features or objects on cluttered or noisy backgrounds, such as encountered in electronics and semiconductor applications, sometimes require higher settings for the recognition effort.

Conformity Tolerance

Conformity Tolerance defines the maximum local deviation between the expected model contours and the contours actually detected in the input image, as illustrated in [Figure 162](#). Based on the imaging resolution during the Learn process, the Locator estimates and sets a default contour conformity tolerance as well as a valid range for conformity tolerance.

- The **Conformity Tolerance** value corresponds to the maximum distance in calibrated units by which a matched contour can deviate from either side of its expected position.
- By default, the **Conformity Tolerance** value is limited by the read only **Valid Conformity Tolerance Range**, in which case any value outside the range is automatically adjusted by the Locator to equal either the minimum or maximum value of the range.

- **Valid Conformity Tolerance Range** can be disabled to allow setting of **Conformity Tolerance** to any required value.
- Increasing the conformity tolerance can help increase the **Match Quality** of objects having small imperfections on their contour.

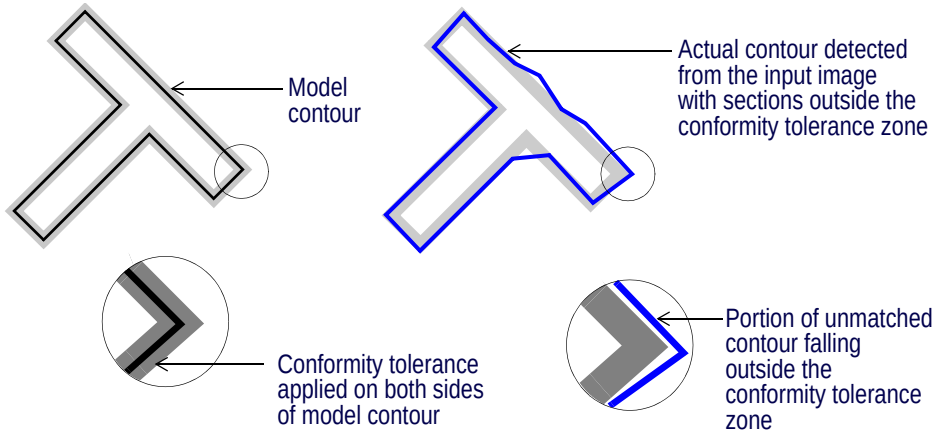


Figure 162 Conformity Tolerance

Minimum Model Percentage

The **Minimum Model Percentage** is the minimum percentage of model contours that need to be matched in the input image in order to consider the instance as valid.

- Only those contours that can be found within the conformity tolerance zone are considered. See the magnified representation in [Figure 164](#).
- Decreasing **Minimum Model Percentage** can help detect objects which are either partially occluded ([Figure 163](#)) or objects having significant defects in their contours ([Figure 164](#)).

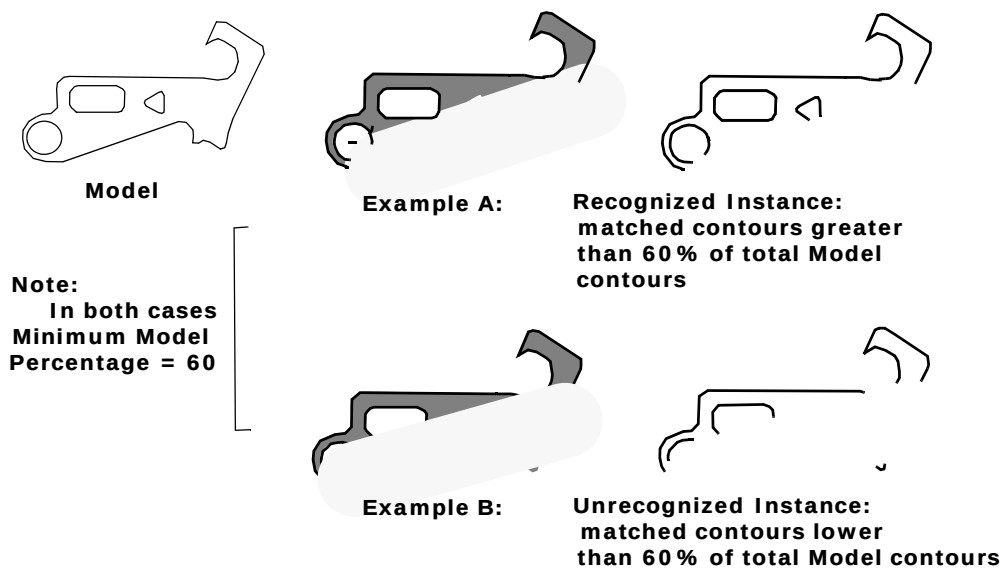


Figure 163 Minimum Model Percentage — Example with Occlusion of Object Contours

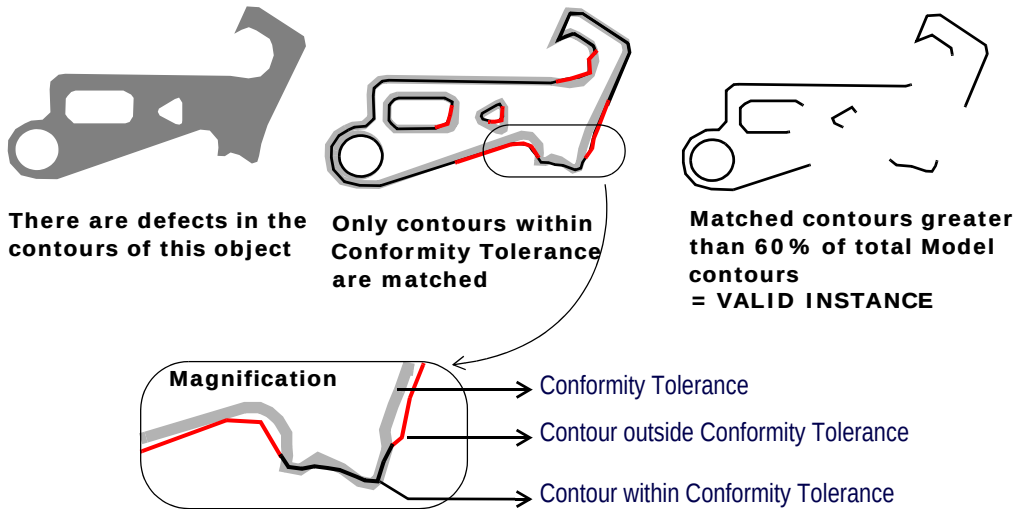


Figure 164 Minimum Model Percentage — Example with Defects in Object Contours

Minimum Clear Percentage

Minimum Clear Percentage sets the minimum percentage of the model bounding box area that must be free of obstacles to consider an object instance as valid. Enabling Minimum Clear percentage may significantly increase the search time; it is intended for use in pick-and-place applications. Enabling **Minimum Clear Percentage** also activates the computation of the **Clear Quality** result for each instance.

Minimum Percentage of Required Features

Required features, as set through the Model Editor, are features that need to be recognized for the Locator to accept a valid instance of an object. Although typically, the presence of all such features (100%) is required, the **Minimum Percentage of Required Features** parameter lets you set a lower percentage of required features.

- Note that this percentage is expressed in terms of the *number* of required features in a model without considering the amount of contour

each required feature represents in the model. For example, if the model contains 3 required features and **Minimum Percentage of Required Features** is set to 50%, an instance of the object will be considered valid as long as 2 out of 3 **Required** features are recognized.

Positioning

At the end of the search process, the Locator uses the Detail Level to refine and finalize the position of recognized object instances. The **Positioning** parameter allows the flexibility to configure the speed/ accuracy trade-off of the positioning process, through a scale that ranges from **Quick** to **Accurate**.

- The Locator has been optimized to provide the best speed/ accuracy trade-off at the default mid-range setting.
- It is possible to sacrifice location accuracy for speed improvement by setting the slider towards **Quick**, for presence/absence detection applications for example.
- Conversely, a slight gain in accuracy can be obtained by setting the slider towards the **Accurate** setting.

Improving Search Performance

A key factor in improving the performance of the Locator is to constrain the search as much as possible.

Effect of the Learn Process on Performance

- After models and/or search parameters have been changed, the first execution of the Locator is always longer because the models to be recognized have to be learned according to the search parameters.
- Exceptions to this behavior exist for changes made to the Scale range, the Rotation range or to the list of enabled Models. See the [Exceptions to the Learn Process](#) section in this chapter.

Model Considerations Affecting Performance

- A large number of models can significantly affect the search speed, especially when the Recognition effort is set towards Exhaustive.
- When you build a model using automatic feature selection in the Model Editor, always start by setting the **Feature Selection** slider to **Normal**. This will usually provide the best robustness/accuracy compromise.
- With complex patterns such as those encountered in electronics and semiconductor applications, it is sometimes preferable to set the **Feature Selection** slider to **more** or **all** to improve robustness.
- The bounding area, in which model contours are detected, has no effect on the search process. It only restricts contour detection to the area of the image in which Model features will be selected. However restricting the bounding area together with the **Crop** function can reduce the size of models databases for the application and thus reduce memory usage at runtime.

Search Constraints Affecting Performance

Scale

- Scale factor range has the biggest impact on the performance. Restrict it to a nominal value or a short range when possible. Bear in mind that when the scale factor is set to a nominal value, the Locator *does not* compute the *actual value* of the parameter. The nominal value is used as *is* by the Locator to position the instance.
- In applications requiring multi-scale capabilities but where the scale factors are few, fixed and known beforehand, it is usually best to define specific models for each of the scale factors that may be encountered and constrain the search to a nominal scale factor of 1.
- In multi-scale applications, setting the Recognition slider towards **Quick** can greatly improve search speed.
- In applications where you are using a Model-based Locator to find sub-features on a parent object that varies in scale, you can set a **Model-**

Based Scale to provide a faster execution time. In such a case, select the **Relative** mode and provide a scale range that will inform the tool on the possible scale values of the selected instance. See the [Model Based Scale and Rotation](#) in the [Using the Locator](#) section for more information.

Rotation

- A smaller rotation range improves the performance of the Locator: Use the minimum range required for the application. Bear in mind that when rotation is set to a nominal value, the Locator *does not* compute the *actual value* of the parameter. The nominal value is used as is by the Locator to position the instance.
- In applications where you are using a Model-based Locator to find sub-features on a parent object that varies in rotation, you can set a **Model-Based Rotation** to provide a faster execution time. In such a case, select the **Relative** mode and provide a rotation range that will inform the tool on the possible rotation values of the selected instance. See the [Model Based Scale and Rotation](#) in the [Using the Locator](#) section for more information.

Contrast Polarity

- Select a specific polarity option when it is known. The default is **Normal** polarity.
- The **Don't Care** mode is necessary to detect objects when there are local variations in contrast. It should only be used in such cases since it can significantly increase location time.
- Similarly, select **Normal and Reverse** only if there will actually occur instances of either polarity at runtime since this mode incurs a longer execution time than **Normal** or **Reverse** polarity options.

Instances to Find

- To reduce the search time, always set **Instances to Find** to the maximum number of instances that you expect to find in the image. An

unconstrained number of instances can lead to a long execution time since the Locator must thoroughly check for all possible instances, especially in cluttered images.

Timeout

- **Timeout** can be activated in time-critical applications to abort the search after a fixed amount of time. When a timeout condition is reached, only the instances found up to the timeout are output by the Locator.

Instance Ordering

- Enabling the **Quality** ordering mode may significantly increase the Search time when the image contains multiple instances of the Model or when there can be confusion with background features that are similar to the Model. In certain applications however, the benefits of Quality ordering may often outweigh this Search time drawback.

Search Based on Outline Only

- Enabling this parameter can significantly reduce the execution time at the expense of a loss in positional accuracy.
- This mode is best used for well-contrasted images that provide clear contours, with little noise.
- Because of the lack in positional accuracy, this mode it is suited to time-critical applications where objects need only be coarsely identified and located.

Symmetric Instances

- Enabling a search for symmetric instances may increase the search time because the Locator will try to refine the hypotheses for each object instance.
- If **Locate All Symmetric Instances** is disabled, the search process will be limited to refining and outputting only the best object hypothesis.

Recognition Parameters Affecting Performance

Recognition Effort

A more **Exhaustive** search setting is preferable in the case of:

- Models made up of small features at the Outline Level.
- Models made up of arrangements of features with few or no curvature variations at the Outline Level.
- Cluttered images and degraded contours broken into small segments.

A **Quicker** search setting is preferable in the case of:

- Models that are rich in curvature variations at the Outline Level.
- Multi-scale applications.
- Low-clutter images.
- Images containing unbroken contours

Conformity Tolerance

- If the features of the models to be recognized can be locally distorted in the runtime images, increasing the **Conformity Tolerance** can improve the robustness of the recognition process. Note that a large **Conformity Tolerance** can also trigger false recognitions and may slow the recognition process.

Minimum Model Percentage

- Increasing **Minimum Model Percentage** can accelerate the recognition process but can also significantly reduce robustness to occlusions. When significant occlusions are expected in the runtime images, **Minimum Model Percentage** should be decreased.
- When the recognition slider is set towards **Exhaustive**, low **Minimum Model Percentage** values can significantly increase the execution time.

- In multi-model applications, when the parts to be recognized are very similar, increasing **Minimum Model Percentage** may help avoid confusion between models.

Positioning Speed

- The **Positioning** slider has only a small impact on the overall speed of the Locator.
- In applications where accuracy is not critical, setting the slider towards **Fast** can provide an improvement in speed.
- Moving the slider towards **Accurate** increases the positional accuracy of found instances.

Locator Results

Locator results provide information on instances found by the Locator. The most frequently used results, described below, appear in the user interface. Other results, such as the position of reference points, are described in the ***Reference Manual***.

Results Viewer

The **Results Viewer** displays located instances in one of two modes: **Matched Model** or **Transformed Model** (see [Figure 165](#)). These modes are useful when setting up an application. Nevertheless, activating the Results Viewer imposes an overhead to the Locator since it must generate the graphical scenes. Thus, once an application is set up, it is preferable to deactivate it.

The graphic mode selected in the results viewer is independent of the output selected in the Output frame of the General panel. The General panel lets the user specify the mode that will be output by the Locator and consequently, the type of graphical Scene that will be made available to subsequent processes.

Matched Model Mode

This mode displays the located instances using only the sections of Model contours that were matched to the actual contours detected in the input image.

- The graphical results can be visualized at the Detail Level only, unless the search has been configured to **Search Based on Outline Only**.

Transformed Model Mode

This mode displays the located instances using the Model contours transformed according to the pose computed by the Locator. These contours should not be confused with the actual contours that were detected in the input image.

- The graphical results can be visualized at the Detail Level only, unless the search has been set as **Based on Outline Level Only**.

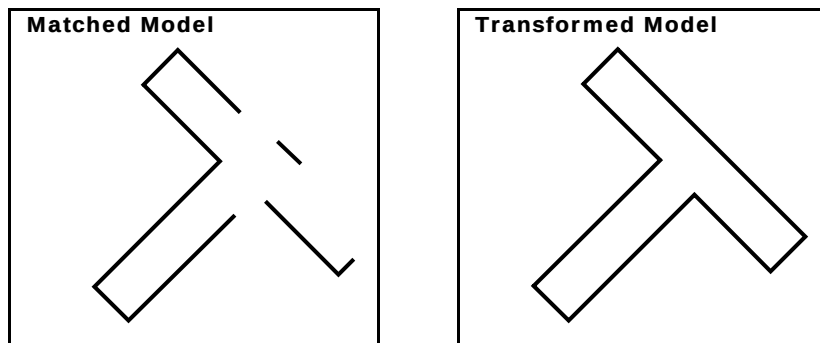


Figure 165 Locator Output Display Modes

Locator Results

The Locator outputs a set of results for every instance found in the input image.

Instance Identification and Model Type

Each located instance is identified by an instance number and the name of its Model type.

Location Properties

The location properties for located instances are: Scale, Rotation, X-Position and Y-Position.

The position of a given object instance corresponds to the origin of its Object coordinate system with respect to the World coordinate system of the input image. Units for position results are expressed in the user-selected unit system.

Rotation is defined by the counterclockwise angle between the Object coordinate system of the instance and the World coordinate system of the input image.

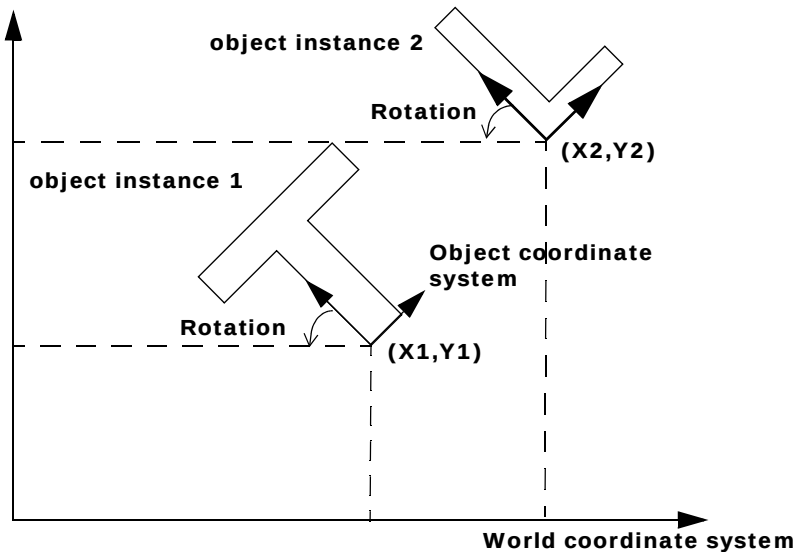


Figure 166 Location Properties

The following equation is used to transform the model contours of a given instance into the World coordinate system of the input image:

$$X_{World} = \bar{S}R_{\theta}X_{Object} + T$$

Where:

S is the scale matrix of the instance

R the rotation matrix of the instance

T is the translation vector of the instance

Fit Quality

The **Fit Quality** score ranges from 0 to 1, with 1 being the best quality. This value is the normalized average error between the matched model contours and the actual contours detected in the input image.

- A value of 1 means that the average error is 0. Conversely, a value of 0 means that the average matched error is equal to **Conformity Tolerance**.

Match Quality

Match Quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that 100% of the model contours were successfully matched to the actual contours detected in the input image.

Clear Quality

The **Clear Quality** result ranges from 0 to 1 with one being the best quality. A value of one means that the found instance is clear of obstacles. Clear Quality is computed only if the [Minimum Clear Percentage](#) search parameter is enabled.

Symmetry of

The **Symmetry of** index value is the index number of the instance of which the given instance is a symmetry of.

- For example, if a rectangular Model, named *Rectangle* is located in the image, with **Output all Symmetric Instances** set to true, the **Symmetry** result of will be output as illustrated in [Figure 167](#).

Instance Index	Model	Rotation	Symmetry of
0	Rectangle	4°	0
1	Rectangle	-140°	0

This is the first instance of the symmetry group

This second instance is a symmetry of instance 0

Figure 167 The “Symmetry of” Index Value

Time

The **Time** result provides the time that was needed to recognize and locate a given instance. The time needed to locate the first instance is usually longer because it includes all of the low-level image preprocessing.

Elapsed Time

The Locator process returns three elapsed time results: **Learn**, **Search** and **Total**.

- The **Learn** time is the time required to learn or re-learn models and parameters. There is a **Learn** process at the first execution of the Locator after any addition or change to Models or after a change to some Locator parameters.
- The **Search** time is the time required to locate all object instances found by the Locator.
- The **Total** elapsed time is the total time needed to execute the Locator. This time includes the overhead needed to create and output the result structures. It does not include the time needed to refresh the Results Viewer.

Search Result Messages

The Locator outputs diagnostic messages for the search process. [Table 5](#) presents the possible diagnostic messages, with their code numbers.

Table 5 Locator Search Result Messages

Code no.	Message
4500	Contour detection parameters are BASED ON ALL MODELS and Models have different detection parameters. Some instances might not be properly recognized.
4502	No hypotheses were generated.
4503	A large number of hypotheses were generated. This might incur a longer execution time.
4504	No features were set for location in the "name" Outline Model. The Locator will use all features for location.
4507	At least one instance was rejected because of a missing Required feature.
4508	At least one instance was rejected since it did not respect the Rotation constraint.
4509	At least one instance was rejected since it did not respect the Scale constraint.
4510	The Models were re-learned because of a change in parameters.
4511	Execution was interrupted since no Input Image was provided.
4512	Unsuccessful recognition. No Outline Models were provided.
4513	At least one instance was rejected because insufficient contours were found.
4514	Execution timed out.
4515	No features were set for Location in the "name" Detail Model. The Locator will use all features for Location.
4516	Unsuccessful recognition. No Detail Models were provided in a dual search.
4517	At least one instance that was recognized at the Outline Level was rejected at the Detail Level.
4518	No instance was found at the Outline Level. Verify Search parameters or edit the Model.

Table 5 Locator Search Result Messages

Code no.	Message
4519	Output of Outline Contour Scene is enabled: processing will be slower.
4520	Output of Detail Contour Scene is enabled: processing will be slower.
4522	Output of graphic instances in Instance Scene is enabled: processing will be slower.
4523	At least two Models are similar. Only one instance can be retained
4524	At least one Model is symmetric. Orientation may be unstable.
4525	Maximum Instance Count constraint is enabled. The Locator retained only the best <i>n1</i> instances of the generated hypotheses.
4526	Execution aborted.
4527	One of the models has a coordinate system type inconsistent with that of the input image. The search process may not work properly.
4530	Model Optimizer is enabled, processing will be slower.
4528	The size of the calibrated pixels in the input image is significantly different than the pixel size in one of the model images. Object recognition may fail.
4529	The size of the calibrated pixels in the input image is inconsistent with the pixel size in one of the model images. Inconsistent models will be ignored.
4534	At least one instance that was identified at the Outline Level was rejected at the Detail Level.
4535	At least one instance was rejected since it did not respect the Clear constraint.

Troubleshooting

- The Results Viewer, along with Model Editor and runtime Messages are useful tools for troubleshooting the Locator.
- Diagnostic messages are provided for Models, through the Model Editor, and for the Search process.
- When no object instance is found, check whether or not the object is recognized at the Outline Level. Do this by activating the **Based on**

Outline Level Only check box in the **Search** panel and executing the Locator again.

- If no instances are found, display the Contour Scene to check if the contours are properly detected.
- If the contours are not detected at the Outline Level, try setting the **Contrast Threshold** to **High Adaptive Threshold** or to a very low **Fixed** value.
- If the contours are correct at the Outline Level but the instances are not detected, try decreasing **Minimum Model Percentage**. If the instances are still not detected, try increasing Conformity Tolerance. The search constraints may also be too restrictive, try relaxing the scale factor and rotation ranges.
- You can use the Matched Model mode to view the instances and check which part of the model was matched to the detected contours.
- If an instance is recognized at the Outline Level but not at the Detail Level, try decreasing the **Minimum Model Percentage**.

Using the Locator

There are two setup Modes for the Locator Tool. By default, the Locator interface opens in the Basic setup mode, as shown in [Figure 168](#), in which “advanced parameters” are hidden from view. The advanced parameters are preset to default settings, which are typically sufficient for most object finding applications. The Advanced setup mode provides access to all Locator capabilities as shown in [Figure 169](#). These setup modes are provided to simplify the setup within the user interface; all Locator parameters are available programatically at any time, regardless of the **Advanced Parameters** setting.

General Panel

Use this panel to set the names for the Views, Scenes and Images output by the Locator and also to set the type of units the Locator tool will use to return results.

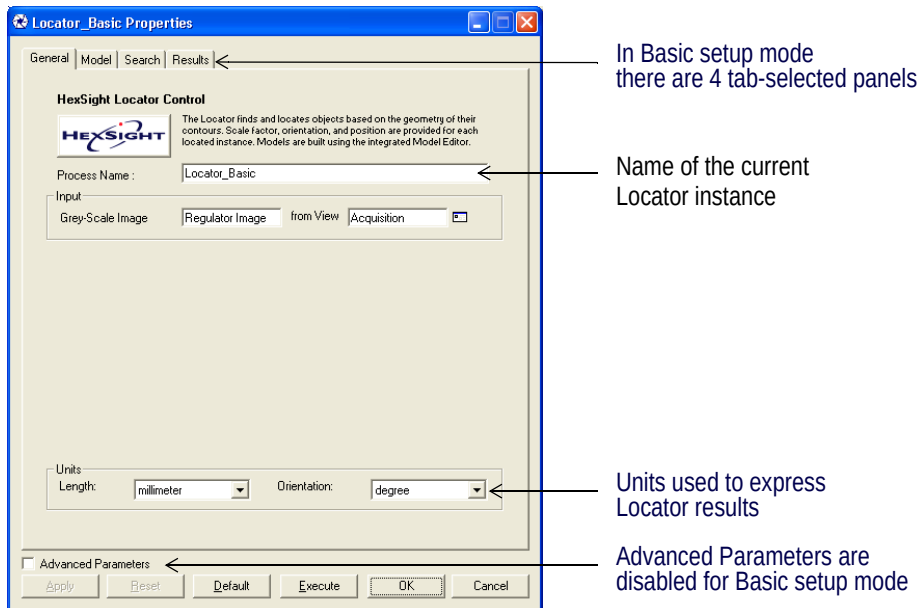


Figure 168 Locator — General Panel in Basic Setup Mode

Positioning Mode

The *positioning mode* defines how the tool's region of interest is positioned on the input Image. Four positioning modes are possible: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based automatic. See the [Inspection Tools](#) chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the inputs you need to provide to the tool as well as the value of the **Express Position and Size using Calibrated Units** parameter.

HexSight provides the capability to configure a Locator tool that uses model-based positioning. To set up a model-based Locator, your application must have at least one initial Locator that is configured to output an [Instance Scene](#) and [Model View](#).

IMPORTANT: THE INITIAL LOCATOR IN YOUR APPLICATION SHOULD USE **IMAGE-BASED CALIBRATED** POSITIONING. SUBSEQUENT MODEL-BASED LOCATORS WILL TYPICALLY USE **MODEL-BASED AUTOMATIC** POSITIONING.

[Table 6](#) provides an overview of the required inputs and settings for the four positioning modes.

Table 6 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Automatic Model-Based	Manual Model-Based
Inputs	Image	Image	Image Model Scene Instance Scene	Image Model Scene
Model Scene Input	disabled	disabled	enabled	enabled
Instance Scene Input	n/a (automatically disabled)	n/a (automatically disabled)	enabled	disabled

Table 6 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Automatic Model-Based	Manual Model- Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	The region of interest is automatically applied to a selected object instance.	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a Locator name, the Process Manager assigns a default name: HSLocator#, where # is replaced by its rank number in the list of instantiated processes.

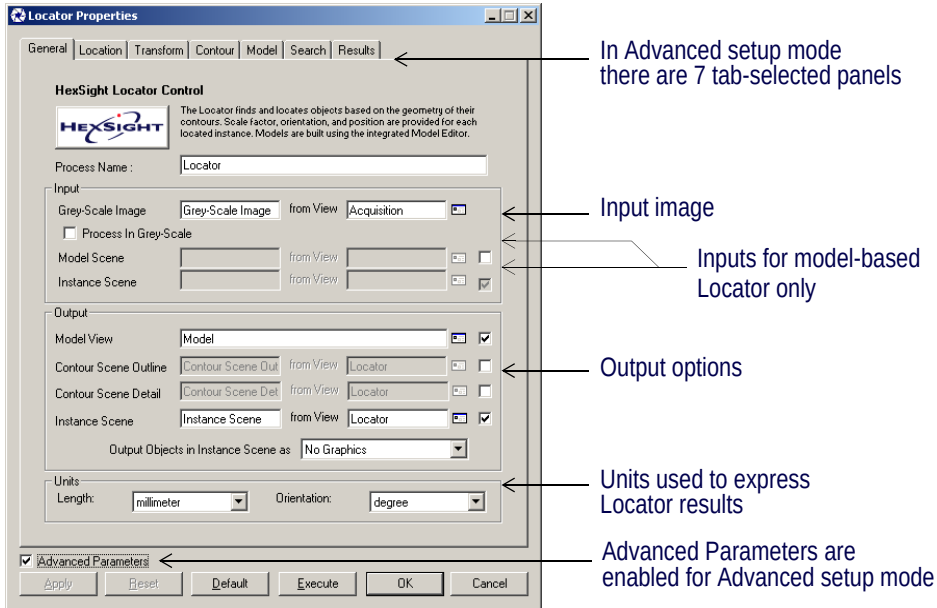


Figure 169 Locator — General Panel in Advanced Setup Mode

Inputs

Image

Image is the Image in the runtime database on which the process is applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale, when selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Model Scene

Enable **Model Scene** only if you are setting up a model-based Locator.

The **Model Scene** is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This Scene is provided by a HexSight Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Instance Scene

Instance Scene is automatically enabled when you enable a **Model Scene**.

The **Instance Scene** is the Scene in the runtime database that provides the transform parameters for object instances found by another Locator tool that is configured to output its Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

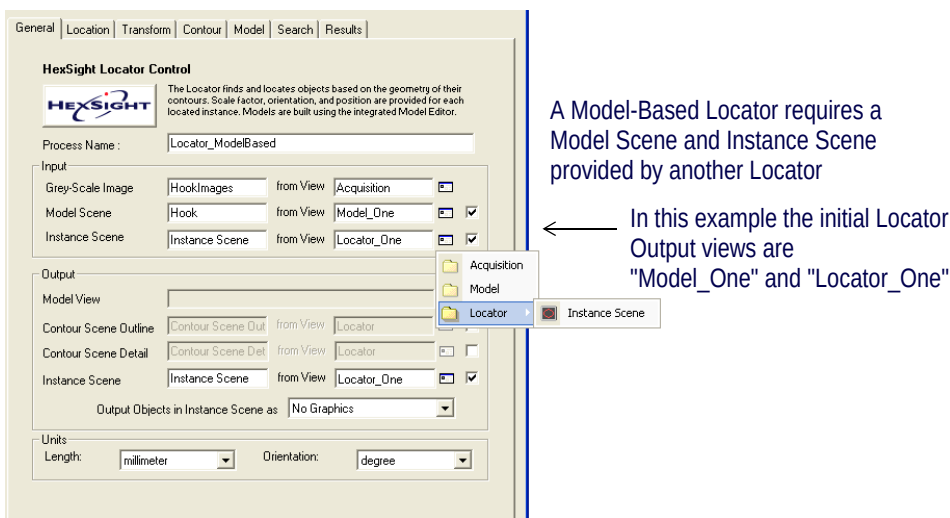


Figure 170 Locator —Model-Based Locator Inputs

Output

Use check boxes to enable or disable the output of the Images, Scenes and Views to the application's runtime database. Disabling non-required Locator outputs typically reduces the Locator execution time.

Model View

The Model View contains both an image and a Model Scene for each Model activated in the models database. This output is required by model-based tools in the application.

- You can disable Model View if no other tools require this output, for example if there are no model-based tools in your application.
- If your application has more than one Locator tool, make sure you name the Model View output differently for each Locator.

Contour Scene Outline

The Outline Contour Scene contains the vectorized description of the contours generated by the Locator at the Outline Level.

- Since **Contour Scene Outline** is not required as input by any other HexSight tools, you should typically disable this output to obtain faster performance.
- It can be useful to enable this output to display detected contours when you are setting up and configuring the Locator from the user interface of your application.

Contour Scene Detail

The Detail Contour Scene contains the vectorized description of the contours generated by the Locator in the Detail Level.

- Since **Detail Scene Outline** not required as input by any other HexSight tools, you should typically disable this output to obtain faster performance.
- It can be useful to enable this output to display detected contours when you are setting up and configuring the Locator from the user interface of your application.

Instance Scene

The Instance Scene contains the pose and type of model of all the object instances found by the Locator within the current image. It may optionally contain a graphic representation of the instance.

- You can disable **Instance Scene** if no other tools require this output, for example if there are no model-based tools in your application.

Output Objects in Instance Scene as

Use **Output Objects in the Instance Scene as** to select the output display mode: **No Graphics**, **Transformed Model** or **Matched Model**.

- **Matched Model** displays only the sections of the Model contours that can be matched to the actual contours detected in the input image.

- **Transformed Model** displays the complete Model transposed according to the pose of the instance.
- The graphic output modes are provided for displaying located instances in the user interface of your application. For faster performance you should always set **Output Objects in the Instance Scene as** to **No Graphics**.
- For troubleshooting the Locator when developing an application, you should use the **Results Viewer** instead of using a graphic output mode for the Instance Scene. See the [Results Viewer](#) section for details on interpreting Locator outputs.

Units

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Advanced Parameters

Use the Advanced Parameters check box to toggle the setup mode. When the Advanced Parameters are disabled, the Locator is in **Basic** setup mode. This mode hides more advanced parameters and functionality from view, to provide a simpler setup. The Basic setup is typically sufficient for most simple (basic) vision application.

When the **Advanced Parameters** check box is enabled, the Locator is in **Advanced** setup mode and all Locator parameters are visible in the user interface.

The **Advanced Parameters** check box affects the user interface only. All Locator properties and methods remain available programmatically even if they are hidden from the user interface.

Note:The following are hidden from the interface in Basic setup mode: All model-based related parameters, the Location, Transform, Contour tabs, Model Optimization and Model Disambiguation parameters, Results Viewer options. The displayed Search parameters and constraints are: Scale, Rotation, Number of Instances to Find and Minimum Model Percentage.

Location Panel

Use this panel to position the tool's **Rectangle**, which defines the region of interest in the image to which the process will be applied.

Note:In **Basic** setup mode, this panel is hidden from the user interface. By default, the Rectangle region of interest is the entire input image.

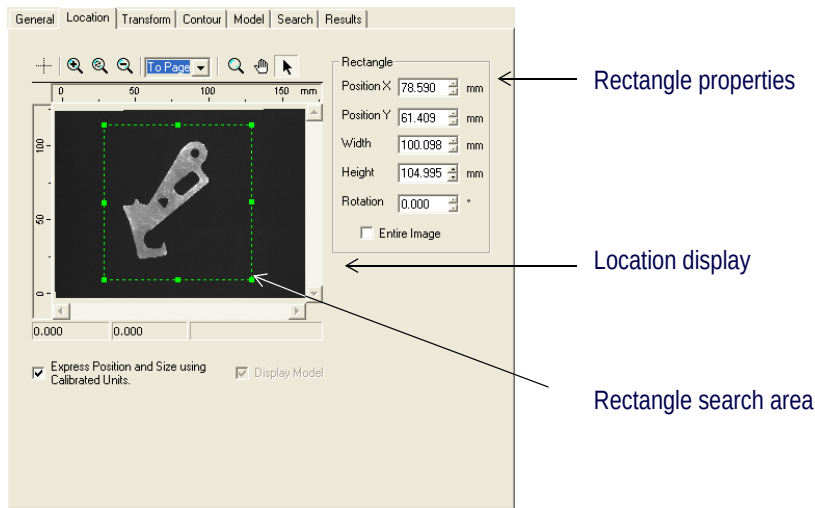


Figure 171 Locator — Location Panel

Location Display

The Location display presents an image, used to position the tool, and the Rectangle region of interest. By default, the **Entire Image** checkbox is

enabled, in which case the Rectangle boundaries are not visible. To manually set Rectangle boundaries, disable the **Entire Image** check box.

- For a non-model based Locator (image-based positioning) the **Rectangle** region of interest is positioned directly on the input Image. The display shows the current input Image.
- When the Locator is Model-based, the **Rectangle** region of interest is positioned on the Model Image. The display shows the currently selected Model.

Rectangle Properties

The tool's region of interest is the **Rectangle** that is defined by: **Position X**, **Position Y**, **Width**, **Height**, and **Rotation**.

In the Image based positioning mode (i.e. not model-based), the **Rectangle** is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position and Size using Calibrated Units** check box.

- When **Express Position and Size using Calibrated Units** is enabled, **Position X**, **Position Y**, **Width**, **Height** are expressed in the World Coordinate system, in the selected calibrated unit.
- **Rotation** is always expressed in the selected orientation unit.

When the Locator is Model-based, the Rectangle is positioned with respect to the Object coordinate system and **Position X**, **Position Y**, **Width**, **Height**, and **Rotation** are always expressed in the selected calibrated unit.

Placing and Sizing the Rectangle

By default the **Rectangle** bounding box is set to **Entire Image**. To change the size and position of the rectangle:

- Disable the **Entire Image** check box.
- Move the bounding box by dragging it into place.
- Size the box by dragging its corner and side handles.

- You can also change height, width and position by entering values in the corresponding boxes.

Express Position and Size using Calibrated Units

Use the check box labeled **Express Position and Size using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning.

- This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.
- With Model-based positioning, positions and dimensions are always expressed in calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Transform Panel

The Transform panel is active only if the Locator is Model-based. Use this panel to set the transform that will be applied to the tool before processing the input Image.

Note:In **Basic** setup mode, this panel is hidden from the user interface.

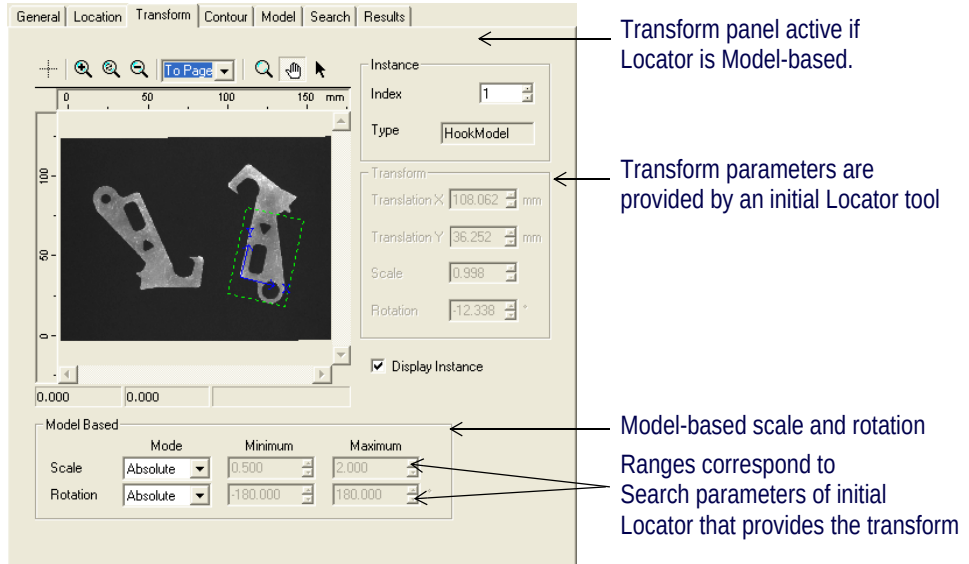


Figure 172 Locator — Transform Panel of a Model-Based Locator

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index and Type

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**,

Translation Y, Scale and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation.

- When Model-based manual positioning is enabled, the **Translation X, Translation Y, Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.
- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene. This Instance scene is provided by another Locator tool.

Model Based Scale and Rotation

When the Locator is model-based, both scale and rotation ranges can be provided here instead of configuring ranges in the Locator's Search parameters.

Scale

- The **Absolute** mode is the default recommended mode.
- In **Relative** mode, you must provide a scale range that will inform the tool on the possible scale values of the selected instance.
- The defined scale range should normally be the same as the **Scale** constraint defined in the **Search** of the initial Locator tool that is providing the Instance Scene. Providing these range values allows relative use of the current tool's **Scale** constraints to achieve better performance.
- If your application must find a sub-feature, on an instance of a "parent" object that may vary in scale, the range specified in the **Relative** mode enables the model-based Locator to search for a sub-feature at a nominal scale corresponding to the scale of currently selected instance. For example, if the initial Locator is configured to locate instances ranging from 0.5 to 2.0 in scale, and the **Scale** constraint of the model-based Locator is enabled to a **Nominal** value of 1.0, a specified instance having a scale value of 0.75 will make the tool search only for sub-features of 0.75 in scale. If instead, the Scale constraint is set to a range from 0.9 to 1.1, for the same 0.75 scale instance, the Locator will search for sub-features scaled between 0.675 and 0.825.

Rotation

- The **Absolute** mode is the default recommended mode.
- The defined rotation range should normally be the same as the **Rotation** constraint defined in the **Search** of the initial Locator tool that is providing the Instance Scene. Providing these range values allows relative use of the current tool's **Rotation** constraints to achieve better performance.
- If your application must find a sub-feature on a "parent" object that may vary in rotation, the range specified in the **Relative** mode enables the model-based Locator to search for a sub-feature at a nominal rotation corresponding to the rotation of currently selected instance.

For example, if the initial Locator is configured to locate instances ranging from -180 to 180, and the **Rotation** constraint of the model-based Locator is enabled to a **Nominal** value of 0, a specified instance at a rotation of 45 will make the tool search only for sub-features that are rotated to 45 degrees. If instead the Rotation constraint is set to a range from -10 to 10, for the same instance rotated at 45 degrees, the Locator will search for sub-features rotated between 35 and 45 degrees.

Contour Panel

Use this panel to set contour detection parameters and view the contours generated by the Locator.

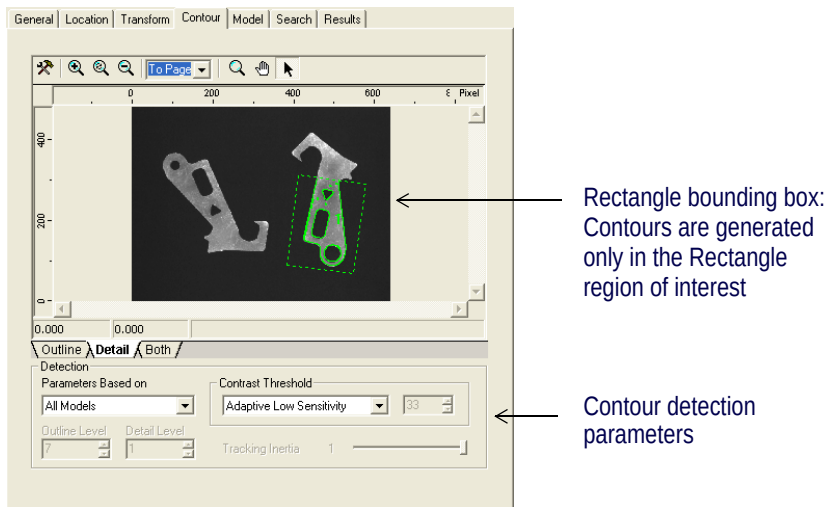


Figure 173 Locator — Contour Panel

The **Outline** display represents contours generated at the Outline Level as magenta lines. The **Detail** represents contours generated at the Detail Level as green lines. The **Both** display shows the superposed Outline and Detail Levels contours.

Contour Detection Settings

The **Detection** frame provides options and settings for contour detection.

Parameters Based on

The **Parameters Based on** combo box sets one of three modes for applying detection parameters.

- The **Custom** mode requires that you manually set the Outline Level and Detail Level parameters. The other two modes: **All Models** or a **(Selected)** Model, automatically apply a set of predetermined detection parameters.
- [Table 7](#) presents how detection parameters are set for each of the three parameter-selection modes.

Table 7 Application of Contour Detection Parameters

Parameters Based on:	Outline Level	Detail Level
Custom	User-defined.	User-defined. Lower than or equal to Outline Level
All Models	From the Model with the lowest Outline Level	From Model with the highest Detail Level that is also lower than or equal to the Outline Level.
<i>Selected</i> Model	Outline Level of the selected Model.	Detail Level of the selected Model.

Outline Level

The **Outline Level** is used to rapidly identify potential instances of the object. Its coarseness setting ranges from 1-16.

- Values can be set manually only if **Parameters Based on** is set to **Custom**.
- The higher the setting, the coarser the contour resolution.
- The **Outline Level** value must be lower than, or equal to, the **Detail Level**.

Detail Level

The **Detail Level** is used to confirm recognition and refine the position of valid instances. Its coarseness setting ranges from 1-16.

- Values can be set manually only if **Parameters Based on** is set to **Custom**.
- The higher the setting, the coarser the contour resolution.
- The **Detail Level** value cannot exceed the **Outline Level**.

Contrast Threshold

Contrast Threshold sets the sensitivity to the contrast when detecting source contours in the input image. With Adaptive settings, the Locator automatically sets the numerical value according to the input image. describes the available Contrast Threshold modes.

- The default **Model Values and Mode** setting is recommended for most applications.
- **Adaptive High Sensitivity** accepts lower contrasts and therefore results in a greater number of source contours. **Adaptive Low Sensitivity** retains high-contrast contours and removes lower contrast contours, such as those caused by noise and shadows
- **Fixed Value** requires that you manually set the Contrast Threshold value, expressed as a step in intensity values, ranging 0 to 255. Higher values reduce sensitivity to contrast, resulting in fewer, high-contrast contours. Conversely lower values increase sensitivity and add a greater amount of source contours.
- The **Contrast Threshold** can also be based on **Model** values. These modes are explained in [Table 8](#).

Table 8 Description of Contrast Threshold Modes

Contrast Threshold Setting	Description
Adaptive Normal Sensitivity	Accepts typical contrasts.
Adaptive Low Sensitivity	Accepts low contrasts, resulting in a greater number of source contours.
Adaptive High Sensitivity	Retains high-contrast contours and removes lower contrast contours, such as those caused by noise and shadows.
Fixed Value	Expressed as the step in intensity values. Higher values reduce sensitivity to contrast, resulting in fewer, high-contrast contours. Lower values increase sensitivity and add a greater amount of source contours.
Model(s) Mode	Uses the most sensitive mode, selected from the Models specified by the Parameters Based on selection.
Model(s) Value	Uses the most sensitive value, selected from the Models specified by the Parameters Based on selection.
Model(s) Value and Mode	Default recommended setting. Uses either the most sensitive mode or the most sensitive value, selected from all the currently active Models specified by the Parameters Based on selection.

Model Panel

The Model sub-process creates a reference or “ideal” contour representation of an object that will be used by the Locator process. Models integrate image calibration parameters and can thus be stored for later on other imaging setups.

You will use this panel to manage the models database and access the Model Editor and model Optimizer dialogs.

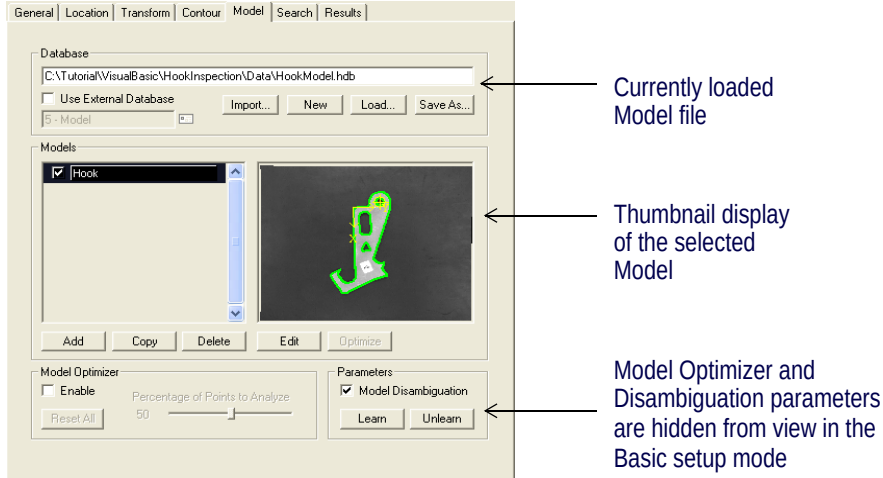


Figure 174 Locator — Model Panel

Database

The **Database** frame contains the file name text box, showing name of the currently loaded Model database, command buttons for managing model databases: **Import**, **New**, **Load** and **Save As** and the **Use External Database** option.

Database Command Buttons

- Use **Import...** to import selected files into the current Model database through the Import Models dialog ([Figure 175](#)). Formats that are supported for importing are hdb (HexSight database), hig (HexSight image), bmp and dxf.
- Use **New...** to clear the current models database. This resets the Model panel interface.
- Use **Load...** to load a Model database file. The newly loaded Model file replaces the current Model database. Only hdb (HexSight database) files can be loaded to this database.

- Use **Save As...** to save the current database to an hdb (HexSight database) file. A models database can contain any number of Models. Unsaved Model information will be lost once you close the application.

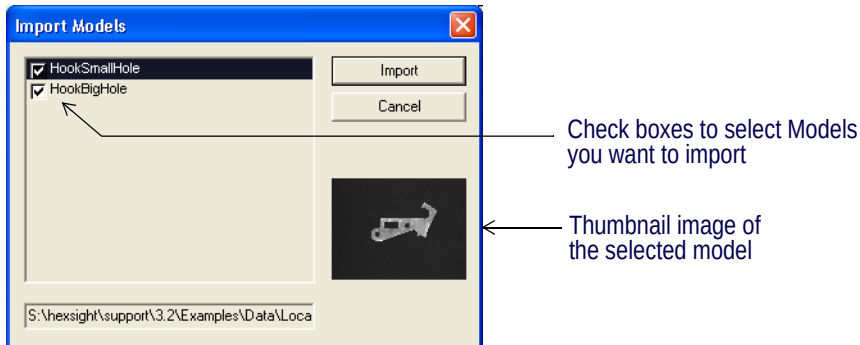


Figure 175 Importing Models

- Enabling **Use External Database** allows the Locator to use a models database provided by another instance of the Locator tool in the current application.

Note: Model databases that are too large are long to load and use a significant amount of memory.

Models

The **Models** frame displays the contents of the current models database, a thumbnail view of the selected and commands for editing the database.

- You can enable or disable individual Models in the database using the associated check boxes.

Models Command Buttons

- Click **Add** to access the Model Editor in which you will create a new Model. You must first provide a Model Name in the Create New Model dialog that appears. To edit the name of a Model in the list, double-click on its name in the list.

- Use **Copy** to create a new Model that is identical to the one currently selected in the list of Models. You are prompted to provide a name for the new Model.
- **Delete** removes the selected Model from the database. Once a Model is deleted, saving the current Model database will override the old database, thus permanently removing the deleted model.
- **Edit** opens the Model Editor dialog, with the Model selected in the list of Models. Unsaved modifications are applied only to the current Models database and lost once you close the application unless you re-save the modified models file. See the [Using the Model Editor](#) section for more details.
- **Optimize** opens the Model Optimizer dialog in which you can set up and observe an ongoing Model optimization process. To activate the **Optimize** command you must first check the **Enable** check box, which launches an optimization of the Model that is currently selected in the list of Models.

Model Optimizer

The optimization process builds a model from multiple instances of an object. Optimization is typically used when objects to be recognized have a variable shape, when it is impossible to build a typical or satisfactory model from a single instance, or when a single model may contain unwanted features resulting from noise or irregularities in the object and manual editing of features is not practical.

Model Optimizer Settings

- Checking the **Enable** box enables the launching of an Optimization process.
- **Reset All** restarts the optimization process and discards the instances that have been appended so far to the process. This resets to zero the counter to the right of the selected Model in the Database frame.
- The **Percentage of Points to Analyze** slider sets the percentage of points that are analyzed in each instance that is used for optimization.

At the default 50% position, one out of two points are used. Increasing this value can increase the accuracy of the optimized model but may incur a longer optimization time.

Executing a Model Optimization

Model optimization must be launched from the Model Panel of the Locator.

1. To start an optimization process you must select an initial model in the Models database.
2. To launch the optimization, check the Model Optimizer **Enable** check box then click **Apply**.
3. Click **Optimize** to open the Model Optimizer window in which you can change parameters and observe the progression of the optimization process. See the [Using the Model Optimizer](#) section for more details.
4. Execute the Locator on various instances. While the optimization process is running, each instance found by the Locator is added to the current optimization. In the Model panel, the number of instances processed appears to the right of the corresponding model. In the Model Optimizer window the number of instances used by the process appears in the **Instance Analyzed** textbox.
5. The optimization can be restarted from the initial model by clicking **Reset All** from the Locator interface or **Reset** from the Optimizer window.
6. To stop the optimization process and use the optimized model, click **Apply** from the Model Optimizer window.

Model Disambiguation

- By default, model disambiguation is always enabled. The default enabled state is recommended in most cases.
- Disabling **Model Disambiguation** can significantly reduce the Learn time when this is an issue.

IMPORTANT: DISABLING MODEL DISAMBIGUATION WILL IMPAIR THE LOCATOR'S ABILITY TO DISCRIMINATE BETWEEN VERY SIMILAR MODELS OR BETWEEN VERY SIMILAR HYPOTHESES OF AN OBJECT.

Learn and Unlearn Commands

- Executing the **Learn** command initiates a Learn process without executing the Locator. The next execution of the Locator will be shorter because no Learn process will be need to be carried out.
- Executing the **Unlearn** command clears all the learned information in memory. The next execution of the Locator will be longer because the Locator will have to re-learn all the enabled models in the Models database.

Using the Model Editor

The Model Editor is used to define new Models and to modify existing ones. Once you have created the Model for a type of object or part, you will save it for later use with any HexSight application.

The typical steps for building a Model are as follows:

- Set a **Bounding Area** to define the area of the input image that will be used to build the Model.
- Set contour detection parameters for the source contours that will be generated from the input image.
- Set feature selection parameters for the automatic selection of model features.
- Launch the **Build** the command to generate a Model.
- If necessary, edit the features that were automatically selected.
- Set the orientation and position of the Object coordinate system.

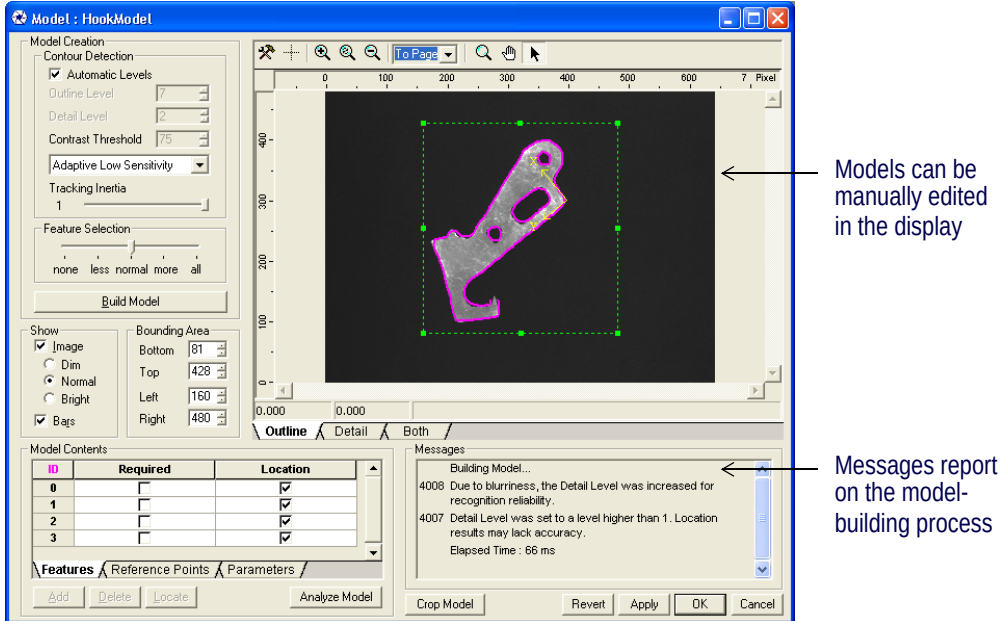


Figure 176 Model Editor

Model Creation

The Model Creation frame provides Contour detection and Feature selection parameters as well as the **Build** command. The Contour Detection parameters are the same as those available in the Contour panel.

Teaching and Building a Model

To teach the Model, simply place and size the Bounding Area marker around the region of interest in the display and click **Build Model**.

- The default Contour Detection settings are recommended.
- Use the **Feature Selection** slider to increase or decrease the amount of features that are selected from the source contours when you teach the Model.
- Unless you want to specify the coarseness for the Outline and Detail Levels, leave the **Automatic Levels** check box enabled.

- If you want to manually select all the features for a Model, set **Feature Selection** to none, teach the Model with the required **Contour Detection** parameters to detect the source contours, then manually add features. See [Adding Features to the Model](#).

Automatic Levels

When the **Automatic Levels** check box is enabled, the Locator automatically optimizes the **Outline Level** and **Detail Level** values.

- **Automatic Levels** is the recommended mode.
- To manually set Outline and Detail Levels parameters you must disable the **Automatic Level** check box.

Outline Level

The **Outline Level** provides a coarser level of contours than the **Detail Level**. The Locator uses Outline contours to rapidly identify and roughly locate potential instances of the object.

- The higher the setting (maximum **16**), the coarser the contour resolution.
- The **Outline Level** value can only be higher than, or equal to the **Detail Level**.

Detail Level

The Locator uses **Detail Level** contours to confirm the identification of an object instance and refine its location within the image.

- The lower the setting (minimum **1**), the finer the contour resolution.
- The **Detail Level** value is always lower than, or equal to the **Outline Level**.

Contrast Threshold

Contrast Threshold sets the Locator's sensitivity to contrast used in generating contours. With **Adaptive** settings, the Locator automatically sets the numerical value according to the input image.

- The default **Adaptive Normal Sensitivity** is recommended for most applications.
- **Adaptive High Sensitivity** accepts lower contrasts and therefore results in a greater number of source contours. **Adaptive Low Sensitivity** retains high-contrast contours and removes lower contrast contours, such as those caused by noise and shadows.
- **Fixed Value** requires that you manually set the **Contrast Threshold** value, expressed in terms of a step in intensity values that ranges from 0 to 255. Higher values reduce sensitivity to contrast, resulting in fewer, high-contrast contours. Conversely lower values increase sensitivity and add a greater amount of source contours.
- The **Fixed** mode is useful for controlled environment applications in which only contours above a predefined threshold should be processed.

Tracking Inertia

Setting the **Tracking Inertia** slider to **1** can close *small* gaps that appear in the source contours.

Build Model

Build Model initiates the model creation process, which clears the current Model, re-detects the source contours using the new settings and selects features from the source contours according to the setting of the **Feature Selection** slider.

- The **Message** frame displays messages pertaining to the model creation process, including problems that the Locator encountered in building the Model or potential problems it may encounter with this Model in the search process.
- **Build Model** does not make any changes to reference points or to the coordinate system.

Show Options

Options in the **Show** box modify the characteristics of the display window but do not affect the model-building process.

Image

When **Image** is enabled, the Model Editor display shows the source image used to build the Model. When disabled, the display shows only the source contours that were detected in the Bounding Area.

- Three **Image** display modes are available: **dim**, **normal** and **bright**. These modes only affect the appearance of the image in the Model Editor. The image intensity values are not modified.
- Switching to a different display mode may help to better visualize elements of the display such as the source contours or the axes marker.

Bars

When **Bars** is enabled, the rulers, scroll bars and the information bar of the Model Editor display are visible.

Bounding Area

The **Bounding Area**, represented in the display as a green rectangle, defines the area of the current image that will be used to create the Model.

- Adjust the size of **Bounding Area** in the display by dragging any of the eight corner and side handles.
- You can also position and size the **Bounding Area** by selecting or entering values for the **Bottom**, **Top**, **Left** and **Right** properties.

Editing Models

The Locator creates features once you use the **Build Model** command in the Model Editor. However you may need to edit the Model to the needs of your application. For example, you may not need the entire contour of a feature.

Adding Features to the Model

You can manually select and add features to a Model once it has been built. Added features can be created at either the Detail Level or the Outline Level.

Follow the steps below to add features to the model. Added features appear in the **Features** sub-panel, at bottom left of the Model Editor interface.

- You cannot edit an existing feature. You must first delete the feature then create a new feature from the required portion of contour.
- In the display, select the level in which you will create the feature: Outline or Detail.
- If you want to manually select all your model features, set the Feature Selection slider to **none**, then click **Build Model**.
- Select the feature either by clicking on its description in grid of the Features tab, or by clicking on the feature in the display.
- To select the entire contour, double-click on the contour.
- To select only a section of a contour, left-click a starting point on the contour. Hold the **Ctrl** key and click the end point of the portion you want to select.
- To modify (add or subtract) from the selected contour section, hold the **Ctrl** key while clicking elsewhere on the line segment.
- To invert the your selection, hold the **Shift** key while clicking on the contour. This is useful for selecting the complementary portion of a closed contour.
- Clicking **Add** adds the selected feature to the list of features in the **Features** sub-panel (tab-selected).

Removing Features from the Model

Use the grid in the **Features** sub-panel to delete features from the Model or work directly in the display.

- Select the feature you want to remove: the feature appears in the display as a bold red contour.
- Click **Delete** to remove the feature.

Keyboard Combinations in the Model Editor Display

[Table 9](#) describes keyboard combinations that can be useful for selecting features, in the Model Editor Display.

Table 9 Model Editor Selection Keys

Key combinations	Effect on an existing feature	Effect on a contour (blue)
[Click]	Selects the feature.	Selects a segment on the contour and highlights the entire unbroken contour.
CTRL+[click]	Selects/Deselects the feature. Allows selection of multiple features.	Removes/adds segments to the selected contour.
[Double-Click]	Selects the feature. (same as [Click])	Selects the entire unbroken contour or selects the contour sections required to close an unbroken contour that already contains existing features.
SHIFT+[click]	Selects the feature. (same as [Click])	Toggles the selected/non-selected portions of the contour.
[Drag]	Selects only the features entirely contained in the dragged area	No effect
CTRL+[Drag]	Adds the features entirely contained in the dragged area to the selection.	No effect
ALT+[Drag]	No effect.	Selects all contour sections included in the dragged area.
CTRL+ALT+[Drag]	No effect.	Adds all contour sections included in the dragged area to the selected contours.

Features Sub-Panel

The grid in **Features** sub-panel lists the features that constitute the Model. The **Required** and **Location** properties can be enabled/disabled independently for each feature in the grid.

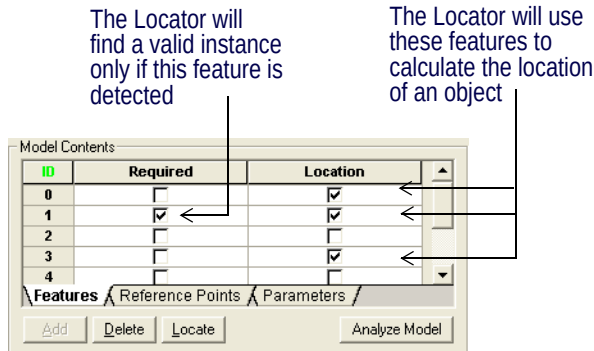


Figure 177 Model Editor — Features Sub-Panel

Location

When **Location** is enabled for a feature, this feature is used by the **Location** to calculate the location of object instances. Features enabled for Location should be stable. If **Location** is disabled, the feature is used only for object recognition.

- For example, a tag or label that is glued to an object often varies in position. Such a label is a good candidate for object recognition but should not be used for determining the precise location of an object.

Required Features

When the **Required** check box is enabled, the Locator will not identify an object in which the **Required** feature has not been found.

- A feature can be enabled as Required even if it is not used for Location.

Choosing Features for the Model

The following considerations should be taken into account when selecting features used to create the Model.

- Select features that are stable, that is features that remain fixed or stable on all occurrences of the part or object you are modeling.

- Select features that distinguish the part from the background and from other similar parts or objects that will be processed within the same application.

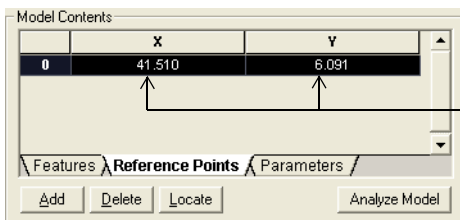
Reference Points Sub-Panel

Reference Points indicate the location of points of interest on an object. Their arrangement remains rigid in the Object coordinate system. Reference Points are not observable features and therefore are not used by the Locator for recognizing object instances. They are simply a set of coordinates that are transformed according to the pose of recognized instances of the model.

Adding Reference Points

Reference points appear as target-shaped markers. To add reference point:

- Select the **Reference Points** tab ([Figure 178](#)) then click **Add**.
- The reference point appears as a red target marker in the display. Drag the marker to the spot on the object you wish to identify with a reference point.
- You can also enter and adjust the reference point position by manually entering X-Y coordinates in the appropriate text boxes.
- To remove a reference point, select it and then click **Delete**.



Coordinates of the Reference point referenced in the Object coordinate system

Figure 178 Model Editor — Reference Points Sub-Panel

Parameters Sub-Panel

The **Parameters** sub-panel ([Figure 179](#)) provides access to the **Coordinate System** parameters and to the Custom Shading Area setup.

Coordinate System

The Coordinate System properties set the origin and the orientation of the Object coordinate system. By default, the origin is set at the center of the image and the orientation is set to zero degrees. You can manually set or change the Object coordinate system parameters:

- Displace the coordinate system origin by dragging its marker in the display. Rotate the axes by dragging the X or Y extremity of the marker.
- You can lengthen the axes of the marker to help you align the coordinate system. Lengthening the axes marker has no impact on the Object coordinate system.
- You can also manually enter the required coordinates and orientation.

X and Y Position

These editable text boxes show the position of Object coordinate system within the World coordinate system.

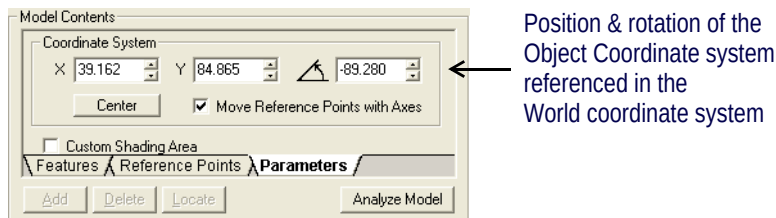


Figure 179 Model Editor — Parameters Sub-Panel

Rotation

The editable rotation text box, indicated by an angle icon, shows the orientation of the Object coordinate system with respect to the World coordinate system of the image used to teach the model.

Center

With a feature selected in the Model Scene, **Center** sets the origin of the Object coordinate system at the feature's center of gravity.

Move Reference Points with Axes

If you have created reference points in your model, this option will lock their position with respect to the coordinate system marker.

- Use this option if your reference points have been manually set by their values, with respect to known coordinates on the object. If you readjust the coordinate system origin, you will not have to re-enter the new **Reference Point** positions.

Custom Shading Area

Enabling the **Custom Shading Area** check box allows you to manually set up an area of the model that the Locator will use for shading consistency analysis. The Locator analyzes shading consistency by comparing the custom area in the Model to the corresponding area on a found instance.

- This shading area is used by the Locator when [Instance Ordering](#) set to *Shading Consistency*.
- If no **Custom Shading Area** is defined, the Locator uses the entire Model area for shading analysis.

Model Editor Command Buttons

- The **Add** button adds a selected feature or a reference point to the Model.
- **Delete** removes the selected feature or reference point from the Model.
- **Locate** repositions the display view so that the selected feature or reference point appears at center of the Model display.
- **Analyze Model** analyzes a Model that you have manually edited or created. A set of messages reports the results of this analysis in the **Messages** frame.
- **Crop Model** reduces the Model to the area contained within the **Bounding Area**. The cropped image may slightly exceed the limits of the bounding box. Cropping the Models reduces the size of the Model

database and can help reduce runtime memory usage and database load/save time, especially in multi-model applications.

Note: *Once you have cropped the Model you cannot return to the original image you used to build the Model.*

- **Revert** undoes any modifications made since the last **Apply** or **OK**.
- **Apply** registers changes made to the Model; the Model Editor remains open.
- **OK** applies the latest modifications made to the Model and closes the Model Editor.
- **Cancel** closes the Model Editor window and cancels all modifications made to the Model since the last **Apply** or **OK**.

Messages

The **Message** frame provides a brief report on Model building as well as troubleshooting information to help resolve Model creation or edition problems. The complete list of messages is provided in of the [Understanding Locator Concepts](#) section.

Using the Model Optimizer

The Model Optimizer window allows you to set parameters for the optimization process and observe the progression of the Model that is being optimized. To open the Model Optimizer window, illustrated in [Figure 180](#), click the **Optimize** command from the Model panel of the Locator interface.

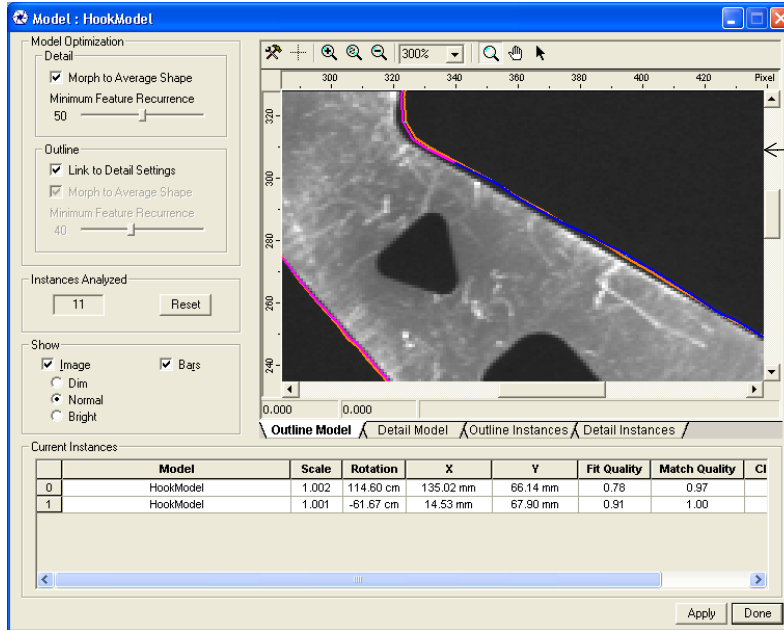


Figure 180 Model Optimizer Window

Setting Model Optimization Parameters

Model optimization parameters can be set independently for both the Detail and Outline level of the model.

Link to Detail

When **Link to Detail** is enabled, the tool automatically sets Outline level settings relative to the Detail level settings.

- **Link to Detail** should typically be enabled.
- To manually set **Outline** level parameters you must disable **Link to Detail**.

Morph to Average Shape

Morph to Average Shape enables the Optimization process to add recurring features that were not present in the initial model.

- If you disable **Morph to Average Shape**, features that were *not* present in the initial model will *not* be appended to the optimized model *even* if they recur within the required **Minimum Feature Recurrence**.
- When **Link to Detail** is enabled, the **Morph to Average Shape** state set for the Detail setting will be the same at the Outline Level.

Minimum Feature Recurrence

Minimum Feature Recurrence sets frequency of recurrence that is required for a feature to be retained and used for the optimization process. For example, at the default setting of 50, a feature will be retained for the optimized model only if it occurs in at least 50% of all analyzed instances.

- If **Link to Detail** is enabled, the Outline level feature recurrence is equal to the Detail level value minus ten (**Minimum Feature Recurrence** - 10).

Instances Analyzed

The number of instances used for the current optimization process is displayed in the **Instances Analyzed** frame.

Use the **Reset** command at any time to restart the optimization process from the initial model. The **Instances Analyzed** counter will be reset to 0.

Show Options

Options in the **Show** box modify the characteristics of the display window but do not affect the optimization process.

Image

When **Image** is enabled, the Model Optimizer display shows the source image used to build the Model. When disabled, the display shows only the source contours that were detected in the Bounding Area.

- Three **Image** display modes are available: **dim**, **normal** and **bright**. These modes only affect the appearance of the image in the Model Optimizer. The actual pixel values are not modified.

- Switching to a different display mode may help to better visualize elements of the display such as the source contours or the axes marker.

Bars

When **Bars** is enabled, the rulers, scroll bars and the information bar of the Model Optimizer display are visible.

Current Instances

The **Current Instance** grid displays the Locator results for each object instance found in the current image. When multiple instances of the object are found in the same image, as illustrated in [Figure 181](#), all found instances are used and appended to the optimized model.

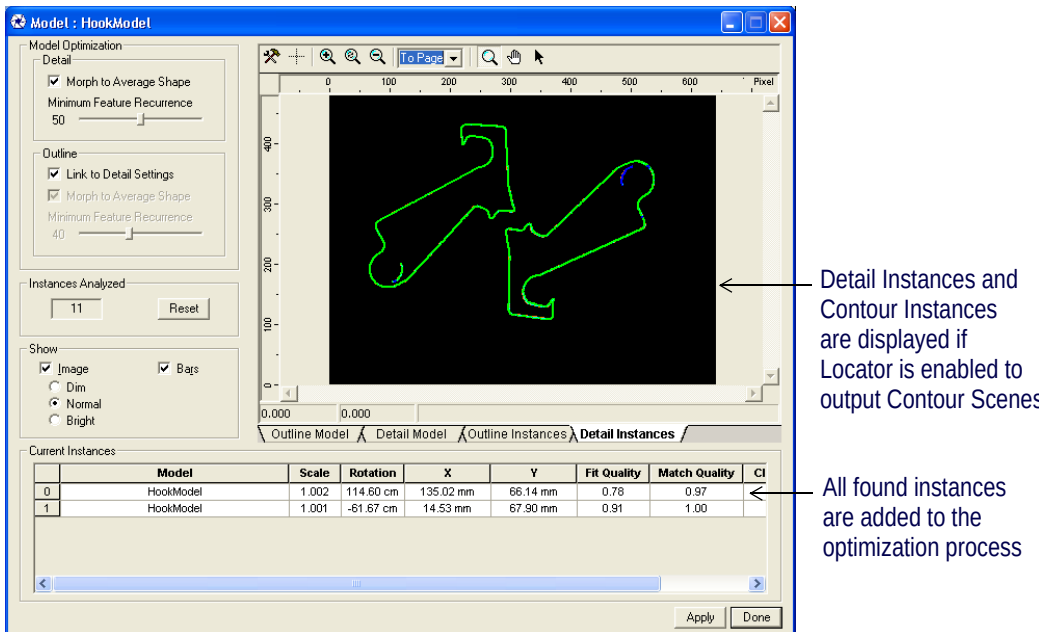


Figure 181 Model Optimizer - Multiple Instances from Single Image

Model Optimizer Display

Use the lower tabs of the Model Optimizer to view the following:

Outline Model: Shows the Outline Level of the Model. This display shows the changes made to the Outline model as new instances are added to the optimization.

Initially, all contours are displayed in blue. After the first execution, Outline level features are displayed in magenta and the deviation between the current features and the initial features are displayed in orange.

Detail Model: Shows the Detail Level of the Model. This display shows the changes made to the Detail model as new instances are added to the optimization. See [Figure 182](#).

Initially, all contours are displayed in blue. After the first execution, Detail level features are displayed in green and the deviation between the current features and the initial features are displayed in orange.

Outline Instances: Shows the Outline level of the instances found in the current input image. To view these instances, the Locator must be enabled to output the [Contour Scene Outline](#) (See **General Panel**).

Detail Instances: Shows the Detail level of the instances found in the current input image. To view these instances, the Locator must be enabled to output the [Contour Scene Detail](#) (See **General Panel**).

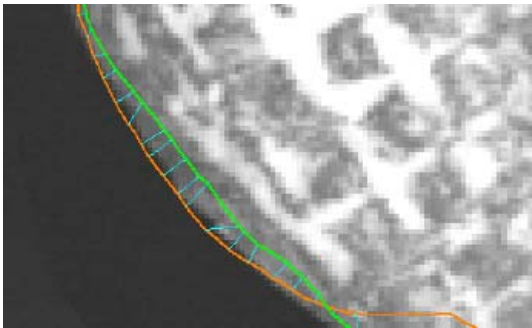


Figure 182 Mode Optimizer Display — Detail of Model Progression during optimization

Search Panel

Use this panel to configure the **Search Constraints** and **Recognition** parameters for the Locator. In the Basic setup mode, only the most essential parameters are shown in the user interface, as illustrated in [Figure 184](#).

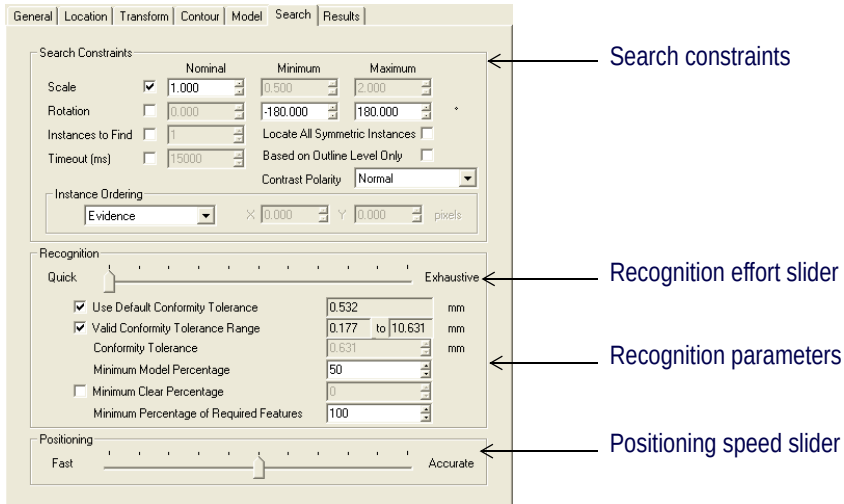


Figure 183 Locator — Search Panel with Advanced Parameters Displayed

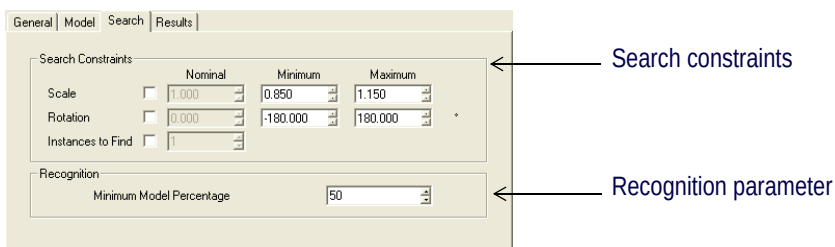


Figure 184 Locator — Search Panel with Advanced Parameters Hidden

Search Constraints

Use **Search Constraints** to restrict the Locator's search process, for example to a specific range of poses or a to specific number of instances.

Scale Factor

Scale constrains the range of scale factors the Locator will search for.

- Enable **Nominal** to search for a specific scale factor. Note that when a nominal value is used, the Locator *does not* compute the actual scale of instances; instead the instances are positioned using the **Nominal** scale value.
- Disable the **Nominal** check box to specify the range of scale the Locator will search for. If you want to search for an instance at a nominal scale factor but need to compute its actual scale, disable the **Nominal** check box and enter a small range such as 0.99 to 1.01.
- A **Nominal** scale factor of **1** is set by default.

Rotation

Rotation constrains the rotation range the Locator will search for.

- Enable **Nominal** to search for objects at a specific angle of rotation. Note that when a nominal value is used, the Locator *does not* compute the actual rotation of instances; instead the instances are positioned using the **Nominal** rotation value.
- Disable the **Nominal** check box to specify the range of rotation the Locator will search for. If you want to search for an instance at a nominal rotation but need to compute its actual rotation, disable the **Nominal** check box and enter a small range such as 89 to 91.
- The maximum rotation range of -180 to 180 is set by default.

Instances to Find

Instances to Find determines the maximum number of instances that can be searched for, regardless of Model type.

- By default, the number of instances that can be found is theoretically unlimited. To optimize search time you should set this value to no more than the expected number of instances.

- When the actual number of instances exceeds the **Instances to Find** value, the Locator stops once it attains the set value.

Timeout

Timeout sets a limit to the Locator's search time.

- When the **Timeout** value is attained, the Locator outputs only the instances found up to the moment of the **Timeout**.

Instance Ordering

Use the **Instance Ordering** combo box to select the order in which object instances are output.

- The default **Evidence** mode outputs instance results in order of their hypothesis strength.
- **Left to Right, Right to Left, Top to Bottom, and Bottom to Top**, output instances according to their position in the image.
- The **Quality** mode outputs results in order of their Match Quality. Instances with the same Match Quality are further ordered according to their Fit Quality.
- **Image Distance** orders instances according to their proximity to a specified point (X,Y coordinates) in the Image coordinate system.
- **World Distance** orders instances according to their proximity to a specified point (X,Y coordinates) in the calibrated World coordinate system.
- **Shading Consistency** orders instance according to their consistency with the shading in the Model.

Locate All Symmetric Instances

If objects in your application are symmetrical or nearly symmetrical in shape, the **Locate All Symmetric Instances** constraint enables the Locator to provide results for all possible variations in position.

- Due to partial occlusion, an object may appear symmetrical although it is not symmetrical in the Model.
- If this constraint is disabled, only the first best quality instance of a symmetric object will be located.

Based on Outline Level Only

When **Based on Outline Level Only** is enabled the Locator searches for object instance using only the Outline Model and Outline Level contours.

- **Based on Outline Level Only** should be enabled only for applications that do not require a high positioning accuracy.

Contrast Polarity

Contrast Polarity indicates the change in polarity between an object and its background. The reference polarity is the polarity in the model image.

- Select **Normal** to search for objects having the same contrast polarity as the model and its background.
- Select **Reverse** if the polarity between object instances and the background is the inverse of the polarities in the Model image. For example dark object instances on a light background using a model created from a light object on a dark background.
- Select **Normal & Reverse** when object instances can present either contrast polarity and when there are no local reversals in polarity along the contours of the object
- Select the **Don't Care** mode only in cases where there are local changes in polarity along an object contour. A typical example is a case in which the background is unevenly colored: striped, checkered, spotted for example.

Recognition

Use recognition parameters to configure the specific search process requirements of the application. Examples of special application requirements include:

- Tolerance to variations in objects contours
- Tolerance to occlusion where the Locator must be able to identify objects that might possibly overlap.
- Very low contrasted and noisy images.

Recognition Effort Slider

The **Recognition** effort slider ranges between **Quick** and **Exhaustive**, with the default center position providing an optimized setting for most applications.

- When changing the recognition effort, test your application to find the optimum speed at which the Locator will still find all necessary objects within the image.
- If recognition effort is too low (**Quick**), the Locator may ignore some instances.
- If recognition effort is too high (**Exhaustive**), your application will run in less than optimal time.
- Recognition speed does not affect positioning accuracy.

Use Default Conformity Tolerance

Disabling Use Default Conformity Tolerance allows you to manually modify the Conformity Tolerance value.

Valid Conformity Tolerance Range

Valid Conformity Tolerance Range defines the upper and lower limits for the Conformity Tolerance.

- This range, which is based on the calibration and resolution of the input Image, cannot be edited.
- Disabling **Valid Conformity Tolerance Range** let you set a Conformity Tolerance that is outside this range.

Conformity Tolerance

Conformity Tolerance defines the maximum allowable local deviation of instance contours from the expected model contours. Its value corresponds to the maximum distance in calibrated units by which a matched contour can deviate from either side of its expected position.

- To manually set Conformity Tolerance you must disable the **Use Default Conformity Tolerance** check box.
- If you set a value lower than the minimum **Valid Conformity Tolerance Range**, this value will be automatically reset to the minimum valid value.
- If you set a value higher than the maximum **Valid Conformity Tolerance Range**, this value will be automatically reset to the maximum valid value.

Minimum Model Percentage

Minimum Model Percentage sets the minimum amount of matched contour required for the Locator to accept a valid object instance.

- Lowering this parameter can increase recognition of occluded instances but can also lead to false recognitions.
- A higher value can help eliminate instances in which objects overlap.

Minimum Clear Percentage

Minimum Clear Percentage sets the minimum percentage of the model bounding box area that must be free of obstacles to consider an object instance as valid.

Enabling **Minimum Clear Percentage** also activates the computation of the **Clear Quality** result for each instance. See Locator [Results Panel](#) section.

Minimum Percentage of Required Features

Minimum Percentage of Required Features determines what proportion of features that are defined as Required must be found for an object instance to be valid.

Positioning Speed Slider

The slider provided for adjusting the positioning accuracy ranges from **Fast** to **Accurate**.

- The more accurate the positioning, the longer the execution time will be.
- The default center position of the slider is the optimized and recommended setting for typical applications.

Results Panel

The Results panel provides results for all the instances found by the Locator, diagnostic messages and access to the graphical display through the Results Viewer.

The screenshot shows the 'Results' tab of the Locator software. It features a table of results, a grid view, display options, elapsed time, and a messages log. Annotations point to specific features:

- Grid of results for all located instances:** Points to the table containing instance data.
- Display options for the Results Viewer:** Points to the 'Results Viewer' section with checkboxes for 'Show Grey-Scale Image', 'Show Contour Scene Outline', 'Show Contour Scene Detail', 'Show Reference Points', 'Show Coordinate System', and 'Show Search Area'.
- Click to open Results Viewer:** Points to the 'Show Results Viewer' button.
- Messages report on the Search process:** Points to the 'Messages' log at the bottom.

	Model	Scale	Rotation	X	Y	Fit Quality
0	HookModel	1.003	-61.36 °	63.35 mm	50.35 mm	0.66
1	HookModel	1.001	118.61 °	100.68 mm	69.44 mm	0.86

Grid

☐ Dynamic Update Coordinate System: World

Results Viewer

☒ Show Grey-Scale Image ☒ Show Reference Points
☐ Show Contour Scene Outline ☒ Show Coordinate System
☐ Show Contour Scene Detail ☒ Show Search Area
☒ Show Instance Scene as : Transformed Model

Elapsed Time (ms)

Learn: 0 Search: 45 Total: 84 [Show Results Viewer](#)

Messages

4513 At least one instance was rejected because insufficient contours were found.

4509 At least one instance was rejected since it did not respect the Scale constraint.

Figure 185 Locator — Results Panel

Grid of Results

Results for each located instance are displayed in the grid. They can be expressed in either the **Tool**, **Image** or **World** coordinate system, according to the setting of the **Coordinate System** parameter. Depending on the

selected coordinate system, some values will not be output because they are non valid or non-significant in the selected frame of reference.

The following results are provided.

- The first column indexes instances in the order in which they were found by the Locator, based on the **Instance Ordering** parameter.
- The **Model** name identifies the Model upon which the Locator based the recognition of the object instance.
- **Scale** is the ratio of the observed object size to its corresponding model size.
- The **Rotation** of the object instance is calculated counterclockwise between the X-axis of the World coordinate system to the X-axis of the Object coordinate system of the instance.
- **X and Y Position** are the **X** and **Y** coordinates of the instance's Object coordinate system, within the selected **Coordinate System**. Position results are expressed in the unit system selected in the General Panel.
- **Fit Quality** score ranges from **0** to **1**, with **1** being the best quality. This value is the normalized average error between the matched model contours and the actual contours detected in the input image.
- **Match Quality** ranges from **0** to **1**, with **1** being the best quality. A value of 1 means that 100% of the model contours were successfully matched to the actual contours detected in the input image.
- **Clear Quality** ranges from 0 to 1 with one being the best quality. A value of one means that the found instance is clear of obstacles.
- **Symmetry of** is the index number of the instance of which the given instance is a symmetry of.
- **Time** shows the time needed to recognize and locate the instance. The time needed to locate the first instance is typically higher because it includes all the preprocessing.

Dynamic Update

By default, **Dynamic Update** is disabled: all the located instances are displayed simultaneously after the Search process is complete. When **Dynamic Update** is enabled, located instances are displayed at the moment they are found.

Note:Enabling this **Dynamic Update** increases the execution time so it is recommended mostly for diagnostic or demonstration purposes.

Coordinate System

The Locator returns results in the selected **Coordinate System**:

- **World coordinate system**: Results are output and displayed in the selected units (General Panel) with respect to the World coordinate system of the input image.
- **Image coordinate system**: Results are output and displayed in pixels. No scale, rotation, quality, time or symmetry results are calculated in this reference system. Position is expressed with respect to the uncalibrated image, (0,0) being the bottom left pixel.
- **Tool coordinate system**: This corresponds to the Locator's **Rectangle** (region of interest). Results are returned in pixel values. No scale, rotation, quality, time or symmetry results are calculated in this reference system. Position is expressed with respect to the **Rectangle**, (0,0) being the center of the bounding box.

Results Viewer

Use **Show Results Viewer** to display Locator results with the following options:.

- **Show Image** displays the current input Image.
- **Show Contour Scene Outline** displays the Outline Level contours.
- **Show Contour Scene Detail** displays the Detail Level contours.

- **Show Instance Scene as** displays the instances found by the Locator. The instance selected (highlighted) in the grid of results appears in red, all other instances appear in blue.
- Select one of two modes for displaying found instances: **Transformed Model** or **Matched Model**. This will not affect the display mode you may have set for the Instance Scene (Locator output).
- **Transformed Model** displays the complete Model transposed according to the pose computed by the Locator. This lets you visualize the position of instances even when the objects are partially hidden from view or when some of their contours have not been found.
- **Matched Model** displays only the sections of the Model contours that can be matched to the actual contours detected in the input image. This allows you to visualize which portions or sections of the instance have not been detected.
- **Show Reference Points** displays reference point markers.
- **Show Coordinate System** displays the Object coordinate system markers.
- **Show Rectangle** displays the **Rectangle** bounding box.

Elapsed Time

- **Learn** time is the time required for the Learn during the execution of the Locator. The time is also displayed for a learn initiated by the **Learn** command.
- **Search** time is the total time used by the search process.
- **Total** time is the total time elapsed for the execution of the Locator process.

Messages

The **Message** box displays messages that provide information on the search process. The complete list of messages is provided in of the [Understanding Locator Concepts](#) section.

Acquisition Device

The Acquisition Device acquires Images from a frame grabber or from a database of Images. Integrated image calibration methods can compensate for pixel aspect ratio, perspective and lens distortion.

Acquisition Device

[Understanding Acquisition Device Concepts](#)

[Image Acquisition](#)

[Left-Handed vs Right-Handed Coordinate Systems](#)

[Image Calibration](#)

[Acquisition Device Output](#)

[Using the Acquisition Device](#)

[General Panel](#)

[Configuration Panel](#)

[Input Sub-Panel](#)

[Calibration Sub-Panel](#)

[Output Sub-Panel](#)

[Results Panel](#)

Understanding Acquisition Device Concepts

This section describes Acquisition Device concepts under three sections that correspond to the main functions carried out by this tool:

- Image Acquisition
- Image Calibration
- Acquisition Device Output

Image Acquisition

The Acquisition Device provides three possible input sources for acquiring images needed to run a HexSight application. These three sources are described below.

File Emulation

The **File Emulation** mode simulates a live acquisition by using a database of images, called an emulation database.

Emulation files are usually created for two reasons:

- a) Storing images used to create object Models. Such images should be taken in almost ideal lighting and pose conditions to ensure creation of accurate, well-defined Models.
- b) Creating a database to test and set up an application. Environmental conditions should be the same as those used for real time operation. This database should include various poses and rotations for a given object.

Database Selection

The **Database Selection** mode uses an image already in the runtime database as the input source for the Acquisition Device. This mode is provided for advanced applications in which images are provided by an external source, instead of a supported frame grabber, for example. These images can then be calibrated through the Acquisition Device and output to other HexSight tools.

Frame Grabbers

The **Frame Grabber** mode is the live acquisition mode that allows input from one or more frame grabbers. The **Generic Frame Grabber** setting uses the first frame grabber detected at runtime by the Acquisition Device as the input source. The following frame grabbers are currently supported.

- [Data Translation DT-3152](#)
- [Data Translation DT-3155](#)
- [Data Translation DT-3157](#)

- [Imagenation PX610A](#)
- [Imagenation PXC200](#)
- [Dalsa IC-ASYNC](#)
- [Dalsa PC-DIG](#)
- [Dalsa PCVision](#)
- [Matrox Meteor-II/Standard](#)
- [Matrox Meteor-II/Multi-Channel](#)
- [Matrox Meteor-II/Digital](#)
- [Matrox Orion](#)
- [Opteon DepictMaster 4x1](#)
- [DirectShow Device](#) (FireWire, IEEE 1394, USB cameras)
- [DCAM Device](#) (IEEE 1394 cameras)
- [GigE Vision Cameras](#)
- [Sony Intelligent Cameras](#)
- [Gocator Sensor](#)

Note: Additional frame grabbers may be added to this list. Contact HexSight technical support for an updated list of supported frame grabbers.

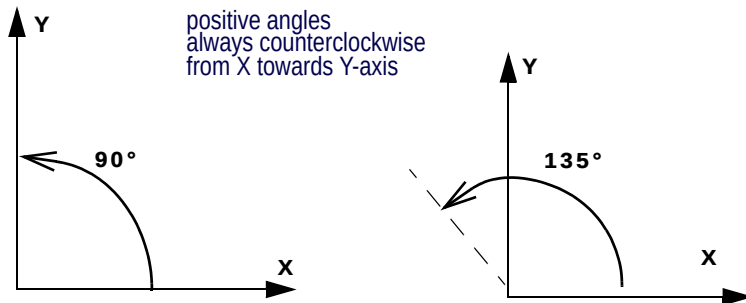
HexSight can directly interface with the standard device drivers for the supported frame grabbers. The Acquisition Device interface provides access to configuration parameters for the selected frame grabber. See the corresponding frame grabber section in ***Using the Acquisition Device*** for details on configuring the parameters.

Left-Handed vs Right-Handed Coordinate Systems

HexSight supports both left- and right-handed coordinate systems, for both the calibrated (World) coordinate system and the Image coordinate system. Other HexSight coordinate systems, Object and Tool, are always right-handed.

- A left-handed (top-left origin) as well as a right-handed (bottom-left origin) image can be calibrated into either a left-handed or a right-handed coordinate system.
- Since models are created on a calibrated image, the type of coordinate system for a given model must be of the same type as the calibrated (World) coordinate system in which instances of the modeled object are being sought.

RIGHT-HANDED COORDINATE SYSTEMS



LEFT-HANDED COORDINATE SYSTEMS

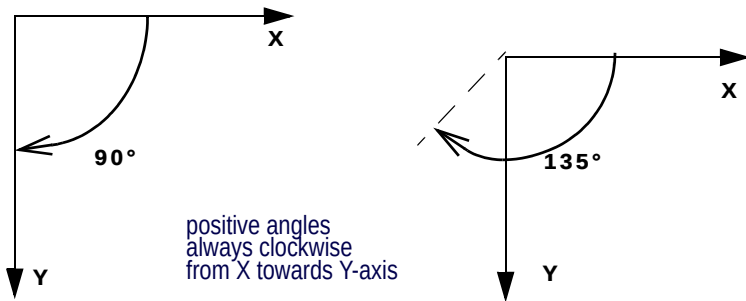


Figure 186 Right-Handed vs. Left-Handed Coordinate Systems

Origin

By default, the origin of a right-handed coordinate system is at the bottom-left of an image whereas the origin of a left-handed coordinate system (top-left origin) as well as a right-handed (bottom-left origin) image can be calibrated into either a left-handed or a right-handed coordinate system.

Rotation

Angles are always measured from the X-axis toward the Y-axis in either clockwise or counterclockwise direction, depending on the type of coordinate system type.

- With a left-handed calibrated coordinate system, the HexSight tools return positive angle values for clockwise rotations, whereas with a right-handed coordinate system, HexSight tools return positive angle values for counterclockwise rotations.

Image Coordinate System

The Image coordinate system type is selected at image acquisition. Coordinates in this system are expressed in terms of pixels.

World Coordinate System

World coordinate system is also called the calibrated coordinate system. Its type is set during the calibration process. Coordinates in this system are expressed in terms of calibrated units.

Image Calibration

Calibration ensures the accuracy of a vision application. It permits correction of image errors and complete application portability from site to site. The following three types of image deformation must be corrected to ensure the robustness of the object detection tools.

- **Non-Square Pixels:** Because pixels are not square (as in most cameras), distortion occurs when any object is rotated in the field of view. This distortion can generate a significant lack of precision. By calibrating, you will provide rotational invariance for non-square pixels. The following HexSight calibration methods can be used to correct for this type of error: XY Scale, Perspective, Distortion LUT, and Distortion Model.
- **Perspective Distortion:** Cameras are rarely perfectly perpendicular with respect to the work surface. They may even be purposely set at an angle. Non-orthogonality generates perspective distortions. Perspective distortion can be corrected with the

following HexSight calibration methods: Perspective, Distortion LUT, and Distortion Model.

- **Camera Lens Distortion:** Camera lenses induce some amount of radial distortion. This distortion is generally stronger for lenses having a short focal distance. Lens distortion can be corrected by the following HexSight calibration methods: Distortion LUT, and Distortion Model.

Calibration Parameters

Calibration parameters can be stored in files that can be loaded by various applications sharing the same optical setup. Images in emulation database files are stored with their original calibration. The Acquisition Device can be set up to use this original calibration when using file emulation as the input source.

Calibration Units

The calibration can be configured to use physical units to express positions.

- The HexSight environment provides automatic conversion between each of the supported unit systems.
- The generic length unit, *Calibration Units*, can be used to calibrate the system when actual physical dimensions are not required, or when the objects to inspect possess a strong 3D aspect (measurements on sections located at relatively large Z distance from the calibration plane might yield to inconsistent values).

Calibrated Coordinate System

The World coordinate system can be calibrated with either a right- or left handed coordinate system, regardless of the coordinate system type that has been set for the input image.

HexSight Calibration Methods

HexSight provides the following five calibration methods.

XY Scale Calibration

This calibration procedure is easy to set up since the calibration target can be any object of known dimensions. Less precise than perspective and distortion calibrations, XY Scale calibration is used in cases where it is not physically possible to place a dot target. It does not compensate for either lens distortion or perspective distortion.

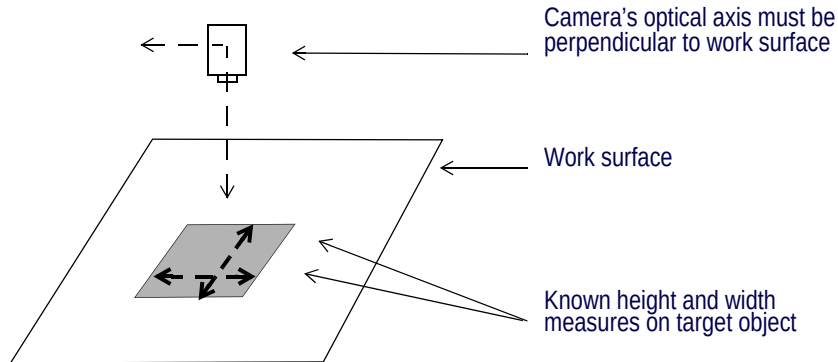


Figure 187 XY Scale Calibration— Target and Camera Position

- The user must provide both a known height and width measure for the object used as a target.
- In absence of a target, the XY Scale method can also be used to manually enter the height and width of a pixel in calibrated World values if they are already known.
- For this type of calibration, the camera's optical axis must be perfectly PERPENDICULAR to the work surface.

Perspective Calibration

As with XY Scale calibration, this type of calibration can be carried out on a known object. This mode is less precise than a distortion calibration. Therefore it is used in cases where it is not physically possible to place a dot target. It is however more precise than an XY Scale calibration since it compensates for perspective distortion. It does not compensate for lens distortion.

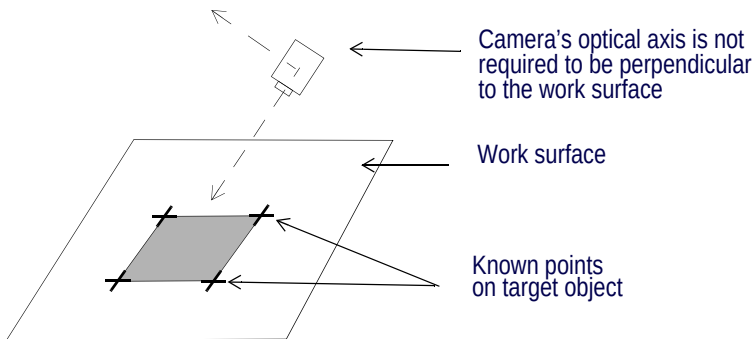


Figure 188 Perspective Calibration — Target and Camera Position

- The target object must provide four non-collinear points, typically corners.
- The user must know and provide the calibrated World coordinates of the four points. These points can be automatically detected in the Image with the integrated corner detector or positioned manually by entering the appropriate position values.

Distortion LUT Calibration

This calibration method requires the use of a target constituted of a grid of dots, referred to as a dot target. A lookup table is created from all the dot positions on the target. The position of any given point in the image is interpolated from the position of the four closest target dots.

- This calibration mode is preferable to the Distortion Model mode in cases where lens distortion is very strong.
- This calibration mode also requires a VERY precise, "perfect" target.

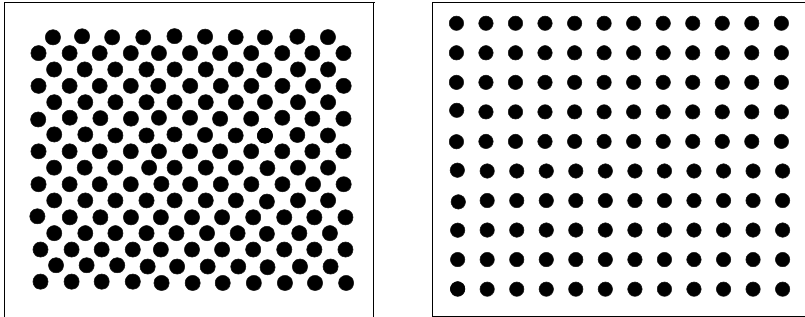


Figure 189 Dot Targets Used for Distortion Calibration Methods

Note: See *Using the Acquisition Device* for more information on dot targets.

Distortion Model Calibration

This is the preferred and recommended calibration mode. As with Distortion LUT, the Distortion Model method requires the use of a dot target. This calibration method uses a mathematical calculation to apply a best fit approach for locating points in the current image.

- This method tolerates a slightly less precise target than the Distortion LUT method since it averages out errors, and is preferred in cases where lens distortion is not very strong.

Programmatic (Automatic) Calibration

HexSight provides a way to programmatically calibrate applications by providing a custom set of calibration target points. With this method, called **Automatic Calibration**, the application developer must establish how the calibration data (coordinates of points on the work surface) will be acquired and how they will be supplied to the Acquisition Device process.

Acquisition Device Output

The Image output by the Acquisition Device is typically used as input by subsequent HexSight processes such as the Locator and inspection tools.

Using the Acquisition Device

General Panel

The **General** panel of the Acquisition Device contains only the process name. You can modify the name in this panel to replace the default name.

- Typically, the default **Process Name** appears as `HSAcquisitionDevice0`.
- Rename the tool by typing in a new name, such as *Acquisition*.

IMPORTANT: A HEXSIGHT APPLICATION CAN CONTAIN ONLY A SINGLE ACQUISITION DEVICE. IF YOU NEED TO ACQUIRE IMAGES FROM MULTIPLE SOURCES, CREATE A CONFIGURATION FOR EACH SOURCE.

Configuration Panel

An Acquisition Device Configuration stores all the parameters set up within the Configuration panel: acquisition, calibration and output.

IMPORTANT: AN ACQUISITION DEVICE CONFIGURATION IS DIFFERENT FROM AN APPLICATION CONFIGURATION, WHICH CONTAINS THE CONFIGURED TOOLS THAT CONSTITUTE A HEXSIGHT APPLICATION. ACQUISITION DEVICE CONFIGURATIONS ARE SAVED AS PART OF THE PROPERTIES OF THE ACQUISITION DEVICE.

A default Configuration is attributed whenever you instantiate a new Acquisition Device process. Typically each frame grabber or camera used in an application will have its own configuration.

Using Multiple Configurations

There can be more than one active configuration; enabling multiple active configurations permits a single *Execute* command to perform an acquisition on each active configuration. Programmatically the *ConfigurationEnabled* property sets the active Configurations. In the interface, you can activate a configuration by enabling its corresponding check box.

Editing the List of Configurations

Set up additional Configurations and modify the list of Configurations using the provided set of command buttons:

Add Configuration

Remove Configuration

Set as Default

Rename Configuration

Copy Configuration

Paste Configuration

The Default Configuration

When an Acquisition Device has multiple Configurations, the execution is always carried out using the default Configuration.

- To set the default configuration, select an item in the list of available Configurations and click **Set As Default**.
- For instance, an Acquisition Device can have one Configuration named *File Emulation*, set up in the file emulation mode, and another named *Camera*, set up to use a frame grabber.
 - a) To test the application with the stored images: Select the *File Emulation Configuration* in the list and click **Set As Default**. Click **Execute Sequence** in the Process Manager to simulate the acquisition.
 - b) To use the frame grabber as the input source, set the Camera configuration as the default and click **Execute Sequence** in the Process Manager.

Using Multiple Configurations

You can set up more than one Configuration for a given Acquisition Device. This can be used to configure multi-camera acquisitions or to set up two or more sets or calibration parameters for a single camera.

- For example, multiple Configurations can be used to calibrate a mobile camera (on a robot arm for instance) that takes images of the same object at different distances. You can then toggle between the two or more configurations to activate the correct calibration for a given position.
- Multiple Configurations can also be used in multiple camera setups to specify channel from which the image is acquired. Instead of programmatically changing the input channel after each acquisition, you can set up two Configurations, one for each channel, and toggle between the two. This is particularly useful when the two cameras are calibrated differently (as it is usually the case) with each Configuration holding the calibration of its attributed camera.

Using Multiple Camera Configurations Programmatically

There are two ways of using the Acquisition Device programmatically: sequential acquisitions and pipelined acquisitions. In the sequential mode, an acquisition request is sent to the frame grabber and the Acquisition Device waits until the image is acquired before returning the control to the application.

The following example shows how to toggle between the two Configurations in the sequential acquisition mode.

```
Dim Acquisition as HSAcquisitionDevice
Set Acquisition = _
    ApplicationControl.ProcessManager.Process("Acquisition")

Do
    Acquisition.ConfigurationDefault = "Camera0"
    Acquisition.Execute
    'Process Camera0 image
    ...
    Acquisition.ConfigurationDefault = "Camera1"
    Acquisition.Execute
    ' Process Camera1 image
Loop While Not ExitLoop
```


In the pipelined acquisition mode, an acquisition request is sent to the frame grabber and the Acquisition Device returns the control to the application immediately. When the application is ready to process the image, it instructs the Acquisition Device to wait for the completion of the request. The following example shows how to use the pipelined acquisition mode with two Configurations.

```
Dim Acquisition as HSAcquisitionDevice
Set Acquisition = _
    ApplicationControl.ProcessManager.Process("Acquisition")

'Place a first request in the pipeline
Acquisition.Start "Camera0"
Acquisition.Start "Camera1"

Do
    'Place a new request in the pipeline
    Acquisition.Start "Camera0"
    Acquisition.Start "Camera1"
    'Wait for Camera0 image
    Acquisition.Wait
    'Process Camera0 image
    ...
    'Wait for Camera1 image
    Acquisition.Wait
    'Process Camera1 image
    ...
Loop While Not ExitLoop

'Flush the acquisition pipeline
Acquisition.Wait
Acquisition.Wait
```

The same pipelined acquisition mode can be used with a single Configuration. This is useful for time-critical applications because a new image can be acquired while another one is processed, reducing the total cycle time.

Input Sub-Panel

Use this panel to select the input device (or mode) that will acquire the images for your application and configure the parameters for the selected device.

In running mode, you will typically use live images input by a camera through a configured frame grabber. However, to set up and configure a new HexSight application, you will most often rely on a database of images to simulate real time acquisitions.

Live images are typically provided by a frame grabber. The Input panel always offers the **Generic Frame Grabber** input mode. Additional frame grabbers can be added and removed using the **Add Frame Grabber** and **Remove Frame** Grabber buttons.

Database Selection

Database Selection lets you use an image already present in the application database as the input source for the current Configuration. Use the selection icon to select the **Selection Image** and its **View** from the application's current database.

File Emulation

File Emulation input uses a database of images, referred to here as an emulation database file, to simulate real time image acquisition. Image database files are particularly useful for: creating models, setting up and testing a new application or analyzing/verifying performance with Images from a real time operation.

To familiarize yourself with the Acquisition Device without installing a complete camera/frame grabber setup, use one of the image databases provided with the Examples on your HexSight CD.

Using the File Emulation Database

An emulation database file is typically constituted by grabbing images of the same objects with various poses or orientations in the workspace. The emulation file illustrated in [Figure 190](#) for example, consists of a single object with various poses. Lighting conditions have been set up to obtain well-contrasted and strongly detailed images.

Enable/Disable check boxes allow you to select which images in the database you want to use for the current application.

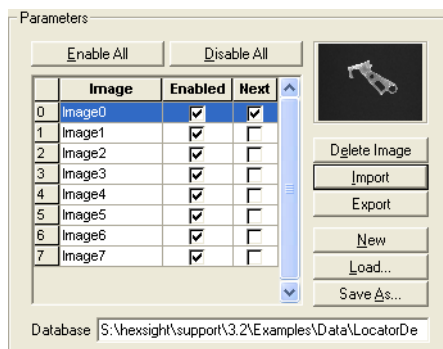


Figure 190 Emulation Database

Use the command buttons to add or remove images from the emulation database:

- **Delete Image** removes the image currently selected in the emulation database. This removes it from the runtime
- **Import** allows you to import selected images into the current emulation database from Bitmap (*.bmp) files, HexSight Image files (*.hig) or selected images from a HexSight Database file (*.hdb). Note that large databases can be long to load and use a significant amount of memory.
- **Export** allows you to export the currently selected image as a Bitmap image or as a HexSight Image (*.hig) file.
- **New** clears the current database.
- **Load...** clears the existing database (*.hdb) for the current Configuration and replaces it by the newly selected database file.
- **Save As...** allows you to save the current database to a database (*.hdb) file.

Selecting Specific Images in the Database

To test your application with specific cases, you can enable or disable images in the emulation database.

- To enable all the images in the database, click **Enable All**.
- To disable all the images in the database, click **Disable All**.
- To enable/disable specific images, click the specific images check box in the **Enabled** column.

The check box in the **Next** column is exclusive and identifies which image will be output the next time the Acquisition Device is executed. For example, to test an application with only a specific image in a given database:

- Click **Disable All**.
- Click the **Enabled** and **Next** check boxes for this image.

The thumbnail image provides a preview of the selected image.

Building an Emulation Database File

An emulation database file is generally constituted by grabbing images of the same objects with various poses or orientations in the workspace. The emulation file illustrated in [Figure 190](#) for example, consists of a single object with various poses. Lighting conditions have been set up to obtain well-contrasted and strongly detailed images.

- First select an existing Configuration in the Configuration list or create a new one and select **File Emulation** as the input source for this Configuration.
- From the list of input sources, select the required frame grabber or use the Generic Frame Grabber interface (see sections on the frame grabber input for more details).
- To view the live image output by the camera, click **Show Real-Time Display**.
- Click **Append to Emulation Database**. In the Emulation Image Name dialog, type the name for the new image. You can also build an Emulation database by importing images.

Generic Frame Grabber

When **Generic Frame Grabber** is selected, the Acquisition Device uses the first detected frame grabber as the input source for the selected Configuration. Since the Configuration parameters vary widely from one frame grabber to another, this input source does not provide any configurable parameters. Instead, the default parameters for the type of frame grabber detected are applied and the first channel is used as the input source.

- Although not very useful for advanced applications, the Generic Frame Grabber interface can be used to build applications compatible with various acquisition setups.
- The Generic Frame Grabber interface only works with the frame grabber types supported by the Acquisition Device.

Adding a Detected Frame Grabber

To configure parameters for a frame grabber you must first add a specific frame grabber as the input source for the selected configuration.

- Use the **Add Frame Grabber** button to open the frame grabber dialog shown in [Figure 191](#). The **Detected Frame Grabbers** frame lists the frame grabbers that are installed in the system.
- If one or more supported frame grabbers are installed in the system, they will appear in the list of detected frame grabbers.
- To use one of the listed frame grabbers, select its name in the **Model** list, then **OK** to quit the window. This will add the frame grabber to the list of input devices available to each Configuration of the Acquisition Device.

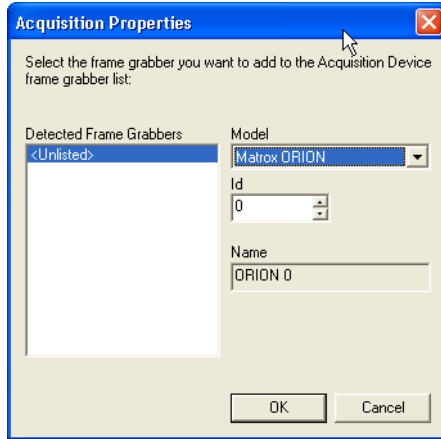


Figure 191 Frame Grabber Selection

Adding a Virtual Frame Grabber

A virtual frame grabber is one that is not physically installed on the system but is added to the list of available input devices.

This virtual frame grabber is useful as a container in which you can store the parameters for future use on a system where a grabber of the same type (as that as of the virtual frame grabber) is installed. When you run the application with an actually installed frame grabber of this type, the parameters set for the virtual frame grabber are applied and images can be acquired.

- Click **Add Frame Grabber** to open the frame grabber selection window illustrated in Figure 191.
- Select <Unlisted> in the list of detected frame grabbers.
- From the **Model** combo box, select the detected frame grabber type.
- Enter the ID used to identify the frame grabber; 0 is attributed to the first frame grabber detected at boot time, 1 is attributed to the second board of the same type, and so forth.

- Click **OK** to add this device to the list and return to the Acquisition Device interface. Setup parameters for the selected frame grabber now appear in the Input sub-panel.

Removing a Frame Grabber

Select a frame grabber from the list then click **Remove Frame Grabber**.

Frame Grabber Command Buttons

The four following command buttons appear in the Input Sub-Panel when you select a frame grabber as input source.

- **Append Image to Emulation Database** allows you to acquire a new image and add it to the emulation database contained in the currently selected Configuration. To browse this emulation database, select File Emulation in the list of available input devices without selecting another Configuration.
- **Show Real-Time Display** opens a display window showing live images output by the camera. You can use this display to adjust the optical setup for the application. You can change the zoom factor of the display through the context menu (right-click on the display).
- **Load Parameters** lets you load a previously saved frame grabber configuration file (*.fgp) and apply it to the currently selected frame grabber, either generic or specified.
- **Save Parameters** lets you save the current configurations for the frame grabber for future use (*.fgp file type).

Dalsa IC-ASYNC

The IC-ASYNC is a versatile progressive scan frame grabber. It provides up to four (4) fully independent inputs. This board is recommended for advanced applications requiring features such as synchronized acquisitions, asynchronous resets, strobe lighting, etc.

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Dalsa Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the IC-ASYNC are available in the **Parameters** frame ([Figure 192](#)).

Note: If you are using more than one Dalsa grabber, see the **Multiple Dalsa Frame Grabbers Application** section.

Port	Grab	Output	Trigger	Timeout
0	☒	☒	☐	5000 ms
1	☐	☐	☐	
2	☐	☐	☐	
3	☐	☐	☐	

Figure 192 IC-ASYNC Setup

- The IC-ASYNC is configured through **Configuration Files** built with the ITI Camera Configurator application provided with the frame grabber. The Acquisition Device only supports the binary cnf file format. By default, the configuration file parameter is **AUTO**. The *AUTO* parameter is not a configuration file but rather a keyword that instructs the IC-ASYNC to use its default configuration. Unless a NTSC interlaced camera is used, this default configuration will not usually work. If you type *AUTO* in the **Configuration File** text box instead of a valid cnf file name, you will restore the default configuration. *AUTO is case sensitive and must be in upper case.*
- When you enter a file name in the **Configuration File** text box or select a file using the **Load** button, the new frame grabber configuration is not loaded immediately but rather at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different cnf files because the IC-ASYNC will pause for a few seconds between each acquisition request.

- **Grab** specifies which channels should be acquired when the Acquisition Device is executed with the selected configuration. When multiple channels must be acquired simultaneously, enable their respective **Grab** check boxes. See the following section for more details on multi-channel acquisitions with the IC-ASYNC.
- **Output** specifies which channel provides the image that is created in the runtime database when the Acquisition Device is executed with the selected Configuration. Unlike **Grab**, **Output** is an exclusive parameter, which means that the image from only one channel can be output at a time. See *Using the IC-ASYNC in a Multi-Camera Application*.
- **Trigger** is used to specify that before acquiring the corresponding channel, the IC-ASYNC should wait for the trigger event configured in the cnf file.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.

Using the IC-ASYNC in a Multi-Camera Application

For an IC-ASYNC multi-camera application in which acquisitions must be simultaneous, you should preferably create an Acquisition Device Configuration for each channel that must be grabbed. This method is recommended rather than programmatically changing the **Output** parameter with a single Acquisition Device Configuration because a specific calibration can be set up for each channel.

Creating an IC-ASYNC Two-Channel Application

To create a two-channel application, define two Acquisition Device Configurations:

- Set the first one, "Channel0", with **Grab** enabled for the two channels and **Output** set for channel 0.
- Set the second Configuration, "Channel1", with **Grab** disabled for all channels and **Output** set for channel 1.

- When the Acquisition Device is executed with the first Configuration, the two channels are grabbed and kept in memory and the image from Channel 0 is output to the runtime database.
- When the Acquisition Device is executed with the second Configuration, no image is grabbed and the image from Channel 1 is output to the runtime database.

The following example shows how to use these two Configurations programmatically in an acquisition pipeline.

```
Dim Acquisition as HSAcquisitionDevice
Set Acquisition = _
    ApplicationControl.ProcessManager.Process("Acquisition")

' Place a first request in the pipeline
Acquisition.Start "Channel0"
Acquisition.Start "Channel1"

Do
    ' Place a new request in the pipeline
    Acquisition.Start "Channel0"
    Acquisition.Start "Channel1"
    ' Wait for Camera0 image
    Acquisition.Wait
    ' Process Camera0 image
    ...
    ' Wait for Camera1 image
    Acquisition.Wait
    ' Process Camera1 image
    ...
Loop While Not ExitLoop

' Flush the acquisition pipeline
Acquisition.Wait
Acquisition.Wait
```

Dalsa PC-DIG

The PC-DIG is a progressive scan frame grabber. It provides inputs for either a single RGB digital camera or up to four 8 bits monochrome digital cameras. When connecting multiple 8 bits cameras to the Meteor-II/Digital, the grabber must be configured to generate the timing signals in order to ensure synchronization between data channels.

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Dalsa Frame Grabber*** in the ***Installing HexSight*** chapter.

Most configurable parameters for the PC-DIG are available in the **Parameters** frame ([Figure 194](#)).

Note: If you are using more than one Dalsa grabber, see the ***Multiple Dalsa Frame Grabbers Application*** section.

- The PC-DIG is configured through **Configuration Files** built with the ITI Camera Configurator application provided with the frame grabber. The Acquisition Device only supports the binary cnf file format. By default, no configuration file is specified; this instructs the PC-DIG to use its default configuration. However, the default configuration will not usually work.

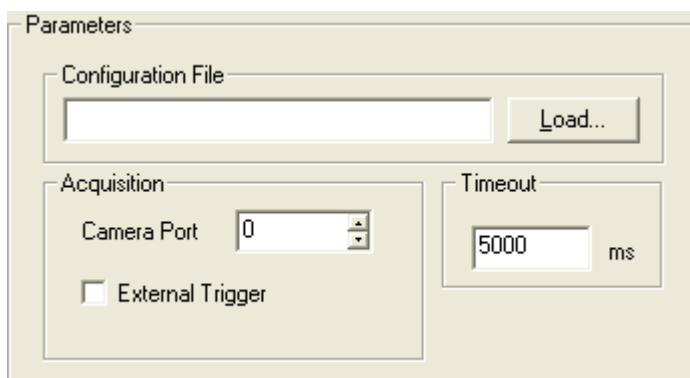


Figure 193 PC-DIG Setup

- When you enter a file name manually in the **Configuration File** text box or select a file using the **Load** button, the new configuration is not loaded immediately. The frame grabber configuration file is loaded and applied at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different cnf files because the PC-DIG will pause for a few seconds between each acquisition request.
- **Camera Port** sets the input channel from which the image is acquired. The PC-DIG implementation actually supports only one (1) 8-bit channel.
- **External Trigger** is used to specify that before acquiring the image, the PC-DIG should wait for the trigger event configured in the cnf file.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.

Dalsa PCVision

The PCVision is a progressive scan frame grabber that provides inputs for four (4) cameras. Since it possesses only one acquisition circuit, each of these four channels must be acquired sequentially. The PCVision supports features such as asynchronous resets, strobe lighting and gen-locking.

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Dalsa Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the PCVision are available in the **Parameters** frame ([Figure 194](#)).

Note: If you are using more than one Dalsa grabber, see the ***Multiple Dalsa Frame Grabbers Application*** section.

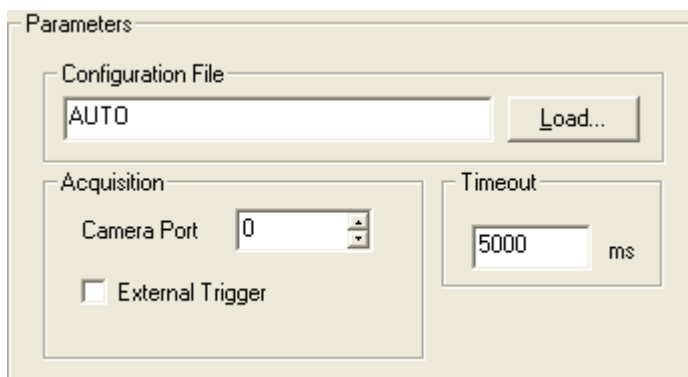


Figure 194 PCVision Setup

- The PCVision is configured through **Configuration Files** built with the ITI Camera Configurator application provided with the frame grabber. The Acquisition Device only supports the binary cnf file format. By default, the configuration file parameter is **AUTO**. The *AUTO* parameter is not a configuration file but rather a keyword that instructs the PCVision to use its default configuration. Unless a NTSC interlaced camera is used, this default configuration will not usually work. If you type *AUTO* in the **Configuration File** text box instead of a valid cnf file name, you will restore the default configuration. *AUTO is case sensitive and must be in upper case.*
- When you enter a file name manually in the **Configuration File** text box or select a file using the **Load** button, the new configuration is not loaded immediately. The frame grabber configuration file is loaded and applied at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different cnf files because the PCVision will pause for a few seconds between each acquisition request.
- **Camera Port** sets the input channel from which the image is acquired. The PCVision supports four (4) input channels but has only one video acquisition circuit. The channels cannot therefore be acquired simultaneously.

- **External Trigger** is used to specify that before acquiring the image, the PCVision should wait for the trigger event configured in the cnf file.
- **Timeout** expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.

Multiple Dalsa Frame Grabbers Application

If you are using more than one Dalsa frame grabber, of same or different types (one IC-ASYN and one PCVision for example) the following issues should be considered:

- You *must* include the parameters for *all* of the installed grabbers in the cnf files used.
- To make sure the application does not pause when processing acquisition requests for each grabber, you must use the *same* cnf file for all of the Configurations used by the Acquisition Device in an acquisition pipeline.
- In summary, you define a single cnf file that will include the configuration of all the boards and cameras installed and use this file in all of the Acquisition Device Configurations used to define an acquisition pipeline. The Configurations that use this cnf file can then be used to control the acquisition pipeline without a pause between each request.

Data Translation DT-3152

The DT-3152 is a progressive scan frame grabber that provides inputs for four (4) cameras. Since it possesses only one acquisition circuit, each of these four channels must be acquired sequentially. Furthermore, compatibility problems with some cameras prevent using features such as asynchronous reset on external triggers and synchronization between channels. When used with HexSight, the DT-3152 is best suited as a single channel progressive scan frame grabber.

Refer to the documentation provided with your frame grabber for hardware and software installation. HexSight uses the standard driver provided with the DT-3152, so no special installation procedure is required. See ***Installing a Data Translation Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the DT-3152 are available in the **Parameters** frame ([Figure 195](#)). The parameters are grouped into 3 tab-selected sub-panels: **Acquisition**, **Clock** and **Field**.

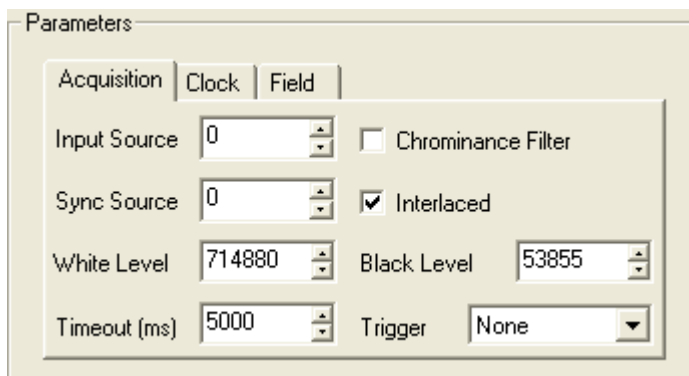


Figure 195 DT-3152 Acquisition Sub-Panel

Acquisition Sub-Panel

The Acquisition sub-panel (See [Figure 195](#)) configures the acquisition request.

- **Input Source** sets the input channel from which the image is acquired. Only one channel at a time can be acquired with the DT-3152 when the Acquisition Device is executed.
- **Sync Source** sets the input channel that is used to synchronize the frame grabber digitization circuit with the incoming image. Usually **Synchronization Source** is the same as **Input Source**. However, when external gen-locking is used to synchronize cameras, **Synchronization Source** should always be set to the same channel to avoid desynchronization when changing the **Input Source**.

- **Chrominance Filter** should be enabled only if you are using a color camera. Otherwise, leave this parameter disabled since it has an undesirable smoothing effect on the image.
- **Interlaced** specifies if the incoming video signal is composed of two fields (interlaced) or a single field (non-interlaced). **Interlaced** should be enabled when using a progressive scan camera as the video source. It should be disabled otherwise.
- **White Level**, expressed in microvolts, sets the digitizer's white level voltage reference. Use this parameter to adjust the frame grabber's sensitivity with non-standard video signals.
- **Black Level**, expressed in microvolts, sets the digitizer's black level voltage reference. Use this parameter to adjust the frame grabber's sensitivity with non-standard video signals.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.
- **Trigger** sets whether image acquisition is continuous or synchronized to an external signal event. Select **None** with the combo box to enable continuous acquisitions. Select **High to Low** or **Low to High** to respectively synchronize acquisitions with high to low or low to high transitions of an external trigger signal. Due to compatibility problems with most cameras, the external trigger mode cannot be used to generate asynchronous resets. Instead, following a trigger event, the grabber will acquire the next image output by the camera.

Clock Sub-Panel

The Clock sub-panel (See [Figure 196](#)) configures the frame grabber input clocking source and related parameters.

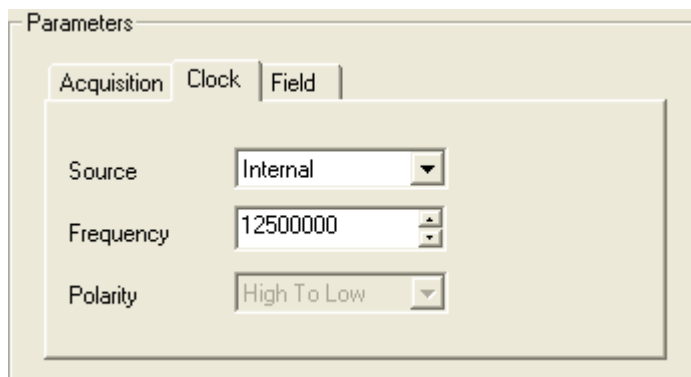


Figure 196 DT-3152 Clock Sub-Panel

- **Source** indicates whether the input clocking should be generated by the frame grabber (**Internal**) or provided by an outside source (**External**).
- **Frequency**, expressed in hertz, sets the internal input (pixel) clock frequency.
- **Polarity** specifies the polarity of the external input clocking source. If set to **High To Low**, synchronization is performed on high to low transitions. If set to **Low To High**, synchronization is performed on low to high transitions.

Field Sub-Panel

The Field sub-panel (See [Figure 197](#)) configures the spatial resolution of input images.

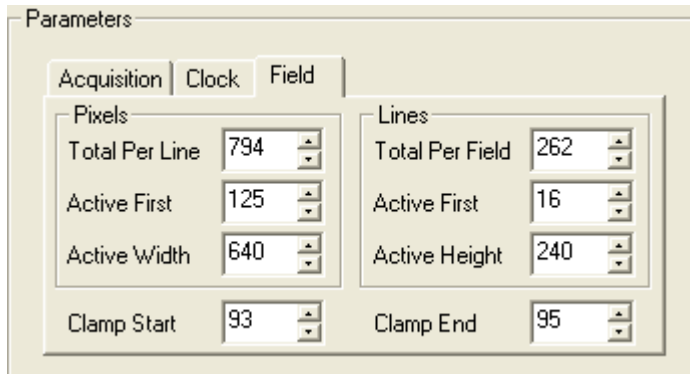


Figure 197 DT-3152 Field Sub-Panel

- **Pixels** parameters are used to configure: the number of pixels per line in the active video area (**Total Per Line**), the beginning of the active video signal on the line, as a pixel value offset from the beginning of the horizontal sync (**Active First**), and the total number of pixels in a single horizontal line of video, where a horizontal line is defined as the area between two consecutive horizontal sync signals (**Active Width**).
- **Lines** parameters are used to configure: the total number of lines in a single field of video, where a field is defined as the area between two consecutive vertical sync signals (**Total Per Field**), the beginning of the active video signal within the field, as a line offset from the beginning of the vertical sync (**Active First**), and the number of lines per field (or non-interlaced frame) in the active video area (**Active Height**).
- **Clamp** parameters are used to configure the beginning of an area where the frame grabber may clamp the video signal, as a pixel value offset from the beginning of the horizontal sync (**Clamp Start**), and the end of an area where the frame grabber may clamp the video signal, as a pixel value offset from the beginning of the horizontal sync (**Clamp End**).

Data Translation DT-3155

The DT-3155 is an RS-170 monochrome interlaced scan frame grabber. It provides inputs for four (4) cameras. However, since it possesses only one acquisition circuit, each of the four channels must be acquired sequentially.

Refer to the documentation provided with your frame grabber for hardware and software installation. HexSight uses the standard driver provided with the DT-3155, so no special installation procedure is required. See ***Installing a Data Translation Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the DT-3155 are available in the **Parameters** frame ([Figure 198](#)).

Figure 198 DT-3155 Setup

- **Input Source** sets the input channel from which the image is acquired. Only one channel at a time can be acquired with the DT-3155 when the Acquisition Device is executed.
- **Sync Source** sets the input channel that is used to synchronize the frame grabber digitization circuit with the incoming image. Usually **Synchronization Source** is the same as **Input Source**. However, when external gen-locking is used to synchronize cameras, **Synchronization Source** should always be set to the same channel to avoid desynchronization when changing the **Input Source**.

- Enable **Chrominance Filter** when using a color camera. Otherwise, leave this parameter disabled since it has an undesirable smoothing effect on the image.
- **White Level**, expressed in microvolts, sets the digitizer's white level voltage reference. Use this parameter to adjust the frame grabber's sensitivity with non-standard video signals.
- **Black Level**, expressed in microvolts, sets the digitizer's black level voltage reference. Use this parameter to adjust the frame grabber's sensitivity with non-standard video signals.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.
- **Trigger** sets whether the image acquisition is continuous or synchronized to an external signal event. Select **None** with the combo box to enable continuous acquisitions. Select **High to Low** or **Low to High** to respectively synchronize acquisitions with high to low or low to high transitions of an external trigger signal. Following a trigger event, the grabber will acquire the next image output by the camera.

Data Translation DT-3157

The DT-3157 is a digital frame grabber. It provides a single camera input.

Refer to the documentation provided with your frame grabber for hardware and software installation. HexSight uses the standard driver provided with the DT-3157, so no special installation procedure is required. See ***Installing a Data Translation Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the DT-3157 are available in the **Parameters** frame ([Figure 199](#)). The parameters are grouped into 3 tab-selected sub-panels: **Acquisition**, **Field** and **Digital I/O**.

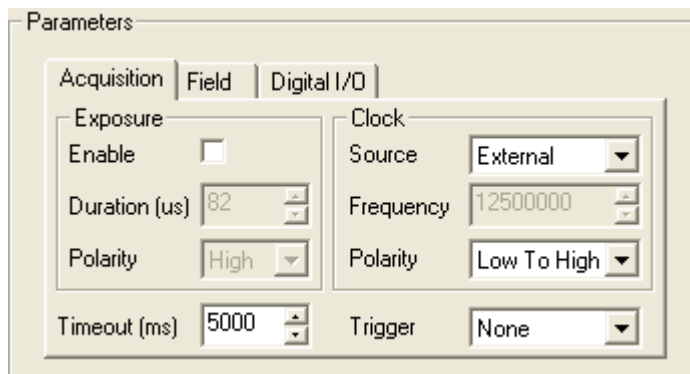


Figure 199 DT-3157 Acquisition Sub-Panel

Acquisition Sub-Panel

The Acquisition sub-panel (See [Figure 199](#)) configures the acquisition request.

- **Exposure** parameters allow you to activate (**Enable**) and define exposure of the camera. **Duration**, expressed in microseconds, sets the desired duration of time of the pulse to generate. **Polarity** sets the logic polarity of pulse which can be active **High** or active **Low**.
- **Clock** parameters allow you to configure the frame grabber input clocking source. **Source** indicates whether the input clocking should be generated by the frame grabber (internal) or provided by an outside source (external). **Frequency**, expressed in hertz, sets the internal input (pixel) clock frequency. **Polarity** specifies the polarity of the external input clocking source. If set to **High To Low**, synchronization is performed on high to low transitions. If set to **Low To High**, synchronization is performed on low to high transitions.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.

- **Trigger** sets whether image acquisition is continuous or synchronized to an external signal event. Select **None** with the combo box to enable continuous acquisitions. Select **High to Low** or **Low to High** to respectively synchronize acquisitions with high to low or low to high transitions of an external trigger signal. Due to compatibility problems with most cameras, the external trigger mode cannot be used to generate asynchronous resets. Instead, following a trigger event, the grabber will acquire the next image output by the camera.

Field Sub-Panel

The Field sub-panel (See [Figure 200](#)) configures the spatial resolution of input images.

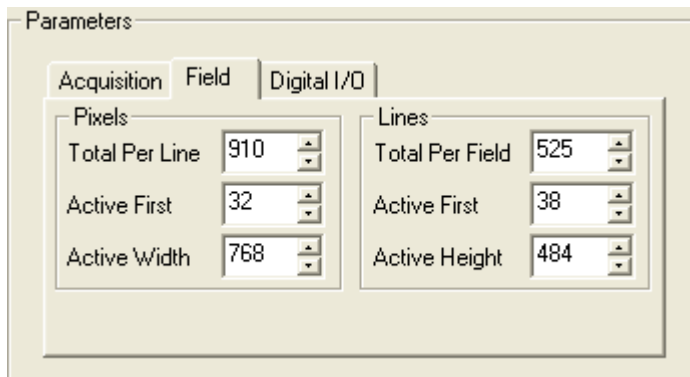


Figure 200 DT-3157 Field Sub-Panel

- **Pixels** parameters are used to configure the number of pixels per line in the active video area (**Total Per Line**), the beginning of the active video signal on the line, as a pixel value offset from the beginning of the horizontal sync (**Active First**), and the total number of pixels in a single horizontal line of video, where a horizontal line is defined as the area between two consecutive horizontal sync signals (**Active Width**).
- **Lines** parameters are used to configure the total number of lines in a single field of video, where a field is defined as the area between two consecutive vertical sync signals (**Total Per Field**), the beginning of the active video signal within the field, as a line offset

from the beginning of the vertical sync (**Active First**), and the number of lines per field (or non-interlaced frame) in the active video area (**Active Height**).

Digital I/O Sub-Panel

The Digital I/O sub-panel (See [Figure 201](#)) configures the digital input/output ports. A specific line can be set as an **Input**, as an arbitrary **Output**, as a fixed output **Set to 0** or as a fixed output **Set to 1**.

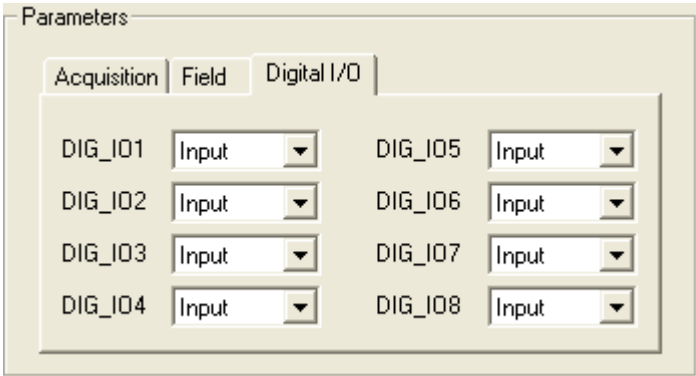


Figure 201 DT-3157 Digital I/O Sub-Panel

Imagenation PX610A

The PX610A is an interlaced/progressive scan frame grabber that provides inputs for four (4) cameras. Since it possesses only one acquisition circuit, each of these four channels must be acquired sequentially.

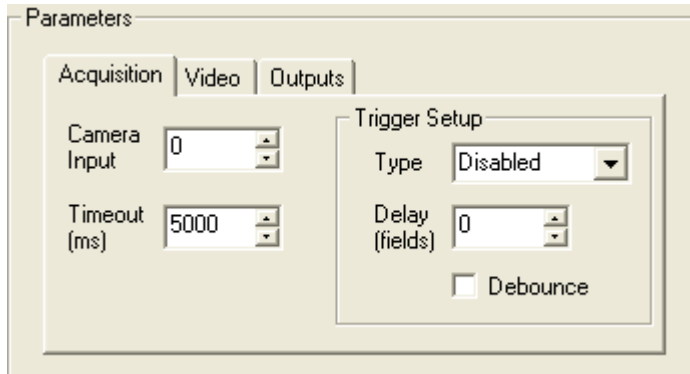


Figure 202 PX610A Setup

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing an Imagination Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the PX610A are available in the **Parameters** frame ([Figure 202](#)). The parameters are grouped into 3 tab-selected sub-panels: **Acquisition**, **Video** and **Outputs**.

Acquisition Sub-Panel

The Acquisition sub-panel (See [Figure 202](#)) configures the acquisition request. It selects from which camera the image is acquired and how it is triggered.

- **Camera Input** sets the camera input from which the image is acquired. The PX610A supports four (4) camera inputs but has only one video acquisition circuit. The inputs cannot therefore be acquired simultaneously.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.
- **Trigger Type** configures how the acquisition request is triggered. **Disabled** does not synchronize the acquisition request to a trigger event. It is processed immediately. **Level - High** causes the

Acquisition Device to wait for a high level signal on the trigger input before processing the acquisition request. **Level - Low** forces the Acquisition Device to wait for a low level signal on the trigger input. **Edge - Rising** synchronizes the acquisition request to a low-to-high transition on the trigger input, while **Edge - Falling** synchronizes it to a high-to-low transition.

- **Trigger Delay** sets a delay between the detection of a trigger event and the initiation of the acquisition request. This delay is expressed in number of video fields.
- When active, **Debounce** specifies that the trigger signal must return to its inactive state (low for **Edge - Rising** or high for **Edge - Falling**) for at least one video field before another trigger can be detected. **Debounce** has no effect with **Level - High** and **Level - Low** trigger types.

Video Sub-Panel

The Video sub-panel ([Figure 203](#)) configures the digitization circuit for the incoming video signal on the selected camera input.

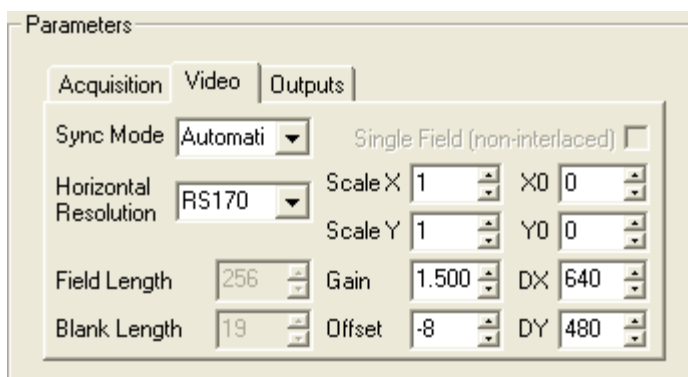


Figure 203 PX610A Video Sub-Panel

- **Sync Mode** specifies how the digitization circuit is synchronized to the incoming video signal. In the **Automatic** mode, the PX610A automatically detects the video format (NTSC/RS170 or PAL/CCIR) and synchronizes its digitization circuit accordingly. In the **Internal** mode, the board generates the video timing signals and ignores

synchronization information in the incoming video signal. In the **User** mode, the board synchronizes horizontally and vertically to the incoming video signal, but unlike the **Automatic** mode, the field length can be modified for compatibility with non-standard video sources. In the **WEN** mode, the board synchronizes vertically to the WEN signal received on the trigger input. Horizontal synchronization is provided by the incoming video signal.

- With the PX610A, the horizontal period cannot be adjusted. With the **Automatic**, **User** and **WEN** sync modes, the horizontal period is determined by the incoming video signal. With the **Internal** sync mode, the period is fixed to approximately 64us. **Horizontal Resolution** sets the number of pixels that should be digitized during each horizontal period. **RS-170** sets the number of pixels per video line to 640. **CCIR** sets the number of pixels per video line to 768. **RS-170** or **CCIR** horizontal resolutions can be selected, regardless of the actual format of the incoming video signal. For instance, if **CCIR** is selected while the actual format of the video source is RS-170/NTSC, 768 pixels per video line will be acquired instead of the standard 640. This setting will therefore affect the aspect ratio of the pixels in the acquired image. **Horizontal Resolution** can be set in any of the sync modes (**Automatic**, **Internal**, **User** and **WEN**).
- **Field Length** sets the number of active lines in a single video field. **Field Length** can be adjusted only when using the **Internal**, **User** or **WEN** sync modes. With the **Automatic** sync mode, it is automatically set to 243 if an RS-170/NTSC video source is detected and to 288 for CCIR/PAL video sources.
- **Blank Length** sets the number of lines in the vertical blank period of the video signal. **Blank Length** can be adjusted only when using the **Internal**, **User** or **WEN** sync modes. With the **Automatic** sync mode, it is automatically set to 19 if an RS-170/NTSC video source is detected and to 24 for CCIR/PAL video sources.
- **Single Field** specifies if the incoming video signal is composed of two fields (interlaced) or a single field (non-interlaced). **Single**

Field should be enabled when using a progressive scan camera as the video source. It should be disabled otherwise.

- **Scale X** sets the horizontal subsampling factor. A **Scale X** setting of 1 will sample all the pixels in the incoming video line. A value of 2 will sample 1 pixel for every 2 pixels in the video line, and so forth.
- **Scale Y** sets the vertical subsampling factor. A **Scale Y** setting of 1 will sample all the lines in the incoming video signal. A value of 2 will sample 1 line for every 2 lines in the video signal, and so forth.
- **X0** is used for cropping the image. This parameter is applied after the image is scaled by the values defined by **Scale X** and **Scale Y**. **X0** defines the first column that is digitized in the subsampled video line. A **X0** setting of 0 will start digitizing the subsampled video line at its first pixel. A value of 1 will start digitizing at its second subsampled pixel, and so forth.
- **DX** is used for cropping the image. This parameter is applied after the image is scaled by the values defined by **Scale X** and **Scale Y**. **DX** defines the number of columns that is digitized in the subsampled video line. A **DX** setting of 320 will digitize 320 pixels per video line, starting at column **X0**.
- **Y0** is used for cropping the image. This parameter is applied after the image is scaled by the values defined by **Scale X** and **Scale Y**. **Y0** defines the first line that is digitized in the subsampled video signal. A **Y0** setting of 0 will start digitizing the subsampled video signal at its first active line. A value of 1 will start digitizing at its second subsampled line, and so forth.
- **DY** is used for cropping the image. This parameter is applied after the image is scaled by the values defined by **Scale X** and **Scale Y**. **DY** defines the number of lines that is digitized in the subsampled video signal. A **DY** setting of 240 will digitize 240 lines per image, starting at line **Y0**.
- **Gain** sets the gain of the video digitizing circuit. The range of this parameter is 0.5 - 8.0 The default gain is 1.

- **Offset** sets the offset of the video digitizing circuit. The range of this parameter is -128 to 127. The default offset is -8, which tends to compensate for the video pedestal.

Typical PX610A Configurations

Examples of PX610A configurations for typical video sources are shown in [Table 10](#).

Table 10 Examples of PX610A Configurations with Typical Video Sources

Parameter	RS-170 Interlaced	RS-170 Non-Interlaced	CCIR Interlaced	CCIR Non-Interlaced
Horizontal Resolution	RS-170	RS-170	CCIR	CCIR
Field Length	243	486	288	576
Blank Length	19	39	24	49
Single Field	Disabled	Enabled	Disabled	Enabled
Scale X	1	1	1	1
Scale Y	1	1	1	1
X0	0	0	0	0
DX	640	640	768	768
Y0	4	4	0	0
DY	480	480	576	576

Outputs Sub-Panel

The Outputs sub-panel ([Figure 204](#)) configures the strobes and video synchronization output signals.

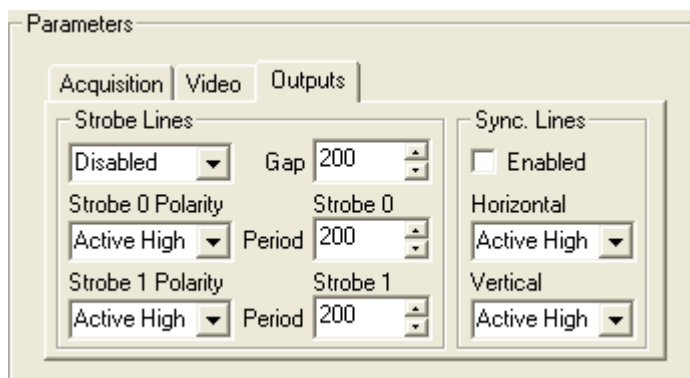


Figure 204 PX610A Outputs Sub-Panel

- The **Strobe Lines** mode configures how the strobe sequence is initiated. **Disabled** specifies that no strobe sequence is output. **Normal** makes it possible to fire a strobe sequence programmatically using the GrabberPx610aFireStrobe method. **Triggered** will fire a complete strobe sequence each time a trigger is detected.
- **Strobe 0 Polarity** specifies the polarity of the active phase of the strobe 0 output. **Active High** will output a high TTL level during the active phase of strobe 0. **Active Low** will output a low TTL level.
- **Strobe 1 Polarity** specifies the polarity of the active phase of the strobe 1 output. **Active High** will output a high TTL level during the active phase of strobe 1. **Active Low** will output a low TTL level.
- **Gap** specifies the gap between active phases of strobe 0 and strobe 1 in the strobe sequence. This parameter is expressed in number of video lines.
- **Strobe 0 Period** defines the duration of the active phase of strobe 0. This parameter is expressed in number of video lines.
- **Strobe 1 Period** defines the duration of the active phase of strobe 1. This parameter is expressed in number of video lines.
- When **Sync Lines Enabled** is set, both horizontal and vertical synchronization signals are driven on the synchronization lines

outputs. Otherwise, these outputs are kept in a high impedance state.

- **Horizontal** specifies the polarity of the active phase of the horizontal synchronization signal. **Active High** will output a high TTL level during the active phase of strobe 1. **Active Low** will output a low TTL level.
- **Vertical** specifies the polarity of the active phase of the vertical synchronization signal. **Active High** will output a high TTL level during the active phase of strobe 1. **Active Low** will output a low TTL level.

Imagenation PXC200

The PXC200 is an interlaced scan monochrome and composite color frame grabber. It provides inputs for up to 4 cameras. However, since it possesses only one acquisition circuit, each of the 4 channels must be acquired sequentially. Only monochrome acquisition modes are supported with the PXC200.

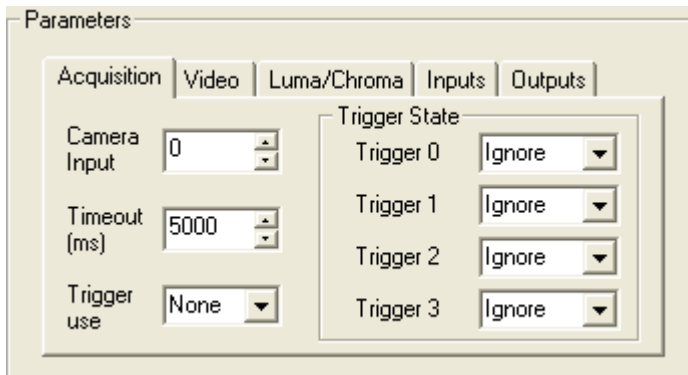


Figure 205 PXC200 Setup

PXC200 Setup

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing an Imagenation Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the PXC200 are available in the **Parameters** frame ([Figure 205](#)). The parameters are grouped into 5 tab-selected sub-panels: **Acquisition**, **Video**, **Luma/Chroma**, **Inputs** and **Outputs**.

Acquisition Sub-Panel

The Acquisition sub-panel (See [Figure 205](#)) configures the acquisition request. It selects the camera from which the image is acquired and how the camera is triggered.

- **Camera Input** sets the camera input from which the image is acquired. The PXC200 supports four (4) camera inputs but has only one video acquisition circuit. Therefore the different inputs cannot be acquired simultaneously.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.
- **Trigger use** configures how the acquisition request is triggered. **None** does not synchronize the acquisition request to a trigger event. **Any** will trigger the acquisition if any of the conditions configured for Trigger 0, 1, 2 or 3 is met. **All** will trigger the acquisition if the conditions configured for Trigger 0, 1, 2 and 3 are all met. See **Trigger State** for more details.
- The **Trigger State** is independently configured for Triggers 0, 1, 2 and 3. It sets the conditions that these triggers must meet in order to perform the acquisition request. Depending on the setting of **Trigger Use**, all or any of the four trigger conditions have to be met. See **Trigger Use** for more details. A trigger configured as **Set** will have to be set to be considered valid in the global trigger condition. A trigger configured as **Not Set** will have to be “not set” to be considered valid in the global trigger condition. A trigger configured as **Ignore** is ignored in the global trigger condition. The conditions defining the set state for each of the four triggers are configured with the Inputs sub-panel.

Video Sub-Panel

The Video sub-panel ([Figure 206](#)) configures the digitization circuit for the incoming video signal on the selected video input.

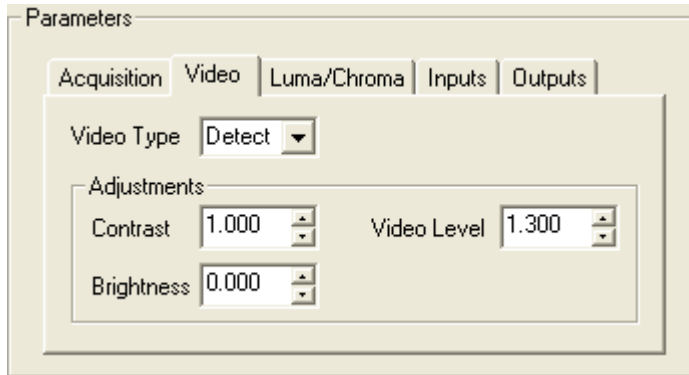


Figure 206 PXC200 Video Sub-Panel

- **Video Type** specifies the format of the incoming video signal. **Detect** will automatically detect the format of the incoming video signal. In this mode, switching between camera inputs may cause a delay because of the time required to detect the format. Specific video formats can also be configured. They are summarized in the following table:

Table 11 Summary of Specific Video Formats

Name	Signal Standard
NTSC	NTSC standard (640x480)
NTSCJ	NTSC with no pedestal voltage (640x480)
PAL	PAL B, D, G, H or I standards (768x576)
PALM	PAL M standard (768x576)
PALN	PAL N standard (768x576)
SECAM	SECAM standard (768x576)

- **Contrast** sets the monochrome gain of the video digitizing circuit. The range of this parameter is 0.0 - 2.0 The default contrast is 1, which leaves the incoming video signal unchanged before it is digitized.

- **Brightness** sets the monochrome offset of the video digitizing circuit. The range of this parameter is -0.5 to 0.5. The default brightness is 0, which leaves the incoming video signal unchanged before it is digitized.
- **Video Level** sets the voltage difference between the bottom of video synchronization pulses and bright white. The default video level is 1.3 volts.

Luma/Chroma Sub-Panel

The Luma/Chroma sub-panel ([Figure 207](#)) configures filters and processing that can be applied to the incoming video signal before it is digitized.

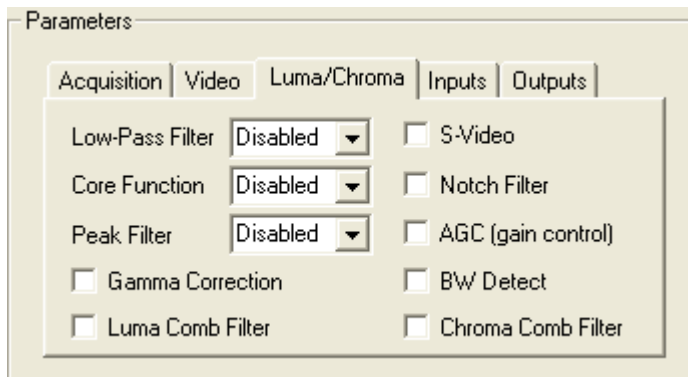


Figure 207 PXC200 Luma/Chroma Sub-Panel

- **Low-Pass Filter** activates a low-pass filter that reduces high-frequency information in the video signal. **Disabled** disables the low-pass filter. **Auto** selects the optimal filter based on the resolution. The low-pass filter can also be selected manually by specifying **Filter_1**, **Filter_2** or **Filter_3**. **Filter_3** has the highest level of filtering. Using a low-pass filter can affect edge positioning accuracy and should usually be disabled.
- **Core Function** forces all pixels with values below a specified level to be digitized as black. **Disabled** disables the coring function. **Core_8**, **Core_16** and **Core_32** will digitize as black pixels with grey-level values below 8, 16 and 32 respectively. Using a coring

function can affect edge positioning accuracy and should usually be disabled.

- **Peak Filter** activates a filter that amplifies high-frequency information in the video signal. **Disabled** disables the peak filter. The filter is selected manually by specifying **Filter_0**, **Filter_1**, **Filter_2** or **Filter_3**. **Filter_0** has the highest gain. Using a peak filter can sharpen edges in a blurry image but will affect edge positioning accuracy. It should usually be disabled.
- **Gamma Correction** activates a filter for gamma correction. For NTSC and PAL, gamma values of 2.2 and 2.8 are respectively used. Enabling gamma correction can affect edge positioning accuracy and should usually be disabled.
- **Luma Comb Filter** activates a comb filter to reduce artifacts in the monochrome signal caused by crosstalk from the color signal. Enabling the luma comb filter can affect edge positioning accuracy and should usually be disabled, *especially* when using a monochrome camera.
- When enabled, **S-Video** specifies that the incoming video signal is an S-Video signal with separate color and monochrome channels.
- **Notch Filter** activates a filter to remove the color burst signal from the video signal before it is digitized. This prevents color artifacts from appearing in the monochrome signal. Enabling the color burst notch filter can affect edge positioning accuracy and should usually be disabled, *especially* when using a monochrome camera.
- **AGC (Gain Control)** activates automatic gain control for color saturation to compensate for non-standard color signals. Since HexSight only uses the monochrome signal from a color video input, this parameter has no effect on the digitized image.
- When **BW Detect** is enabled, the color signal is set to zero when the board detects a missing or a weak color burst signal. Since HexSight only uses the monochrome signal from a color video input, this parameter has no effect on the digitized image.

- **Chroma Comb Filter** activates a comb filter to reduce color artifacts. Since HexSight only uses the monochrome signal from a color video input, this parameter has no effect on the digitized image.

Inputs Sub-Panel

The Inputs sub-panel ([Figure 208](#)) configures the trigger inputs (I/O lines 0-3).

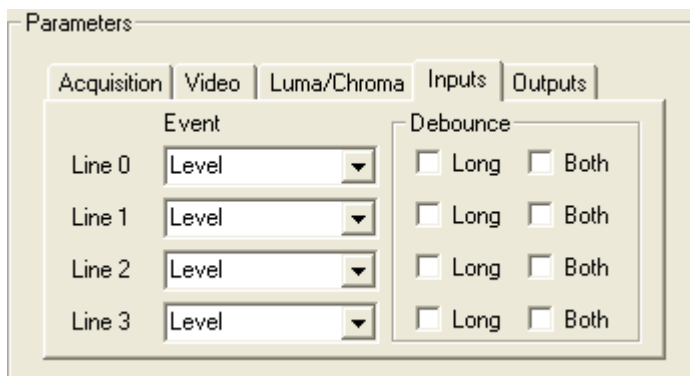


Figure 208 PXC200 Inputs Sub-Panel

- A specific **Event** parameter is defined for each of the four trigger inputs. It specifies the event which causes the trigger to be set. **Level** will set the trigger condition if a high TTL level is detected on the input. **Low to High** will set the trigger condition if a low-to-high transition is detected on the input. **High to Low** will set the trigger condition when a high-to-low transition is encountered.
- A specific **Debounce Long** parameter is defined for each of the four trigger inputs. When enabled, a delay of at least two vertical blanks is required before another low to high or high to low transition can be detected on the input. When **Debounce Long** is disabled, the minimum delay is one vertical blank. **Debounce Long** has no effect when the trigger **Event** for this input is **Level**.
- A specific **Debounce Both** parameter is defined for each of the four trigger inputs. When enabled, the delay is applied to both the latched low-to-high or high-to-low transition and the returning

transition on the input. When **Debounce Both** is disabled, only the latched low-to-high or high-to-low transition is debounced, not the returning transition.

Outputs Sub-Panel

The Outputs sub-panel ([Figure 209](#)) configures the strobe output signals (I/O lines 4-7).

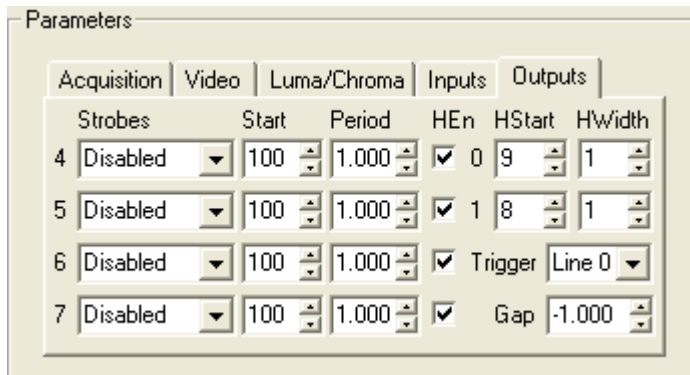


Figure 209 PXC200 Outputs Sub-Panel

- A specific **Strobe** parameter is defined for each of the four strobe outputs. **Strokes** sets the type of strobe that is generated on each output line. **Strokes** configures how the strobe signal is initiated. **Disabled** specifies that no strobe is output. **Sync Field 0** will generate a strobe pulse each time the line specified by **Start Line** is encountered in the video field 0. **Sync Field 1** will generate a strobe pulse each time the line specified by **Start Line** is encountered in the video field 1. **Sync Either** will generate a strobe pulse each time the line specified by **Start Line** is encountered in the video fields 0 and 1. **Trigger** will generate a strobe pulse each time a trigger condition is detected on the input line selected by the combo box **Trigger**.
- A specific **Start Line** parameter is defined for each of the four strobe outputs. **Start Line** sets the line in the video field at which to generate the strobe pulse. The valid range is 1 to 512.

- A specific **Period** parameter is defined for each of the four strobe outputs. **Period**, expressed in milliseconds, sets the width of the strobe pulse. The valid range is 64 microseconds to 4.2 seconds.
- A single **Trigger** parameter is defined for the four strobe outputs. **Trigger** selects which of the four input lines (**Line 0**, **Line 1**, **Line 2** or **Line 3**) is the trigger source for output strobes configured in the **Trigger** mode.
- A specific **HEn** (Holdoff Enable) parameter is defined for each of the four strobe outputs. When **HEn** is checked, the holdoff period configured with **HStart** and **HWidth** applies the associated strobe output.
- A specific **HStart** (Holdoff Start) parameter is defined for video fields 0 and 1. It sets the video line at which starts an holdoff period where no strobe can be fired. Strokes occurring during this period are delayed until the end of the holdoff period. The strobe holdoff period is ignored when using **Sync Field 0**, **Sync Field 1** and **Sync Either** strobe modes.
- A specific **HWidth** (Holdoff Width) parameter is defined for video fields 0 and 1. It sets the length, in number of video lines, of the strobe holdoff period. Strokes occurring during this period are delayed until the end of the holdoff period. The strobe holdoff period is ignored when using **Sync Field 0**, **Sync Field 1** and **Sync Either** strobe modes.
- **Gap**, expressed in milliseconds, sets the time between the two strobe pulses on line 7 when double pulse strobe output is enabled. Double pulse strobe output on line 7 is enabled when **Gap** is set to a value greater than or equal to 0. When set to a negative value, double pulse strobe output on line 7 is disabled.

Matrox Meteor-II / Standard

The Meteor-II/Standard is a standard (interlaced scan) monochrome and composite color frame grabber. It provides inputs for up to 7 cameras. However, since it possesses only one acquisition circuit, each of the 7

channels must be acquired sequentially. Only monochrome acquisition modes are supported with the Meteor-II/Standard.

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Matrox Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the Meteor-II/Standard are available in the **Parameters** frame ([Figure 210](#)).

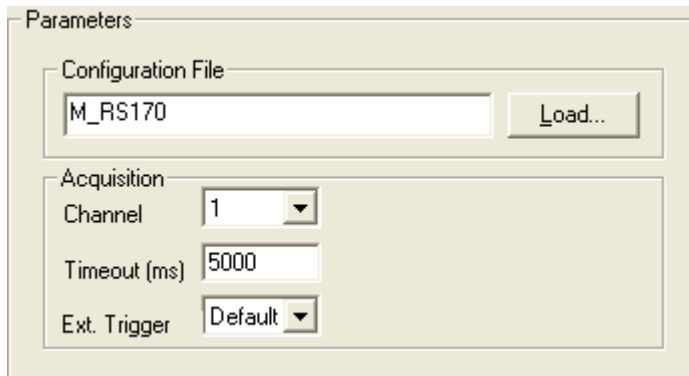


Figure 210 Meteor-II/Standard Setup

- The Meteor-II/Standard is configured through **Configuration Files** built with the Matrox Intellicam application provided with MIL or MIL-Lite software. The Acquisition Device directly supports the dcf file format. By default, the configuration file parameter is **M_RS170**. The *M_RS170* parameter is not a configuration file but rather a keyword that instructs the Meteor-II/Standard to use its standard *M_RS170* configuration. Other standard Meteor-II/Standard configurations are available and listed in [Table 12](#).

Table 12 Standard Configurations for the Meteor-II/Standard

Keyword	Standard Configuration
M_DEFAULT	NTSC, 640x480, 3x8 bits, 12.5MHz, composite (Color)
M_RS170	RS-170, 640x480, 8 bits, 12.5MHz, analog (Monochrome)

Table 12 Standard Configurations for the Meteor-II/Standard

Keyword	Standard Configuration
M_CCIR	CCIR, 768x576, 8bits, 14.8MHz, analog (Monochrome)
M_NTSC	NTSC, 640x480, 3x8 bits, 12.5MHz, composite (Color)
M_NTSC_YC	RS-170 Y/C (SVHS), 640x480, 3x8 bits, 12.5MHz (Color)
M_PAL	PAL I, 768x576, 3x8 bits, 14.8MHz, composite (Color)
M_PAL_YC	PAL Y/C, 768x576, 3x8 bits, 14.8MHz (Color)

- Entering any of the previous keywords in the **Configuration File** text box instead of a valid dcf file name, activates the corresponding standard Meteor-II/Standard configuration. In this version of HexSight only monochrome standard configurations for the Meteor-II/Standard are supported (M_RS170 and M_CCIR). Standard configuration keywords are case insensitive.
- When you enter a file name manually in the **Configuration File** text box or select a file using the **Load** button, the new configuration is not loaded immediately. The frame grabber configuration file is loaded and applied at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different dcf files because the Meteor-II/Standard will pause for a few seconds between each acquisition request.
- When using more than one Matrox frame grabber in the same application, each board is configured through its own dcf file. The frame grabbers can thus have different dcf files and still be used in an acquisition pipeline, as long as all requests for each board use the same dcf file.
- **Channel** sets the input channel from which the image is acquired. The Meteor-II/Standard supports 7 input channels with board revision 2.0 or higher and four (4) input channels otherwise. It has

only one video acquisition circuit. The channels cannot therefore be acquired simultaneously.

- **External Trigger** is used to specify that before acquiring the image, the Meteor-II/Standard should wait for the trigger event configured in the dcf file. **Default** uses the external trigger settings defined in the dcf file, **Enable** enables trigger detection and **Disable** disables trigger detection.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame.

Matrox Meteor-II / Multi-Channel

The Meteor-II/Multi-Channel is an RGB progressive/interlaced scan frame grabber that provides inputs for 2 RGB cameras. Since it possesses only one acquisition circuit, each of the 2 RGB channels must be acquired sequentially. HexSight processes the images acquired from the Meteor-II/Multi-Channel as three independent grey-scale images. This frame grabber can also be used for simultaneous synchronized multi-channel acquisitions for up to three gen-locked monochrome cameras. The Meteor-II/Multi-Channel supports features such as asynchronous resets, strobe lighting, and gen-locking.

Meteor-II / Multi-Channel Setup

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Matrox Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the Meteor-II/Multi-Channel are available in the **Parameters** frame ([Figure 211](#)).

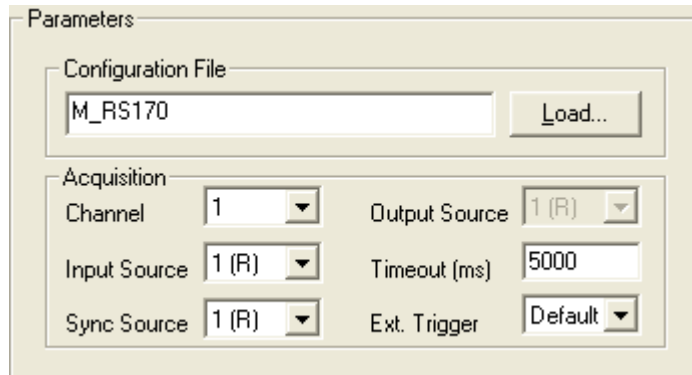


Figure 211 Meteor-II/Multi-Channel Setup

- The Meteor-II/Multi-Channel is configured through **Configuration Files** built with the Matrox Intellicam application provided with the MIL-Lite software. The Acquisition Device directly supports the dcf file format. By default, the configuration file parameter is **M_RS170**. The *M_RS170* parameter is not a configuration file but rather a keyword that instructs the Meteor-II/Multi-Channel to use its standard M_RS170 configuration. Other standard Meteor-II/Multi-Channel configurations are available and listed [Table 13](#).

Table 13 Standard Configurations for the Meteor-II/Multi-Channel

Keyword	Standard Configuration
M_DEFAULT	RS-170 RGB, 640x480, 3x8 bits, 12.5MHz (Color)
M_RS170	RS-170 using RGB module (Monochrome)
M_RS170_VIA_RGB	RS-170 using RGB module (Monochrome)
M_CCIR	CCIR using RGB module (Monochrome)
M_CCIR_VIA_RGB	CCIR using RGB module (Monochrome)
M_NTSC	RS-170 RGB, 640x480, 3x8 bits, 12.5MHz (Color)
M_NTSC_RGB	RS-170 RGB, 640x480, 3x8 bits, 12.5MHz (Color)

Table 13 Standard Configurations for the Meteor-II/Multi-Channel

Keyword	Standard Configuration
M_PAL	PAL I RGB, 768x576, 3x8 bits, 14.8MHz (Color)
M_PAL_RGB	PAL I RGB, 768x576, 3x8 bits, 14.8MHz (Color)

- Entering any of the previous keywords in the **Configuration File** text box instead of a valid dcf file name, activates the corresponding standard Meteor-II/Standard configuration. In this version of HexSight only monochrome standard configurations for the Meteor-II/Multi-Channel are supported (M_RS170, M_RS170_VIA_RGB, M_CCIR and M_CCIR_VIA_RGB). Standard configuration keywords are case insensitive.
- When you enter a file name manually in the **Configuration File** text box or select a file using the **Load** button, the new configuration is not loaded immediately. The frame grabber configuration file is loaded and applied at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different dcf files because the Meteor-II/Multi-Channel will pause for a few seconds between each acquisition request.
- When using more than one Matrox frame grabber in the same application, each board is configured through its own dcf file. The frame grabbers can thus have different dcf files and still be used in an acquisition pipeline, as long as all requests for each board use the same dcf file.
- **Channel** sets the RGB channel from which the image is acquired. The Meteor-II/Multi-Channel supports 2 RGB channels, but has only one RGB acquisition circuit. The two channels cannot therefore be acquired simultaneously.
- **Input Source** defines which of the inputs is acquired from the selected RGB channel. Individual R, G and B inputs can be acquired by selecting respectively **1 (R)**, **2 (G)** or **3 (B)**. This mode is useful for acquiring an image provided by a monochrome camera plugged to one of the RGB inputs. The three RGB inputs can also be

acquired simultaneously by selecting **RGB**. This is useful for simultaneous synchronized acquisitions of gen-locked monochrome cameras (up to 3). **None** can also be specified as the input source. In this case, no actual acquisition will take place. See the ***Using the Meteor-II / Multi-Channel in a Multi-Camera Application*** section for more details.

- **Output Source** specifies which of the acquired RGB inputs provides the image that will be created in the runtime database when the Acquisition Device is executed with the selected Configuration. Only one of the R, G or B image can be output at a time.
- When **Input Source** is set to an individual R, G or B input, the corresponding image is automatically set in **Output Source**, which means that when the Acquisition Device is executed, the image from a single input is acquired and then output to the runtime database.
- **Sync Source** sets the input that is used to synchronize the frame grabber digitization circuit with the incoming image. Any of the RGB inputs can be used as the synchronization source. An additional SYNC input can also be used. Usually **Sync Source** should be the same as **Input Source**. However, when external gen-locking is used to synchronize cameras, **Sync Source** should always be set to the same input to avoid desynchronization when changing the **Input Source**.
- **External Trigger** is used to specify that before acquiring the image, the Meteor-II/Multi-Channel should wait for the trigger event configured in the dcf file. **Default** uses the external trigger settings defined in the dcf file, **Enable** enables trigger detection and **Disable** disables trigger detection.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.

Using the Meteor-II / Multi-Channel in a Multi-Camera Application

When using the Meteor-II/Multi-Channel in a multi-camera application where the acquisition must be simultaneous, it is preferable to create a Configuration in the Acquisition Device for each of the R, G and B input to grab. This method is recommended instead of programmatically changing **Output Source** with a single Acquisition Device Configuration because a specific calibration can be set up for each input.

Creating a Three-Channel Application for the Meteor Multi-Channel

To create a three-channel application, define three Acquisition Device Configurations:

- Set the first one, named "Input1", with **Input Source** to **RGB** and **Output Source** to **1 (R)**.
- Define a second Configuration, "Input2", with **Input Source** set to **None** and Output Source set to **2 (G)**.
- In a third similar Configuration named "Input3", **Output Source** should be set to **3 (B)**.
- When the Acquisition Device is executed with the first Configuration, the three RGB inputs are grabbed and kept in memory and the image from the R input is output to the runtime database.
- When the Acquisition Device is executed with the second Configuration, no image is grabbed and the image from the G input is output to the runtime database.
- Similarly, when the Acquisition Device is executed with the third Configuration, no image is grabbed and the image from the B input is output to the runtime database.

The following example shows how to use these three Configurations programmatically in an acquisition pipeline.

```
Dim Acquisition as HSAcquisitionDevice
Set Acquisition = _
    ApplicationControl.ProcessManager.Process("Acquisition")
```

```

' Place a first request in the pipeline"
Acquisition.Start "Input1"
Acquisition.Start "Input2"
Acquisition.Start "Input3"

Do
    ' Place a new request in the pipeline
    Acquisition.Start "Input1"
    Acquisition.Start "Input2"
    Acquisition.Start "Input3"
    ' Wait for Input1 image
    Acquisition.Wait
    ' Process Input1 image
    ...
    ' Wait for Input2 image
    Acquisition.Wait
    'Process Input2 image
    ...
    ' Wait for Input3 image
    Acquisition.Wait
    ' Process Input3 image
    ...
Loop While Not ExitLoop

' Flush the acquisition pipeline
Acquisition.Wait
Acquisition.Wait
Acquisition.Wait

```

Matrox Meteor-II / Digital

The Meteor-II/Digital is a digital frame grabber. It provides inputs for either a single RGB digital camera or up to four 8 bits monochrome digital cameras. When connecting multiple 8 bits cameras to the Meteor-II/Digital, the grabber

must be configured to generate the timing signals in order to ensure synchronization between data channels.

Meteor-II/Digital Setup

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Matrox Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the Meteor-II/Digital are available in the **Parameters** frame ([Figure 212](#)).

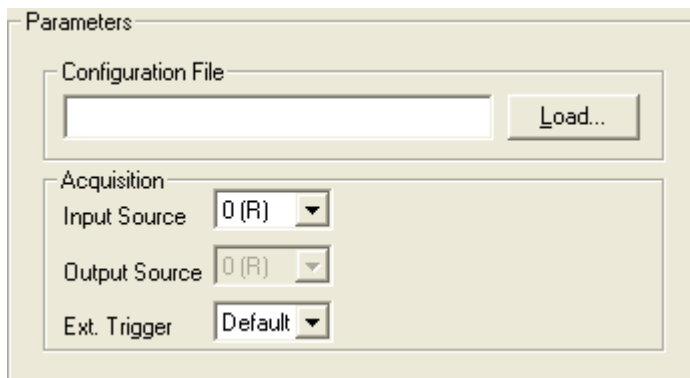


Figure 212 Meteor-II/Digital Setup

- The Meteor-II/Digital is configured through **Configuration Files** built with the Matrox Intellicam application provided with MIL or MIL-Lite software. The Acquisition Device directly supports the dcf file format. No default configuration is defined. You must provide a valid dcf file, or else an error is fired and no internal actions are taken.
- When you enter a file name manually in the **Configuration File** text box or select a file using the **Load** button, the new configuration is not loaded immediately. The frame grabber configuration file is loaded and applied at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different dcf files because

the Meteor-II/Digital will pause for a few seconds between each acquisition request.

- When using more than one Matrox frame grabber in the same application, each board is configured through its own dcf file. The frame grabbers can thus have different dcf files and still be used in an acquisition pipeline, as long as all requests for each board use the same dcf file.
- **Input Source** defines which of the inputs is acquired from the digital data input. Individual inputs can be acquired by selecting respectively **0 (R)**, **1 (G)**, **2 (B)** or **3**. This mode is useful for acquiring an image provided by a monochrome camera plugged to one of the data inputs. The three RGB inputs can also be acquired simultaneously by selecting **RGB**. This is useful for simultaneous synchronized acquisitions with monochrome cameras (up to 3). **None** can also be specified as the input source. In this case, no actual acquisition will take place. See the ***Using the Meteor-II / Digital in a Multi-Camera Application*** section for more details.
- In the RGB mode, **Output Source** specifies which of the acquired RGB inputs should provide the image that is created in the runtime database when the Acquisition Device is executed with the selected Configuration. Only one of the R, G or B images can be output at a time.
- When **Input Source** is set to an individual input, respectively **0 (R)**, **1 (G)**, **2 (B)** or **3**, the corresponding image is automatically set in **Output Source**, which means that when the Acquisition Device is executed, the image from a single input is acquired and then output to the runtime database.
- **External Trigger** is used to specify that before acquiring the image, the Meteor-II/Digital should wait for the trigger event configured in the dcf file. **Default** uses the external trigger settings defined in the dcf file, **Enable** enables trigger detection and **Disable** disables trigger detection. Note that the Meteor-II/Digital does not provide a configurable timeout. It is hard-coded to 20 seconds. If no trigger event is received or no camera is plugged to

the Meteor-II/Digital, the application will freeze for approximately 40 seconds between each acquisition request.

Using the Meteor-II /Digital in a Multi-Camera Application

When using the Meteor-II/Digital in a multi-camera application where the acquisition must be simultaneous, it is preferable to create a Configuration in the Acquisition Device for each of the R, G and B inputs to grab. This method is recommended instead of programmatically changing **Output Source** with a single Acquisition Device Configuration because a specific calibration can be set up for each input.

Creating a Meteor-II /Digital three-channel application

To create a three-channel application, define three Acquisition Device Configurations:

- Set the first one, named "Input1", with **Input Source** to **RGB** and **Output Source** to **0 (R)**.
- Define a second Configuration,"Input2", with **Input Source** set to **None** and Output Source set to **1 (G)**.
- In a third similar Configuration named "Input3", **Output Source** should be set to **2 (B)**.
- When the Acquisition Device is executed with the first Configuration, the three RGB inputs are grabbed and kept in memory and the image from the R input is output to the runtime database.
- When the Acquisition Device is executed with the second Configuration, no image is grabbed and the image from the G input is output to the runtime database.
- Similarly, when the Acquisition Device is executed with the third Configuration, no image is grabbed and the image from the B input is output to the runtime database.

The following example shows how to use these three Configurations programmatically in an acquisition pipeline.

```
Dim Acquisition as HSAcquisitionDevice
```



```

Set Acquisition = _
    ApplicationControl.ProcessManager.Process("Acquisition")

    ' Place a first request in the pipeline"
Acquisition.Start "Input1"
Acquisition.Start "Input2"
Acquisition.Start "Input3"

Do
    ' Place a new request in the pipeline
    Acquisition.Start "Input1"
    Acquisition.Start "Input2"
    Acquisition.Start "Input3"
    ' Wait for Input1 image
    Acquisition.Wait
    ' Process Input1 image
    ...
    ' Wait for Input2 image
    Acquisition.Wait
    ' Process Input2 image
    ...
    'Wait for Input3 image
    Acquisition.Wait
    'Process Input3 image
    ...
Loop While Not ExitLoop

' Flush the acquisition pipeline
Acquisition.Wait
Acquisition.Wait
Acquisition.Wait

```

Matrox Orion

The Orion is a standard (interlaced scan) monochrome and composite color frame grabber. It provides inputs for up to 8 cameras. However, since it possesses only one acquisition circuit, each of the 8 channels must be acquired sequentially. Only monochrome acquisition modes are supported with the Orion.

Refer to the documentation provided with your frame grabber for hardware and software installation and see ***Installing a Matrox Frame Grabber*** in the ***Installing HexSight*** chapter.

The configurable parameters for the Orion are available in the Parameters frame ([Figure 213](#)).

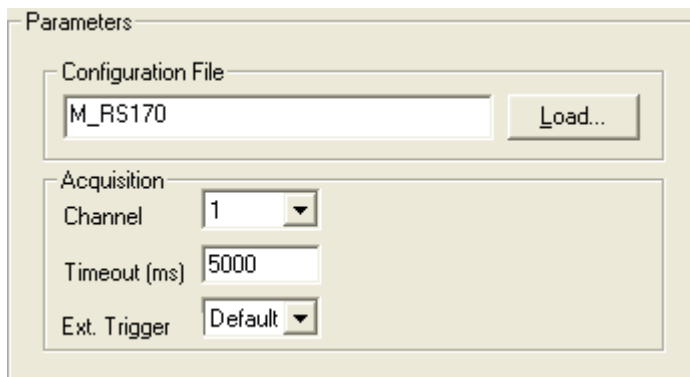


Figure 213 Orion Setup

- The Orion is configured through **Configuration Files** built with the Matrox Intellicam application provided with MIL or MIL-Lite software. The Acquisition Device directly supports the dcf file format. By default, the configuration file parameter is **M_RS170**. The *M_RS170* parameter is not a configuration file but rather a keyword that instructs the Orion to use its standard M_RS170 configuration. Other standard Orion configurations are available and listed [Table 14](#).

Table 14 Standard Configurations for the Orion

Keyword	Standard Configuration
M_DEFAULT	NTSC, 640x480, 3x8 bits, 12.5MHz, composite (Color)
M_RS170	RS-170, 640x480, 8 bits, 12.5MHz, analog (Monochrome)
M_CCIR	CCIR, 768x576, 8bits, 14.8MHz, analog (Monochrome)
M_NTSC	NTSC, 640x480, 3x8 bits, 12.5MHz, composite (Color)
M_NTSC_YC	RS-170 Y/C (SVHS), 640x480, 3x8 bits, 12.5MHz (Color)
M_PAL	PAL I, 768x576, 3x8 bits, 14.8MHz, composite (Color)
M_PAL_YC	PAL Y/C, 768x576, 3x8 bits, 14.8MHz (Color)

- Entering any of the previous keywords in the **Configuration File** text box instead of a valid dcf file name, activates the corresponding standard Orion configuration. In this version of HexSight only monochrome standard configurations for the Orion are supported (M_RS170 and M_CCIR). Standard configuration keywords are case insensitive.
- When you enter a file name manually in the **Configuration File** text box or select a file using the **Load** button, the new configuration is not loaded immediately. The frame grabber configuration file is loaded and applied at the first execution of the Acquisition Device. The first acquisition can thus take a few seconds to complete. Two Acquisition Device Configurations cannot therefore be used in a pipeline if they use different dcf files because the Orion will pause for a few seconds between each acquisition request.
- When using more than one Matrox frame grabber in the same application, each board is configured through its own dcf file. The frame grabbers can thus have different dcf files and still be used in

an acquisition pipeline, as long as all requests for each board use the same dcf file.

- **Channel** sets the input channel from which the image is acquired. The Orion supports 8 input channels. Since it has only one video acquisition circuit, the channels cannot be acquired simultaneously.
- **External Trigger** is used to specify that before acquiring the image, the Orion should wait for the trigger event configured in the dcf file. **Default** uses the external trigger settings defined in the dcf file, **Enable** enables trigger detection and **Disable** disables trigger detection.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame.

Opteon DepictMaster 4x1

The Opteon DepictMaster 4x1 is an integrated imaging system that combines frame grabber and video camera functionality. It consists of a PCI controller board into which up to four Opteon camera units can be plugged. HexSight only supports monochrome Opteon cameras.

Controller Board Setup

Refer to the documentation provided with your controller board for hardware and software installation. HexSight uses the standard driver provided with the Opteon DepictMaster 4x1 so no special installation procedure is required. See ***Installing an Opteon Controller Board*** in the ***Installing HexSight*** chapter.

The configurable parameters for the Opteon DepictMaster 4x1 are available in the Parameters frame ([Figure 214](#)).

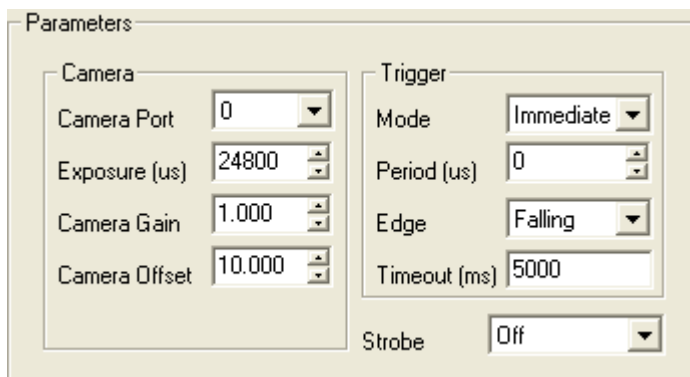


Figure 214 Opteon DepictMaster 4x1 Setup

- **Camera Port** specifies from which camera port the image is acquired. For information on synchronized multi-channel acquisitions, see the note in the **Trigger Mode** section.
- **Exposure**, expressed in microseconds, sets the exposure time of primary acquisitions on the camera selected with **Camera Port**.
- **Camera Gain** sets gain of the camera selected with **Camera Port**. The range of values is 0 to 255. Refer to the ***Depict Imaging System Programming Guide*** for more details.
- **Camera Offset** sets offset of the camera selected with **Camera Port**. The range of values is 0 to 255. Refer to the ***Depict Imaging System Programming Guide*** for more details.
- **Trigger** sets the trigger mode of the camera selected with **Camera Port**. **Immediate** uses no trigger, the acquisition request is processed immediately after all other pending acquisition requests are completed. **Timebase** mode is similar to **Immediate** except that the acquisition request will not be processed until the minimum amount of time, set by **Period**, has elapsed since the last acquisition request. **External A** will trigger acquisition requests on events (defined by **Edge**) of the external trigger input of the selected camera. **Master** will trigger acquisition requests on events (defined by **Edge**) of the master trigger input. **Slave** indicates that the acquisition request on the selected camera will be triggered by

the next non-slave acquisition. See ***Synchronizing Multi-Camera Acquisitions*** for more details.

- **Period** has two usages. In the **Timebase** trigger mode, it specifies the minimum time period between two acquisition requests. In **External A** and **Master** trigger modes, it specifies the hold-off time between the trigger event and the start of the acquisition.
- **Edge** sets the polarity of the event on the external trigger input of the selected camera in the **External A** trigger mode. It sets the polarity of the event on the master trigger input in the **Master** trigger mode.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a camera is not connected to the controller board.
- **Strobe** sets the strobe logic for the camera selected with **Camera Port**. **Off** disables the strobe of the selected camera. **PrimaryOnly** sets the strobe of the selected camera so it is fired at the start of each primary acquisition. The strobe signal stays high for the duration of the exposure.

Synchronizing Multi-Camera Acquisitions

The **Slave** mode is useful for synchronizing multi-camera acquisitions. Synchronized acquisitions are accomplished by placing an acquisition request in the pipeline for the slave camera and placing another acquisition request for the non-slave camera in the pipeline. The two cameras will be acquired simultaneously. The first Wait statement will retrieve the image acquired by the slave camera and the second Wait statement will retrieve the image from the non-slave camera.

For example to synchronize acquisitions on camera 1 to acquisitions on camera 0, define two Acquisition Device Configurations:

- Set the first one, named "Camera1", with **Camera Port** to 1 and **Trigger Mode** to **Slave**.

- Define a second Configuration, "Camera0", with **Camera Port** set to 0 and **Trigger Mode** set to **Immediate**.
- When a Acquisition Device Start statement is issued with the first Configuration, the acquisition request is left pending until a non-slave acquisition is requested. Note that if a Wait statement is issued at this point, the request will time out and no image is acquired.
- When a Acquisition Device Start statement is issued with the second Configuration, acquisition of both cameras is triggered.
- The first Wait statement issued will output the image from camera 1 to the runtime database.
- The second Wait statement issued will output the image from camera 0 to the runtime database.

The following example shows how to use these two Configurations programmatically in an acquisition pipeline.

```
Dim Acquisition as HSAcquisitionDevice
Set Acquisition = _
    ApplicationControl.ProcessManager.Process("Acquisition")

' Place a first request in the pipeline"
Acquisition.Start "Camera1"
Acquisition.Start "Camera0"

Do
    ' Place a new request in the pipeline
    Acquisition.Start "Camera1"
    Acquisition.Start "Camera0"
    ' Wait for Camera1 image
    Acquisition.Wait
    ' Process Camera1 image
    ...
    ' Wait for Camera0 image
    Acquisition.Wait
```

```

' Process Camera0 image
...
Loop While Not ExitLoop

' Flush the acquisition pipeline
Acquisition.Wait
Acquisition.Wait

```

DirectShow Device

The Generic DirectShow Device supports IEEE 1394 devices (FireWire camera for example) that use DirectX. Windows XP or Windows 2000 and DirectX 8.0 or later is required to use this type of device. The user interface provides access to camera parameters.

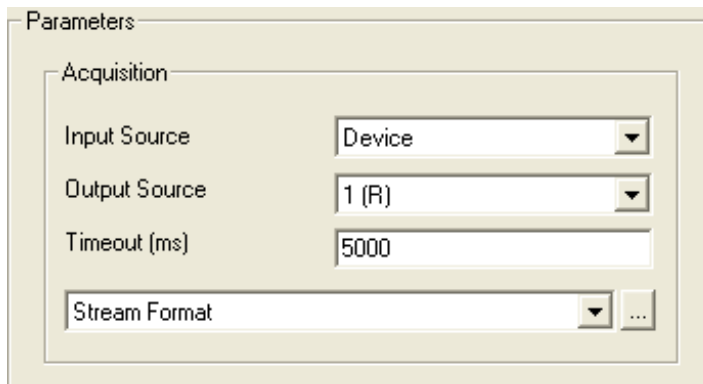


Figure 215 DirectShow Device Setup

- **Input Source** selects the device that is used for grabbing images.
- **Output Source** specifies which of the acquired RGB inputs should provide the image that is created in the runtime database when the Acquisition Device is executed with the selected Configuration. Only one of the R, G or B images can be output at a time.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not

freeze if, for example, a video source is not connected to the frame grabber.

- Use the combo box to select **Stream Format** or **Video Parameters**; then use the browser button (...) to access to the appropriate set of parameters.

GigE Vision Cameras

HexSight supports all GigE Vision compliant cameras. When you add a GigE vision camera, the driver is automatically installed for that camera, and it is ready to use right away.

NOTE: Windows Vista (or higher) users will have to run the HexSight application “As Administrator”, when adding their camera. To do this, right click on the executable, or the shortcut, and select “Run As Administrator”. This is *only* for the first time adding the camera. After the camera has been added successfully, the application can be run normally.

If you are using an Intel PRO based card, we provide an additional, optimized driver for the card itself. This can be enabled by checking the check-box marked “Force Optimal Drivers” when adding your camera in the Acquisition Device. If you decide to use this optimized driver, you will have to reboot before being able to access the camera.

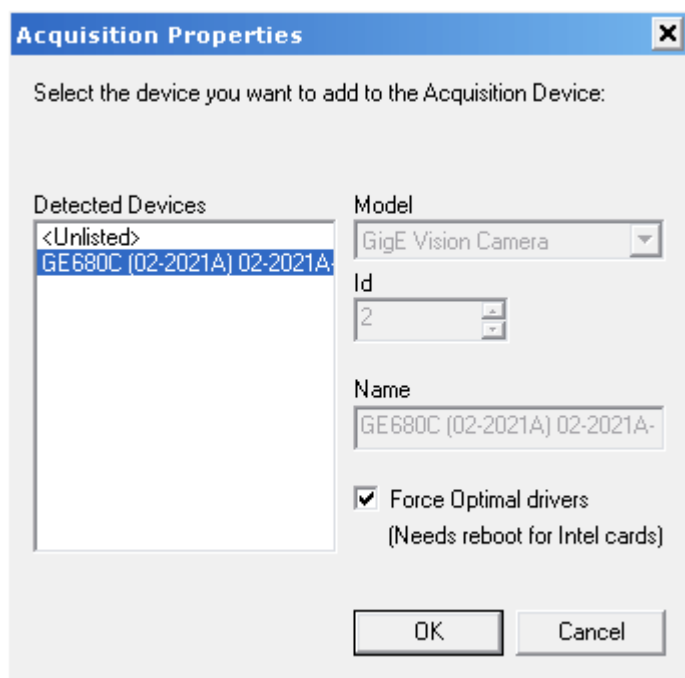


Figure 216 GigE Device Setup

Sony Intelligent Cameras

HexSight supports several Sony Intelligent Cameras and their APIs. The supported cameras are as follows:

- XCI-SX100
- XCI-SX100C
- XCI-V100
- XCI-V100C

HexSight also provides convenient access to the camera properties via GUI controls, seen below:

Acquisition Type Timeout (ms) 5000

Partial Scan

☐ Horizontal Start X Width

☐ Vertical Start Y Height

Shutter

Trigger Enabled ☐ Trigger Inhibit ☐

Trigger Polarity Shutter Delay 0

Shutter Speed Strobe Delay 0

Gain

AFE Gain 0.0

Auto Gain

Auto Gain ☐

Correction 0

Figure 217 Sony Intelligent Camera Setup

While not all parameters are available in the GUI, they are available via the HexSight programming interface. For full documentation on the API for these cameras, please refer to the Acquisition Device section of the HexSight Reference Manual.

DCAM Device

The support for IEEE 1394 devices (FireWire camera) can also be done using the CMU driver. The installation information on this driver can be found here:

<http://www.cs.cmu.edu/~iwan/1394/install.html>

Use this link to download the distributed driver:

<http://www.cs.cmu.edu/~iwan/1394/downloads/index.html>

The driver should replace the previous driver, if already present. Once it is installed, the AcquisitionDevice will see your installed devices when adding a new device.

Once added, the user interface provides access to new device parameters.

Acquisition Sub-Panel

The Acquisition sub-panel configures the acquisition request. It presents the video mode and other acquisition parameters.

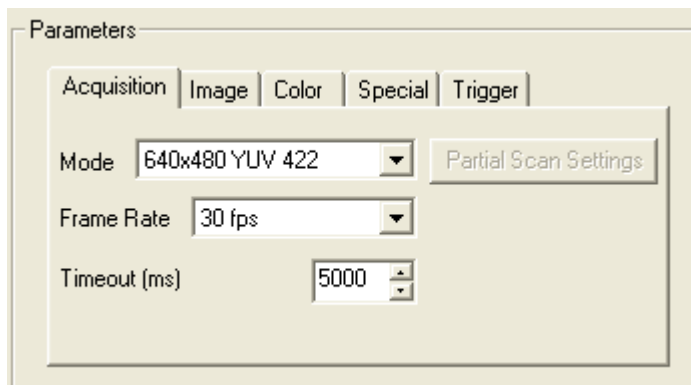


Figure 218 DCAM Device Acquisition Setup

- **Mode** selects the DCAM mode in which the device will be grabbing images.
- **Frame Rate** specifies the frame rate acquisition. Only the supported frame rates are shown.

- **Partial Scan Settings** is enabled when the Partial Scan mode is selected and applied. When depressed, a user interface for modifying the image size and pixel format will be shown.
- **Timeout**, expressed in milliseconds, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze if, for example, a video source is not connected to the frame grabber.

Image-Color-Special Sub-Panels

The following sub-panels configure the image captured. They present brightness, color, sharpness adjustments, and other acquisition parameters.

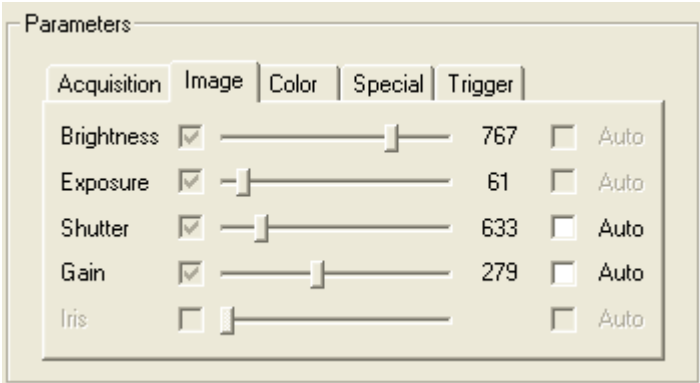


Figure 219 DCAM Device Image Setup

Trigger Sub-Panel

The Trigger sub-panel configures the triggering of the images captured. If the device supports triggering, the trigger mode, source, polarity and delay are defined in this sub-panel.

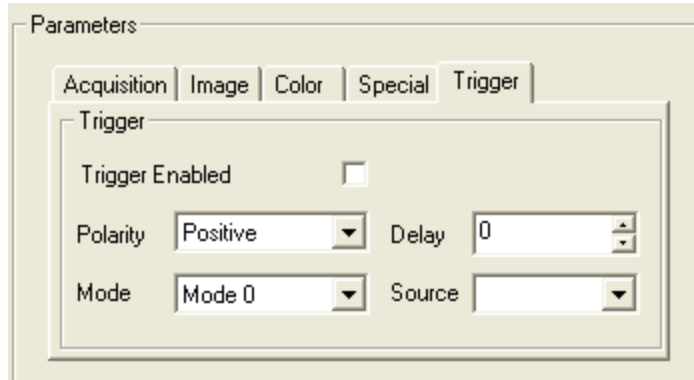


Figure 220 DCAM Device Trigger Setup

While not all parameters are available in the GUI, they are available via the HexSight programming interface. For full documentation on the API for these cameras, please refer to the Acquisition Device section of the HexSight Reference Manual.

Gocator Sensor

There is nothing to install to add a Gocator Sensor to the AcquisitionDevice. If you do not see the Gocator in the Add Device dialog box, use the drop down list and select the LMI sensor entry.

Once added, the user interface provides access to new device parameters.

Attributes Parameters

The Attributes sub-panel contains the information to connect with the Gocator sensor. Enter the IP address of the Gocator to reach and set the current acquisition mode setup in the Gocator. Validate all other acquisition parameters.

Warning: Selecting a different mode in the parameters and in the Gocator web-interface, as well as not properly configuring the outputs in the Gocator web-interface may result in an acquisition time out.

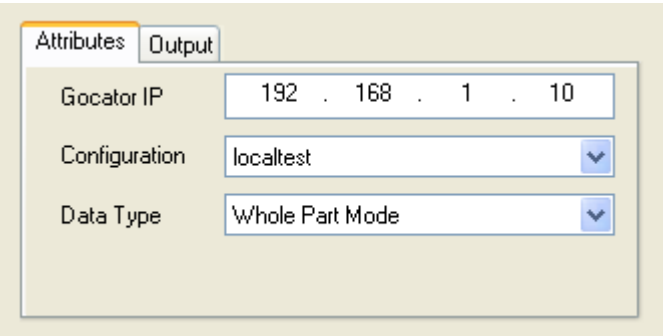


Figure 221 Gocator Sensor Attributes Setup

- **Gocator IP** selects the IP address to reach the device that will be grabbing images.
- **Configuration** specifies the configuration to set on the device.
- **Data Type** specifies the acquisition mode. Only the supported modes are shown, see table below.

Table 15 Data Types available

Data Type	Description
Profile Mode	Stitches a number of profiles to create a height map image
Profile Intensity Mode	Stitches a number of intensity profiles to create a grey-scale image
Profile and Intensity Mode	Dual acquisition of profiles stitching and intensity profiles stitching, resulting in two acquired images
Whole Part Mode	Acquisition of a whole part as one height map image

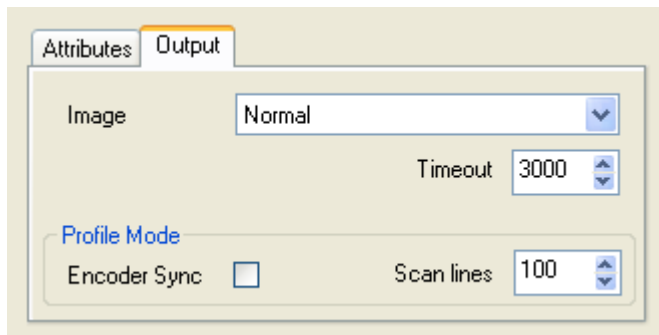
Table 15 Data Types available

Data Type	Description
Whole Part Intensity Mode	Acquisition of a whole part as one intensity image
Whole Part and Intensity Mode	Dual acquisition of a whole part as two images of both height map and intensity types

Output Parameters

The Output sub-panel contains the information on how to handle the Gocator sensor's data; whether the image needs to be transposed or not and how to stitch the profiles.

Warning: Selecting a different mode in the parameters and in the Gocator web-interface, as well as not properly configuring the outputs in the Gocator web-interface may result in an acquisition time out.

**Figure 222** Gocator Sensor Output Setup

- **Image setting** can be used to modify the way data is acquired. Image can be normal or transposed.
- **Timeout**, sets the maximum acceptable time between the initiation of the acquisition request and its completion. This parameter ensures that the application does not freeze.

- **Encoder Sync** is used to reorder the captured image to synchronize the beginning of the image with the encoder's tick in profile mode. Useful in rotating applications.
- **Scan lines** is the number of profiles captured that will form the image. Once this number of profile has been acquired, the sensor is stopped.

The only control that HexSight is exercising on the Gocator is the Start and Stop command. HexSight does not have the ability to detect or adapt to unexpected Gocator configurations. For full documentation on the API for these cameras, please refer to the Acquisition Device section of the HexSight Reference Manual.

Calibration Sub-Panel

Use this panel to select a calibration mode and access the selected calibration method dialog. Note that to achieve subpixel accuracy, you **MUST** calibrate your optical setup. This panel also allows you to save and load calibration files.

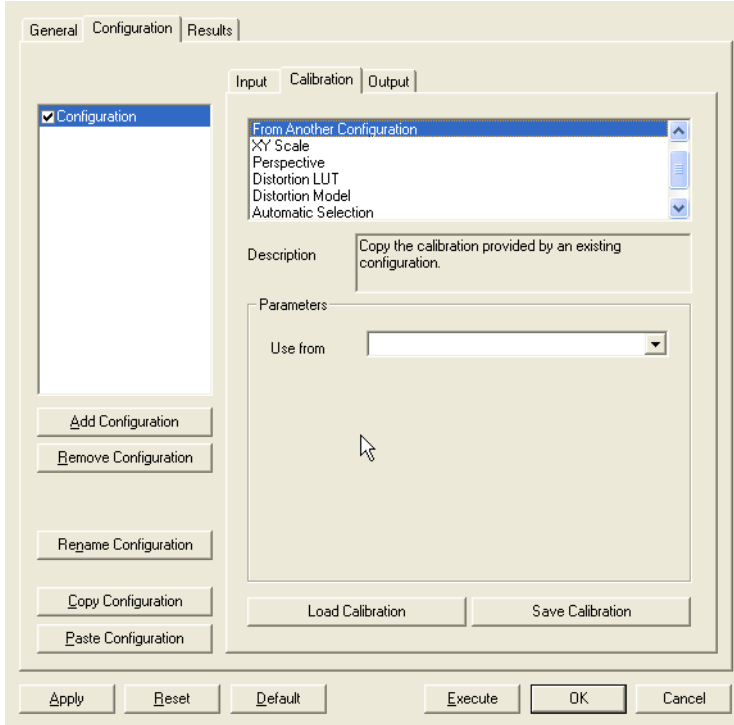


Figure 223 Calibration Sub-Panel

Why is Calibration Necessary?

Calibration of the optical setup ensures accurate results, especially for inspection and location applications that must be tolerant to rotation. Due to the rectangular aspect ratio of image pixels, possible misalignment of the camera with respect to the work surface and lens distortion, deformation of parts that are rotated and/or translated in the image can be substantial and exceed a few pixels.

Using Calibration Files

Calibration files (*.cal) store the information from a calibration setup. These files can be saved and loaded to share calibration parameters between applications using the same optical setup.

- Use the **Load Calibration** button to load a HexSight calibration file into the selected Acquisition Device Configuration.
- Use **Save Calibration** button to save calibration parameters to be used with another application sharing the same optical setup.

Selecting a Calibration Method

Besides the five basic calibration methods: **XY Scale**, **XYZ Calibration**, **Perspective**, **Distortion LUT** and **Distortion Model** described in detail further in this chapter the Calibration Panel also presents the following three options:

- The **Original Calibration** mode is useful only when using **File Emulation** as the input device for the selected Acquisition Device Configuration. With this calibration mode the calibration parameters stored with the emulation image are copied to the Image that is output to the runtime database.
- The mode named **From Another Configuration** allows you to apply calibration parameters defined for another Acquisition Device Configuration.
- **Automatic Selection** provides a display of calibration parameters that have been defined programmatically. See the ***Programmatic (Automatic) Calibration*** section.

XY Scale Calibration

XY Scale Calibration is a quick and handy calibration method but is less precise than distortion type methods. You should use this method when it is not possible to set up a target in the operating environment.

- This method does not correct perspective or lens deformations. To achieve proper precision, the camera must therefore be aligned to ensure that its optical axis is perfectly perpendicular with respect to the work surface.
- To use an object as a calibration target with this method, you must provide one height and one width measure in calibrated world units and use the calipers in the calibration dialog to compute the corresponding height and width in the image.

Procedure for XY Scale Calibration

1. In the **Inputs** sub-panel, select the input source that will provide the target image. In the **Calibration** sub-panel and select the **XY Scale** method.
2. Click **Show Calibration Dialog** to open **XY Scale Calibration** dialog. See [Figure 224](#).

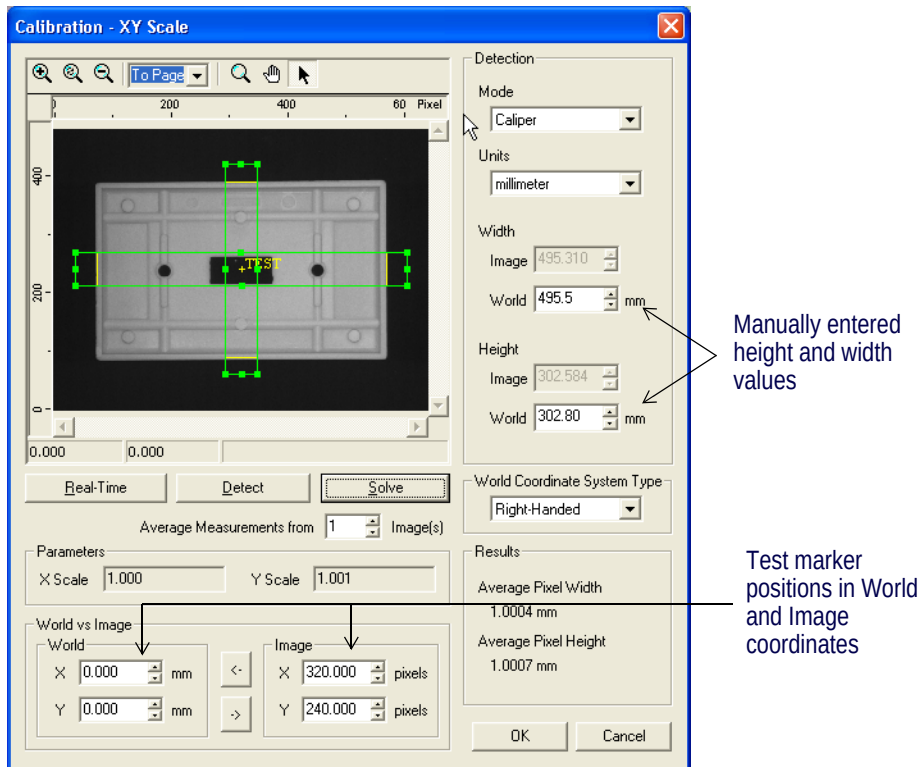


Figure 224 XY Scale Calibration Dialog

3. Select the **World Coordinate System Type**, either **Right-Handed** or **Left-Handed**. This can be different from the selected **Image Coordinate System Type**, set through the Input Sub-Panel.
4. Click the **Real-Time** button to view and place the target in the field of view. Make sure the image is well-contrasted and in focus.

5. Select a detection mode: **Caliper** or **Manual** and follow the corresponding setup:

Caliper Mode Setup

The Caliper mode provides measurement tools that are used to determine the height and the width of the target object in the image.

- a) If necessary, drag the rectangles into position to measure object features not centered in the image.
- b) Adjust the caliper measurement tools (green rectangles) to fit the object. The length of a caliper rectangle must exceed the boundaries of the object by at least 10 pixels.
- c) The thickness of the caliper rectangle adjusts the projection length, that is the number of pixels that are averaged to build the one-dimensional signal used to detect edges.
- d) The two outermost edges enclosed by the rectangle are used to compute the measurement.
- e) Click **Detect** to acquire a new image and carry out automatic width and height measurements. Edges used to compute the measurements appear as yellow lines.

Manual Mode Setup

When Image and World measures are already known by external means you can use the Manual mode to set up the XY Scale Calibration.

- a) Enter the object's width and the height in pixels in the Image Width and Image Height text boxes. The position provided for the points must be calculated with respect to the selected **World Coordinate System Type**.
- b) When you already know the average width and height of the pixel when projected on the work. Enter 1 pixel for both the **Image Width** and **Height** measures, select the appropriate unit system and enter the actual width and height in **World** units.
- c) The manual mode can also be used to compensate only for the pixel aspect ratio in applications where the actual dimensions are not required. If the X/Y ratio is 1.1, enter 1 pixel for both the **Image**

Width and **Height** measures. Select **calibration unit** in the **Units** combo box, enter 1 as the **World Width** and 0.909 as the **World Height**.

4. Using the **Units** combo box, select the unit system into which you want to calibrate the system.
5. In the **World Width** and **World Height** text boxes, enter the actual physical width and height corresponding to the caliper measurements.
6. Enter the number of images to use for the calibration in the **Average Measurements from** text box. The calibration is carried out on each image and the results are averaged to compute the calibration parameters.
7. Click **Solve**. The **Results** frame displays the average pixel width and height computed from the calibration parameters.
8. Use the **World vs. Image** controls to visually test the calibration.
 - a) To find World coordinates of a feature in the image: Drag the TEST crosshair marker over the feature then click <- to convert the Image position to a World position.
 - b) To find an Image position corresponding to a given World position: Enter the World position in the **X World** and the **Y World** text boxes then click the -> button. The TEST crosshair moves to corresponding image location displayed in the **X Image** and **Y Image** text boxes.

XYZ Calibration

XYZ Calibration is a specific calibration method used for 3D sensors. It will be automatically set in images captured with a Gocator sensor, unless specified otherwise. It is the same as a XY Scale calibration, but also holds the Z dimension information: offset and resolution.

To compute the height information of a given pixel, use the following formula:

$$\text{Height} = \text{ZOffset} + \text{PixelValue} * \text{ZResolution}$$

This calibration cannot be set manually, it will depend on the sensor's current settings. It can be consulted in the HSAcquisitionDevice control, or obtained programmatically. It is also possible to load and save this calibration for external usage.

All HexSight images, saved in HexSight's format, contain the calibration it was captured with. It is possible to get the calibration's information from an HIG image, for future reference.

Perspective Calibration

Perspective calibration is a simple method that uses four points to define the transform between Image and World coordinates. This method is more precise than XY Scale and compensates for perspective deformation but does not correct lens distortion. It is therefore less precise than the available distortion calibration methods.

- You should use this method when it is not possible to set up a target in the operating environment.
- To use an object as a calibration target with this method, you must provide the calibrated world coordinates of four (4) points on the object and use the corner detection tools to detect the position of these four points in the image. Optionally, you can enter the image positions manually by dragging markers onto the four points in the image.

Procedure for Perspective Calibration

1. In the **Inputs** sub-panel, select the input source that will provide the target image. In the **Calibration** sub-panel, select the **Perspective** method.
2. Click **Show Calibration Dialog** to open the **Perspective Calibration**. See [Figure 225](#).
3. Select the **World Coordinate System Type**, either **Right-Handed** or **Left-Handed**. This can be different from the selected **Image Coordinate System Type**, set through the Input Sub-Panel.

- Click the **Real-Time** button to view and place the target in the field of view. Make sure the image is well contrasted and in focus. Deactivate the real time view by clicking the button again.

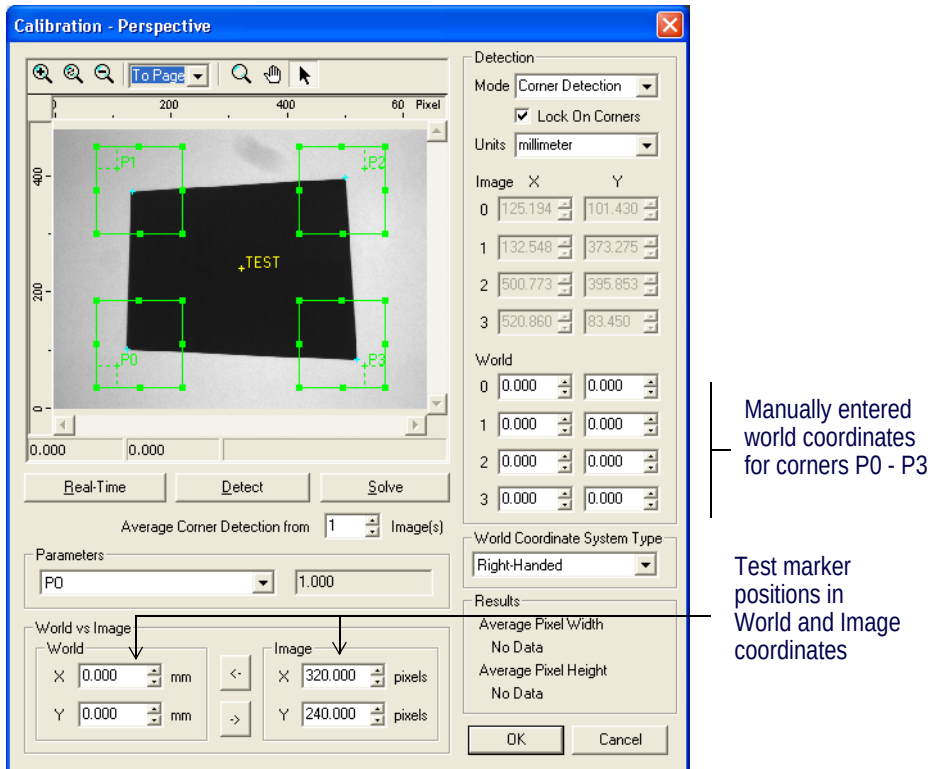


Figure 225 Perspective Calibration Dialog

- Select the **Detection Mode**, either: **Corner Detection** or **Manual**, and follow the corresponding setup:

Corner Detection Mode Setup

The Corner Detection mode provides corner detection tools that can automatically locate corners on objects. These tools appear as four (4) green rectangles in the display, as shown in [Figure 225](#). [Figure 226](#) illustrates the positioning of the corner detection rectangles.

- a) Click **Detect** to acquire a new image and carry out automatic detection of corners. The detected corners are highlighted by light blue crosshairs. If no valid corner is detected by a given corner detection tool, it will appear in red instead of green.
- b) If necessary, drag the corner detection tools to place their approximate corner position area onto actual corners in the image. You can also stretch the tool to match its linear regression segments to actual straight edge segments in the image. Click **Detect** again to acquire a new image and detect the 4 corners of the target object.
- c) The **Lock on Corners** check box locks the position of the corners to the detected corners, which means that if you click **Detect**, the detection tools will stick to the object if it is moved.

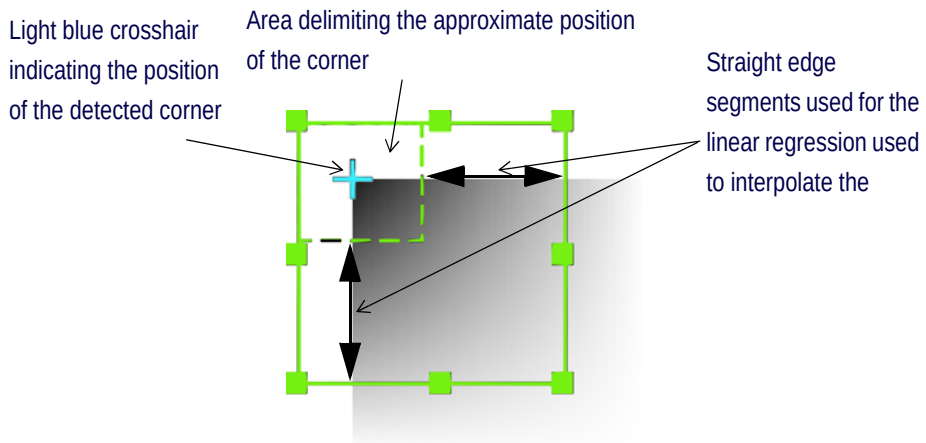


Figure 226 Corner Detection Mode

Manual Mode Setup

When the four points of known position on the object do not consist of corners between straight segments you can manually place the points in the image.

- a) Drag the green crosshair markers onto points of known position on the object.

- b) Using the **Image** text boxes, you can refine the image positions for the four points.
3. Using the **Units** combo box, select the unit system into which you want to calibrate the system.
4. Enter the actual World coordinates of the four detected corners in the **World** text boxes. The position provided for the points must be calculated with respect to the selected **World Coordinate System Type**.
5. Enter the number of images to use for the calibration in the **Average Measurements from** text box. The calibration is carried out on each images and the results are averaged to compute the calibration parameters.
6. Click **Solve**. The **Results** frame will display the average pixel width and height computed from the calibration parameters.
7. Use the **World vs Image** controls to visually test the calibration:
 - a) To find World coordinates of a feature in the image: Drag the yellow TEST crosshair marker over the feature then click <- to convert the Image position to a World position.
 - b) To find an Image position corresponding to a given World position: Enter the World position in the **X World** and the **Y World** text boxes then click the -> button. The yellow TEST crosshair moves to corresponding image location displayed in the **X Image** and **Y Image** text boxes.

Distortion LUT

Distortion calibration methods are usually recommended over the XY Scale or Perspective methods since they compensate for lens distortion as well as perspective deformations.

- The Distortion LUT method requires a grid of dots target, from which it builds a lookup table to convert between World and Image positions.
- The position of a point is interpolated using the four closest dots. This method is *VERY* sensitive to the accuracy of the calibration

target. It is recommended over the Distortion Model method only for applications where lens distortion is very significant.

Distortion Model

As with Distortion LUT, the Distortion Model method compensates for lens distortion as well as perspective deformations.

- This method requires a grid of dots target from which it builds a non-linear model using a best-fit approach, taking into account all of the target dots.
- Since it averages out errors, this method is less sensitive to the accuracy of the target. It is usually more precise than the Distortion LUT method with lens of 16 mm or greater focal length.
- For typical applications, Distortion Model is therefore the recommended calibration method.

Procedure for a Distortion Calibration

The following procedure for can be used for either a Distortion Model or a Distortion LUT calibration.

1. In the Inputs sub-panel, select the input source that will provide the target image. In the Calibration sub-panel, select the Distortion LUT or the Distortion Model method.
2. Click **Show Calibration Dialog** to open the Distortion Calibration dialog illustrated in [Figure 227](#).
3. The display in the dialog shows the following elements:
 - **Yellow crosshair markers** indicate detected calibration dots. These are the dots that are used for calibration calculations.
 - **Blue crosshair markers** indicate detected validation dots. These dots are not mandatory but are useful for error calculations in the integrated validation procedure.
 - **Red crosshair markers** indicate detected calibration dots that were not used to calculate the calibration. This generally occurs when using an invalid calibration target. Another cause may be an incorrectly placed coordinate system marker.

- **The X-Y axes marker** indicates the origin of the coordinate system. By default, this origin is centered on the calibration dot closest to the bottom left corner for a right-handed coordinate system and on the dot closest to the top left corner for a left-handed coordinate system.
4. Select the **World Coordinate System Type**, either **Right-Handed** or **Left-Handed**. This can be different from the selected **Image Coordinate System Type**, set through the Input Sub-Panel.
 5. Click the **Real-Time** button to view and place the target in the field of view. The dots should cover the entire image. Make sure the image is well-contrasted and in focus. Deactivate the real time view by clicking the button again.

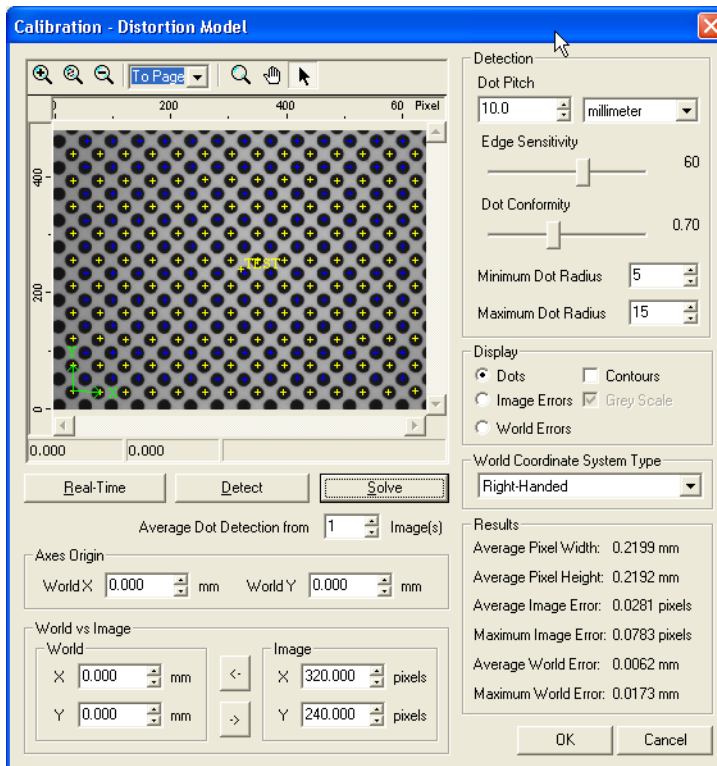


Figure 227 Distortion Model Dialog

6. All dots should be detected, except for those in rows and columns that border the image. If target dots are not being properly detected, you may need to adjust **Detection** parameters:
 - **Edge Sensitivity:** Sets the sensitivity to contrasts for the dots detection process. Increasing this parameter will help to detect low contrast dots but may add unwanted noise that could cause false detections. Decreasing this parameter may reduce noise, but cause misses in the target. The default value should work fine with well-contrasted calibration targets.
 - **Dot Conformity:** When the dots are detected, an average form factor is computed. The dot conformity parameter determines how closely a given dot must match the average form factor to be accepted as a valid dot. The default value should work in most cases. However in cases where lens distortion is extreme or the perspective deformation is substantial, this parameter should be decreased to detect all the dots on the target.
 - **Minimum Dot Radius:** Defines the minimum radius (in pixels) required for a dot to be valid. Optimal dot radius is approximately 10 pixels. Actual dot radius in the image should be between 6 to 12 pixels.
 - **Maximum Dot Radius:** Defines the maximum radius (in pixels) required for a dot to be valid. Optimal dot radius is approximately 10 pixels. Actual dot radius in the image should be between 6 to 12 pixels.
7. In the Detection frame, set the **Dot Pitch**, and the calibration units used to express the dot pitch. **Dot Pitch** is the distance between the centers of each calibration dot. The same dot pitch is used for validation dots.
8. To detect the target dots, click **Detect**. This will acquire a new image and detect the dots using the configured detection parameters.

If the detection parameters are not correctly adjusted there can be misses in the lower left corner that will cause problems with the placement of the World coordinate system axes such as shown in [Figure 228](#).

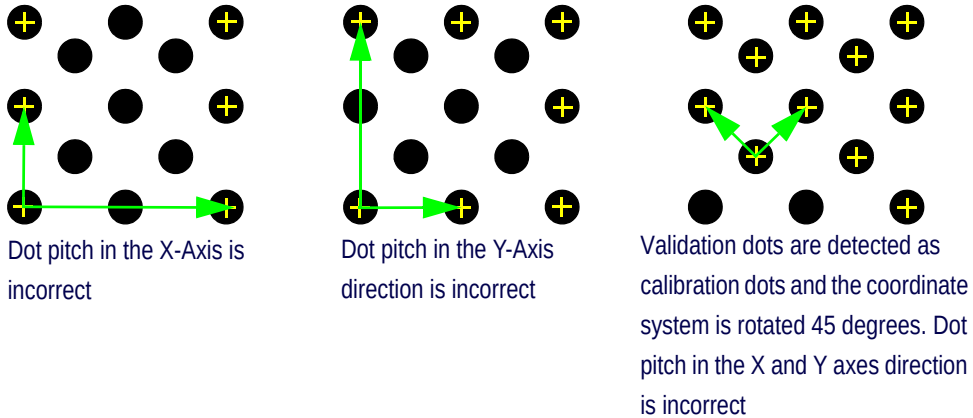


Figure 228 Dot Detection Problems

9. If necessary properly place the World coordinate system axes marker. If the axes are wrongly placed because dots are missing, modify the detection parameters and click **Detect**. Repeat until all the dots are detected. Drag the origin of the World coordinate system axes to a dot on the target. The actual World coordinates for this dot are entered in the **Axes Origin** text boxes.
10. Once the origin is set, drag the X and Y axes to the two orthogonal dots next to the origin. This will set the orientation for the World coordinate system.
11. Click **Detect** again to detect the dots using the new coordinate system. Dots crosshair markers should appear in yellow for calibration dots and blue for validation dots.
12. If some dots cannot be detected properly, display their contours by selecting **Dots** in the **Display** frame and enabling the **Contours** check box. Irregularities on the contour may show up, in which case the **Edge Sensitivity** and **Dot Conformity** parameters can be adjusted to accept the missing dot. For optimal results with the Distortion LUT method, no dot should be missing, except on the border of the image. The Distortion Model method is tolerant to a few missing dots.
13. Enter the number of images to use for the calibration in the **Average Dot Detection from** text box. Dot detection and location are carried

out on the number of images specified in the text box and results are averaged to compute the calibration parameters.

14. Click **Solve**. Once the calibration is complete, the **Results** frame, as illustrated in [Figure 229](#), displays the error statistics.

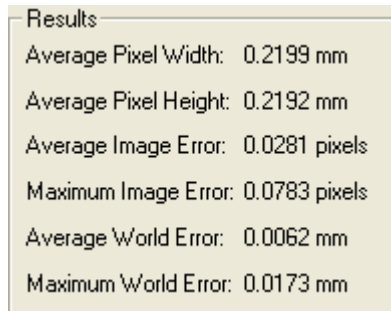


Figure 229 Calibration Results after Solving the Calibration

15. Use the **World vs. Image** controls to visually test the calibration.
16. To find World coordinates of a position in the image: Drag the **TEST** crosshair marker to a position in the image. Click <- to convert the Image position to a World position.
17. To find an Image position corresponding to a given World position: Enter the World position in the **X World** and the **Y World** text boxes then click the -> button. The **TEST** crosshair moves to corresponding image location displayed in the **X Image** and **Y Image** text boxes.
18. Use **Display** options to view a graphical representation of either Image or World errors.

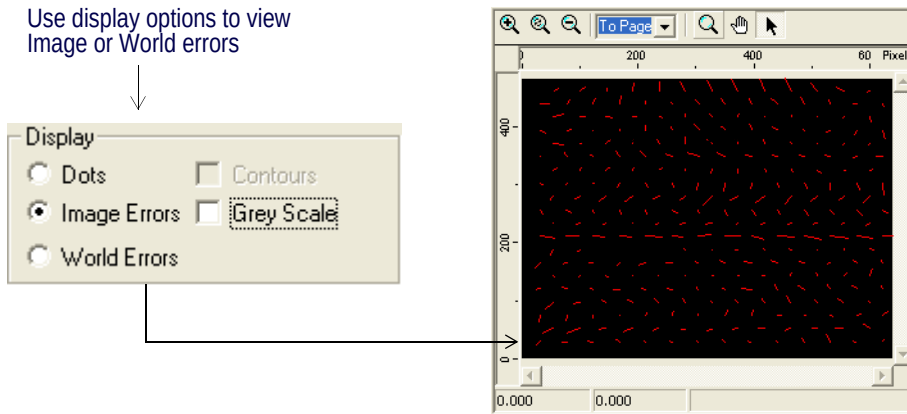


Figure 230 Viewing Errors Vectors

Note: These error vectors are largely magnified to provide an indication of the areas where the errors are more significant and should not be used as an indication of the absolute error.

Dot Targets

The quality and precision of a grid of dots target have a direct impact on the overall precision of the results of your application. Dot targets are commercially available but you can also create your own targets by following the guidelines provided below.

Description of Dot Target Elements

- A dot target is made up of a matrix of evenly spaced identical calibration dots.
- A secondary matrix of validation dots can be added in offset to calibration dots to validate the calibration process. Although they are not absolutely required, these dots are useful for error calculations.
- Dots in both matrices should be identical in size and have the same pitch.

- **Dot Pitch** describes the distance between the centers of two dots in the same matrix. Dot pitch must be the same in the X and Y directions.
- The offset between the calibration dots matrix and the validation dot matrix must be $\frac{1}{2}$ dot pitch in the X and Y axes.

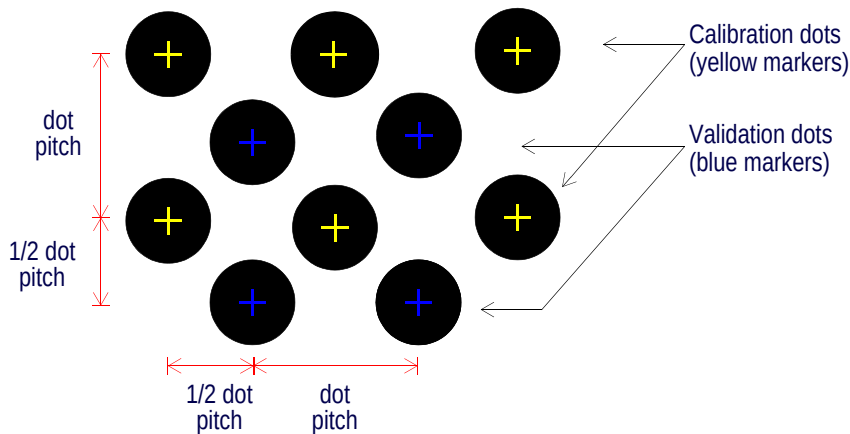


Figure 231 Dot Target Elements

Guidelines for Creating a Dot Target

- Dots should be circular; the recommended radius range is 6 to 12 pixels.
- Dot pitch must be the same in the X and Y axes.
- Dot pitch should be 4 times the dot radius.
- The target should cover the entire field of view.
- Dots should be well-contrasted.
- For best results, targets should be of high photo quality on a stable medium, not printed.
- Validation dots are not required to measure calibration errors, but are recommended for validating the LUT calibration.
- Validation dots should be offset by $\frac{1}{2}$ the dot pitch in the X and Y-axes.

Automatic Calibration

This calibration method provides a way of calibrating an application entirely programmatically, without the use of the HexSight calibration interface. This feature is especially useful for integrating HexSight into another software that already has its own calibration procedure.

- If you select Automatic Selection in the user interface you can read but not set any of the calibration parameters.
- The number of calibration points that you provide determines the actual calibration method that is applied: XY Scale, Perspective, Distortion Model or Distortion LUT.

Procedure for an Automatic Calibration

You must programmatically select the Automatic Selection calibration method and provide calibration data as pairs of corresponding Image/World points. The Automatic Selection method mode selects the type of calibration to apply according to the number of calibration points that have been entered, as described in [Table 16](#).

Table 16 Use of Calibration Points by the Automatic Calibration Method

Number of pairs of points	Automatically selected calibration method
0 or 1	Invalid
2 or 3	Scale
4 to 8 included	Perspective
9 and more	Distortion Model

Setting up the Automatic Selection Calibration

1. You must first programmatically select the calibration method using the Acquisition Device control property **CalibrationMethod** (**SetCalibrationMethod** in Visual C++).
2. Next you must provide the calibration data through the indexed **CalibrationParameter** property (**SetCalibrationParameter** in Visual C++). The description of each index is provided in the following table:

Table 17 Index of Automatic Calibration Parameter Properties

Index	Description
0	Number of pairs of calibration points used for the calibration
1	Reserved for future use
2	Reserved for future use
3	Reserved for future use
4	Reserved for future use
5	Reserved for future use
6	Width of the image in pixels
7	Height of the image in pixels
8	Read only: average width of an image pixel
9	Read only: average height of an image pixel
10	X coordinate of point 1 expressed with respect to the Image coordinate system
11	Y coordinate of point 1 expressed with respect to the Image coordinate system
12	X coordinate of point 1 expressed with respect to the World coordinate system
13	Y coordinate of point 1 expressed with respect to the World coordinate system
14	X coordinate of point 2 expressed with respect to the Image coordinate system
15	Y coordinate of point 2 expressed with respect to the Image coordinate system
16	X coordinate of point 2 expressed with respect to the World coordinate system
17	Y coordinate of point 2 expressed with respect to the World coordinate system
...	...
$10+4(n-1)$	X coordinate of point n expressed with respect to the Image coordinate system
$10+4(n-1)+1$	Y coordinate of point n expressed with respect to the Image coordinate system

Table 17 Index of Automatic Calibration Parameter Properties

Index	Description
$10+4(n-1)+2$	X coordinate of point n expressed with respect to the World coordinate system
$10+4(n-1)+3$	Y coordinate of point n expressed with respect to the World coordinate system

Performing the Calibration

- To perform the calibration, the Acquisition Device must be executed once after the modification of the calibration parameters. Subsequent executions of the Acquisition Device will not trigger a calculation of the calibration unless the calibration parameters are modified once more.

Validating the Calibration

- You can verify the calibration procedure through the **ValidateCalibration** method of the Acquisition Device control. **ValidateCalibration** returns True if the system was able to calculate a calibration from the given points. Otherwise, the method returns False.

Example of a Programmatic Calibration

Lets consider a case where four points are used for the calibration.

Table 18 Programmatic Calibration Example

Point	Image X	Image Y	World X	World Y
1	100.0	50.0	-1500.0	2300.0
2	500.0	50.0	1500.0	2300.0
3	100.0	400.0	-1500.0	6100.0
4	500.0	400.0	1500.0	6100.0

The programming code for the automatic calibration would then be:

```
Dim lManager as HSProcessManager
Set lManager = HSApplication1.ProcessManager
```

```

Dim AcquDev as HSAcquisitionDevice
Set AcquDev = lManager.Process("Acquisition Device")

AcquDev.CalibrationMethod = hsAuto
' Number of pairs of calibration points
AcquDev.CalibrationParameter(0) = 4
' Calibration point 1
AcquDev.CalibrationParameter(10) = 100.0
AcquDev.CalibrationParameter(11) = 50.0
AcquDev.CalibrationParameter(12) = -1500.0
AcquDev.CalibrationParameter(13) = 2300.0
' Calibration point 2
AcquDev.CalibrationParameter(14) = 500.0
AcquDev.CalibrationParameter(15) = 50.0
AcquDev.CalibrationParameter(16) = 1500.0
AcquDev.CalibrationParameter(17) = 2300.0
' Calibration point 3
AcquDev.CalibrationParameter(18) = 100.0
AcquDev.CalibrationParameter(19) = 400.0
AcquDev.CalibrationParameter(20) = -1500.0
AcquDev.CalibrationParameter(21) = 6100.0
' Calibration Point 4
AcquDev.CalibrationParameter(22) = 500.0
AcquDev.CalibrationParameter(23) = 400.0
AcquDev.CalibrationParameter(24) = 1500.0
AcquDev.CalibrationParameter(25) = 6100.0
' Execute the process in order to perform the calibration
AcquDev.Execute
' Validation of the Calibration
Dim CalibrationOk as Boolean
CalibrationOk = AcquDev.ValidateCalibration

```

Note: The previous example is provided only to show the different steps of an automatic calibration. In a real application, calibration data would be acquired dynamically.

Output Sub-Panel

Use this panel to define the output for the selected Acquisition Device Configuration.

View

View sets the name of the view to which the **Image** acquired by the Acquisition Device is stored in the runtime database. The default value for the name of the output View is *Acquisition*.

- Modifying this parameter can be useful in a multi-channel acquisition application to prevent images from different channels from overlapping each other in the runtime database.
- To store the **Image** from each channel in its own View in the runtime database, assign a different value to **View** (such as *Acquisition0*, *Acquisition1*, etc.) for each Configuration used by the Acquisition Device to acquire the images.

Image

Image sets the name of the Image created in the runtime database by the Acquisition Device. This Image is stored in the View set by **View**. The default value for the name of the output image is *Image*.

- Modifying this parameter can be useful in a multi-channel acquisition application to prevent Images from different channels from overlapping each other in the runtime database.
- To store the image from each channel with a specific name in the runtime database, assign a different value to **Image** (such as *Image 0*, *Image 1*, etc.) for each Configuration used by the Acquisition Device to acquire the images.

Results Panel

This panel shows the following read only results.

- **Elapsed time:** time required by the last execution of the Acquisition Device.
- **Configuration Name:** name of the Configuration used for the last execution of the Acquisition Device.
- **Time Stamp:** indicates the time at which the image was acquired
- **Error Code:** error code encountered for the last execution of the Acquisition device.
- **Error Description:** description of the last error encountered.

Inspection Tools

This chapter introduces the basic notions and procedures needed for setting up and using HexSight inspection tools as well as other HexSight tools that can be model-based.

Inspection Tools

[List and Description of HexSight Tools](#)

[Inspection Tool Basics](#)

[Rectangle Characteristics](#)

[Sector Characteristics](#)

[Tool Positioning Modes](#)

[Using Inspection Tools](#)

[Basic Method](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

List and Description of HexSight Tools

This section lists HexSight tools that are documented in greater detail in the ***HexSight Tools User Guide***, which can be installed from the HexSight installation CD. The content of the User Guides and the ***HexSight Reference Manual*** is contained in the online HexSight HTML Help.

Finder Tools

Arc Finder: The Arc Finder finds and locates circular features on objects and returns the coordinates of the center of the arc, the start and end angles, and the radius.

Line Finder: The Line Finder finds and locates linear features on objects and returns the line angle and point coordinates.

Point Finder: The Point Finder finds and locates point features on objects and returns the angle as well as the coordinates of the found point.

Builder: The Builder creates geometric entities such as points, lines or arcs from user-defined parameters or by combining other geometric entities.

Image Processing Tools

Image Processing Tool: The Image Processing Tool processes grey-scale images by applying arithmetic, assignment, logical, filtering, morphological or histogram operators. Users can define custom filtering operators.

Note: This tool cannot be model-based. It is typically used to preprocess images that will be used by other HexSight tools.

Image Sharpness Tool: The Image Sharpness Tool computes the sharpness of preponderant edges in a user-defined region of interest.

Image Histogram Tool: The Image Histogram tool computes greylevel statistics within a user-defined region of interest.

Sampling Tool: The sampling tool is used to extract a section of an image and output it as a separate Image. The sampling tool can be configured to operate as model-based. It can be used by a HexSight application to apply a custom processing on a region of interest on an object.

Inspection Tools

Edge Locator: The Edge Locator finds and locates an edge or a set of edges that meet user-defined criteria.

Arc Edge Locator: The Edge Locator finds and locates an edge or a set of edges in an arc- or circular-shaped area.

Caliper: The Caliper finds and locates one or more edge pairs and measures distances between the two edges within each pair.

Arc Caliper: The Arc Caliper finds and locates one or more edge pairs on an arc or circular shaped area and measures distances between the two edges within each pair.

Pattern Locator: The Pattern Locator finds and locates instances of a greyscale pattern occurring within an Image. Pattern images are provided by the HexSight Locator control.

Blob Analyzer: The Blob Analyzer finds and locates *blobs* within a defined area on an Image and returns various results for each blob.

Symbology Tools (optional)

Barcode Reader: The Barcode Reader reads and extracts information from 1D symbologies commonly known as bar codes.

Data Matrix Reader: The Data Matrix Reader reads and extracts information from 2D symbologies commonly known as data matrixes.

OCR Fixed Font: The OCR Fixed Font tool recognizes and reads fixed-font character strings.

Inspection Tool Basics

This section explains the basic terms and notions used in the Inspection tools. Inspection tools use pixel greylevel values within bitmapped images to process information.

Inspection tools are applied on a specific area of an Image corresponding to a set of features to be analyzed. This area, or region of interest, within which an inspection tool carries out its action, is usually bounded by a user-defined affine rectangle, simply referred to as the Rectangle. In Arc inspection tools the user-defined region of interest is a Sector instead of a Rectangle. This chapter mostly covers Rectangle regions of interest but the concepts usually apply to Sector regions of interest.

All Inspection tools can use a Model-based positioning mode that facilitates the development and implementation of HexSight applications.

Rectangle Characteristics

The properties used to define the Rectangle region of interest, are its position, height, width, rotation and skew.

Tool Coordinate System

The center of the Rectangle, defined as the intersection of diagonals joining opposing corners, determines the origin of the Tool coordinate system. The Tool coordinate system is displayed as an X-Y axes marker in the HexSight interface. See the [Coordinate Systems](#) chapter of this guide for more details.

Rectangle Position

The Rectangle's position is specified as the X-Y coordinates of the Tool coordinate system origin.

The coordinate system and unit system used to express the Rectangle position depend on which tool positioning mode is configured. See the [Tool Positioning Modes](#) section for more details.

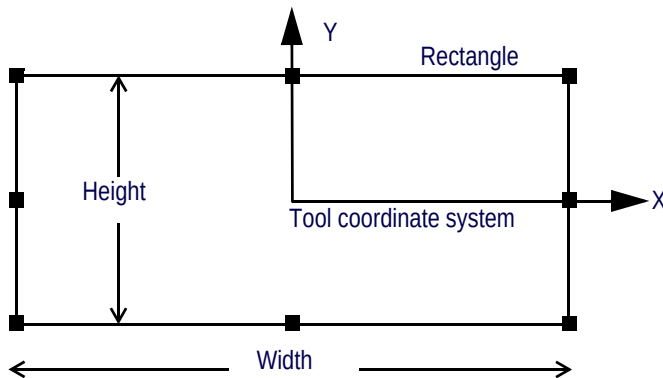


Figure 232 Inspection Tool Rectangle

Height and Width

The Rectangle's height corresponds to the length of its Y-axis side. The Rectangle's width corresponds to the length of its X-axis side.

The coordinate system and unit system used to express the Rectangle width and height depend on which tool positioning mode is configured. See the [Tool Positioning Modes](#) section for more details.

Rotation

Rotation is the angle between the Rectangle's X-Axis and the X-Axis of the coordinate system used for tool positioning. This coordinate system depends on the positioning mode that is configured. See the [Tool Positioning Modes](#) section for more details.

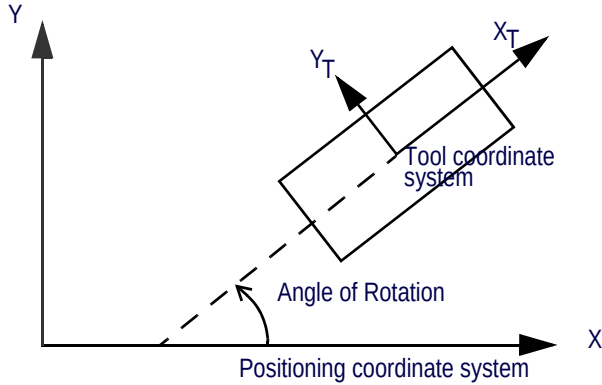


Figure 233 Rotation

Skew

Skewing can be used with some Inspection tools for matching the inclination of features within an object. The angle of skew is defined as the angle between the Y-axis of an orthogonal Tool coordinate system and the Y-axis of the skewed Tool coordinate system.

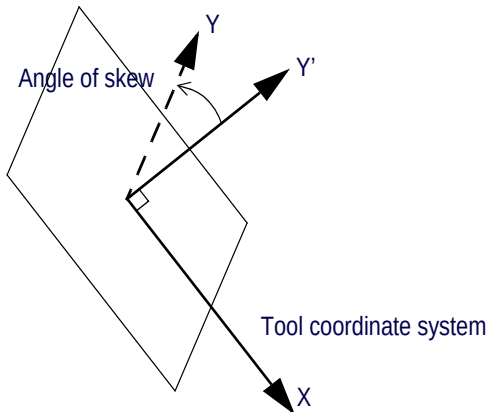


Figure 234 Skew

Sector Characteristics

The properties used to define the Sector are its position, opening, thickness, rotation and radius.

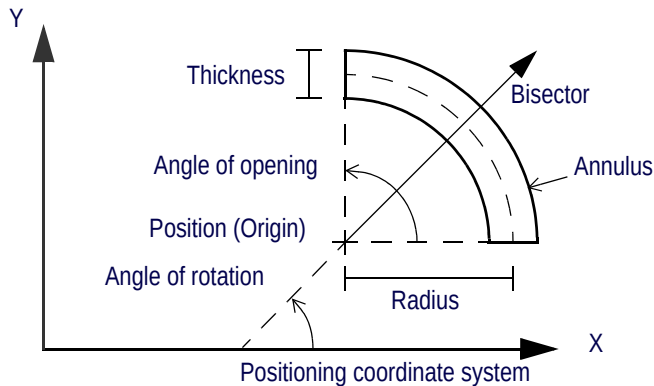


Figure 235 Inspection Tool Sector

Tool Coordinate System

The origin of the Sector determines the origin of the Tool coordinate system. This origin corresponds to the intersection of the two bounding radii shown as dotted lines in the HexSight interface. The Sector defines a polar coordinate system that can only be used to measure angles and radii. While radii are expressed with respect to the origin, angles are expressed with respect to the orientation of the bisector. Thus, for a given Sector, the radius values range from:

$$(\text{Radius} - (\text{Thickness}/2)) \text{ to } (\text{Radius} + (\text{Thickness}/2))$$

and angle values are defined between:

$$(- \text{Opening}/2) \text{ and } (\text{Opening}/2).$$

Sector Position

The Sector's position is specified as the X-Y coordinates of the Tool coordinate system origin.

The coordinate system and unit system used to express the Sector position depend on which tool positioning mode is configured. See the ***Tool Positioning Modes*** section for more details.

Opening

The Sector's opening is defined as the angle between its two bounding radii.

Thickness

The Sector's thickness corresponds to the radial distance between its two bounding annuli.

The coordinate system and unit system used to express the Sector thickness depend on which tool positioning mode is configured. See the [Tool Positioning Modes](#) section for more details.

Rotation

Rotation is the angle between the Sector's bisector and the X-Axis of the coordinate system used for tool positioning. This coordinate system depends on the positioning mode that is configured. See the [Tool Positioning Modes](#) section for more details.

Radius

The Sector's radius corresponds to the radius of its median annulus.

The coordinate system and unit system used to express the Sector radius depend on which tool positioning mode is configured. See the [Tool Positioning Modes](#) section for more details.

Tool Positioning Modes

HexSight Inspection tools provide four positioning modes that can be useful, depending on the application. These positioning modes are:

- [Image-Based Uncalibrated Mode](#)
- [Image-Based Calibrated Mode](#)
- [Model-Based Manual Mode](#)

- [Model-Based Automatic Mode](#)

The following sections describe the basic concepts of these modes, along with the coordinate system and unit used to express tool positioning.

Image-Based Uncalibrated Mode

This is the most basic tool positioning mode. It is typically used to take uncalibrated measures on specific parts of an image. When this mode is used in an inspection application, it usually requires the inspected object to be somewhat fixtured.

- **Positioning coordinate system:** In the Image-based Uncalibrated mode, the tool is positioned with respect to the Image coordinate system as shown in [Figure 236](#).

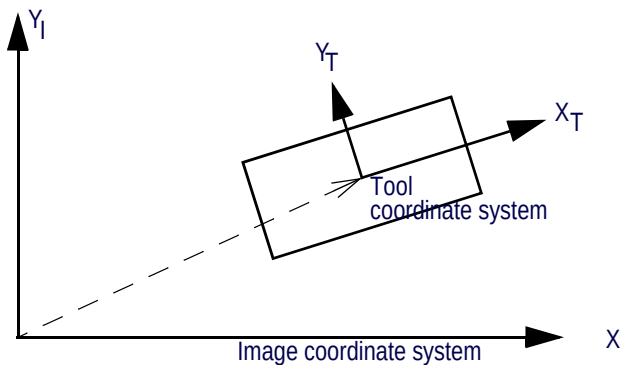


Figure 236 Image-Based Uncalibrated Mode: Relationship between Tool and Image Coordinate Systems

- **Units used for positioning:** Since the Image coordinate system is used to position the tool, X and Y positions as well as Rectangle width and height must be expressed in pixels. Rotation and skew can be expressed either in degrees or in radians depending on the units selected with the **Orientation Units** combo box.
- **Results:** In this positioning mode, inspection results can only be returned with respect to the Tool coordinate system and the Image

coordinate system. Dimensions and positions are always expressed in pixels in the Image-based Uncalibrated mode.

Image-Based Calibrated Mode

This mode is similar to the uncalibrated positioning mode, except that the World coordinate system is used to position the tool instead of the Image coordinate system. This mode is typically used to take calibrated measures on specific parts of an image. In this positioning mode, the tool is fixed with respect to the World coordinate system; part fixturing is therefore usually required.

- **Positioning coordinate system:** In the Image-based Calibrated mode, the tool is positioned with respect to the World coordinate system as shown in [Figure 237](#).

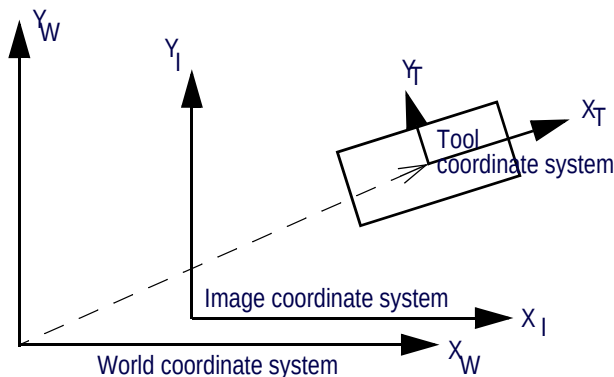


Figure 237 Image-Based Calibrated Mode: Relationship between Tool and World Coordinate Systems

- **Units used for positioning:** As its name implies, the Image-based Calibrated mode uses calibrated units to express the Rectangle's X and Y positions, as well as its width and height. The actual calibrated units used for these dimensions are set with the **Length Units** combo box in the General panel. Rotation and skew can be expressed either in degrees or in radians, depending on the units selected with the **Orientation Units** combo box.

- **Results:** In the Image-based Calibrated positioning mode, inspection results can be returned in one of the following coordinate systems:

The Tool coordinate system

The World coordinate system

The Image coordinate system

Dimensions and positions are expressed in calibrated units in the World and Image coordinate systems while pixels are used in the Image coordinate system.

Model-Based Manual Mode

When building an inspection application in which the parts to be inspected are not fixtured, HexSight Inspection tools provide a model-based mode in which the tool is positioned on a Model Image. A transform can then be applied to the tool to fit detected instances of the object in the Image. In the Model-based Manual mode, the Scale, Rotation and Translation components of the transform must be provided by the vision application using the Inspection tool. In addition to an image, you must provide a Model onto which the tool will be positioned.

- **Positioning coordinate system:** The Model-based mode is designed for the inspection of randomly oriented parts. The coordinate system used to position the tool is the Object coordinate system defined in the Model, as shown in [Figure 238](#).

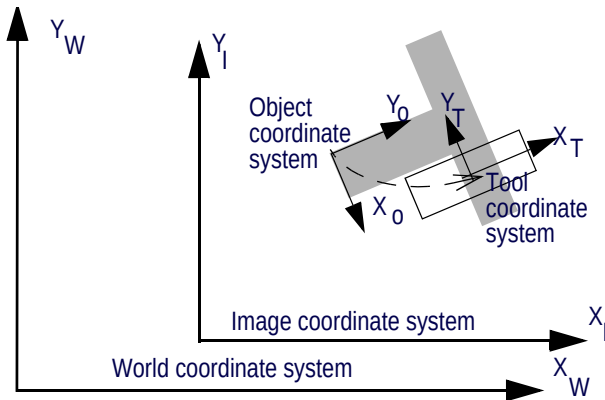


Figure 238 Model-Based Mode: Relationship between Tool and Object Coordinate Systems

- Units used for positioning:** Since reliable object location can only be carried out with calibrated images, Model-based positioning uses calibrated units to express the Rectangle's X and Y positions as well as its width and height. The actual calibrated units used for these dimensions are set with the Length Units combo box in the General panel. Rotation and skew can be expressed either in degrees or radians, depending on the units selected with the **Orientation Units** combo box.

Tool Transform Steps in Model-Based Positioning

With Model-based positioning, the Rectangle is placed with respect to the Object coordinate system defined in the Model. To fit the inspection tool to an instance of the object in the image, the transform provides four (4) parameters: Scale, Rotation, Translation X and Translation Y. The transform components are applied in the following order:

1. Scale Factor
2. Rotation
3. Translation

The following figures illustrate how the transform components are used to position the Rectangle on an object instance.

Tool Position Before a Transform

[Figure 239](#) shows that the Tool coordinate system is positioned and oriented with respect to the Object coordinate system.

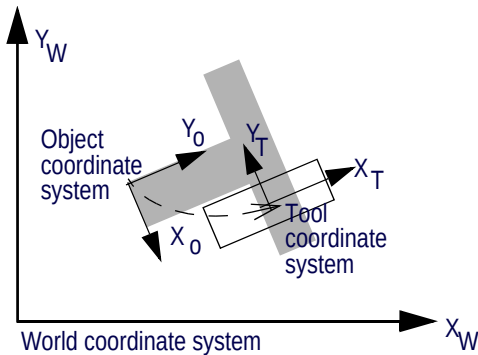


Figure 239 Model Rectangle Positioning

Before the transform is applied, the Rectangle is positioned as if the origin of the Object coordinate system was moved to the origin of the World coordinate system, with a scale factor of **1** and no rotation. See [Figure 240](#).

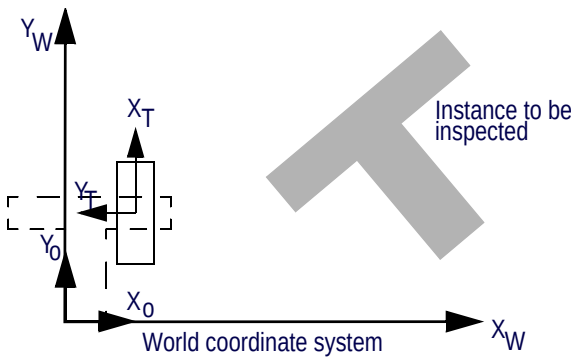


Figure 240 Tool Positioning before Transform is Applied

First Transform Step

The Rectangle is scaled to match the scale factor of the instance to be inspected. See [Figure 241](#).

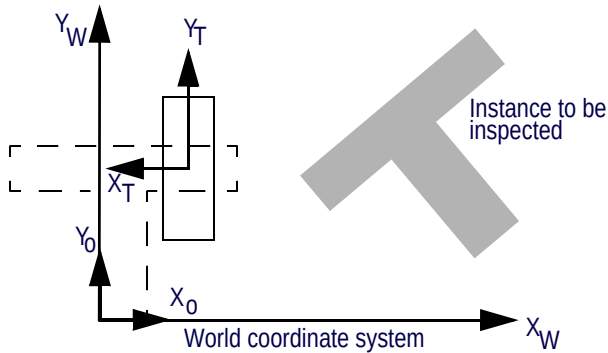


Figure 241 Tool Positioning after Scale is Applied

Second Transform Step

The Rectangle is rigidly attached to the Object coordinate system, which is rotated to match the orientation of the instance to be inspected. See [Figure 242](#).

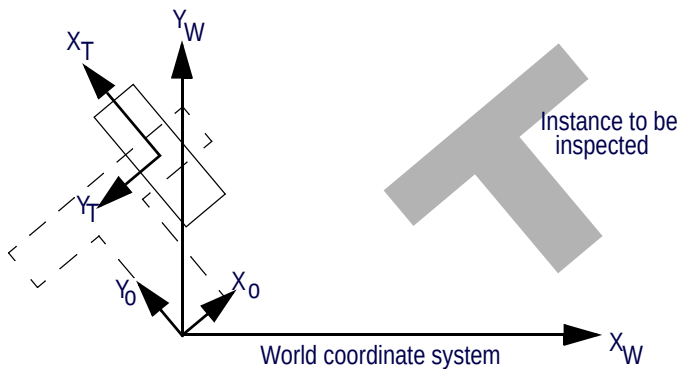


Figure 242 Tool Positioning after Rotation is Applied

Third And Last Transform Step

The Object coordinate system is translated to fit the instance to be inspected. See [Figure 243](#).

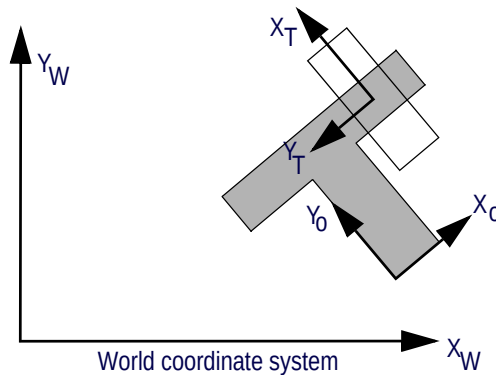


Figure 243 Tool Positioning after Translation is Applied

The transform concept can be very useful in inspection applications where the objects are randomly placed. Instead of repositioning the tool in the World coordinate system according to the pose of the object by modifying the Rectangle properties, the pose information can be input directly. This is particularly useful when the Rectangle is offset with respect to the Object coordinate system origin, in which case this offset must be taken into consideration when applying a rotation to position the tool in the World coordinate system. In the Model-based Manual mode, the components of the transform must be provided by the application.

Results in Model-based Manual Mode

In the Model-based Manual positioning mode, inspection results can be returned in one of the following coordinate systems:

- The Tool coordinate system
- The Object coordinate system
- The World coordinate system
- Image coordinate system

Position and dimensions in the Tool, Object and World coordinate systems are expressed in calibrated units, while pixels are used in the Image coordinate system.

- The **Tool coordinate system** is mostly useful for applications in which the position of a feature must be compared to an expected position. In such applications, you position the Rectangle in a manner that the origin of the Object coordinate system coincides with the expected position. Positions returned in the Tool coordinate system then correspond directly to the offset from the expected position.
- The **Object coordinate system** is typically used to measure the position of features with respect to a specific point on an object. In such applications, you define the Object coordinate system on the Model to make its origin coincide with the point from which positions have to be measured. Positions returned in the Object coordinate system directly correspond to the required measurements.
- The **World coordinate system** is used in applications where the absolute position of features must be known. Positions returned in the World coordinate system can be used to directly control actuators.
- The **Image coordinate system** can directly return the position of features in the Image. It is mostly useful for image processing applications.

Model-Based Automatic Mode

The Model-based Automatic positioning mode is conceptually the same as the Manual mode. The only difference is that transform components are automatically applied from an object instance located by the Locator instead of being provided manually. This mode greatly reduces the amount of code needed to carry out the inspection task.

- In addition to an image and a Model, you must provide an Instance Scene output by the Locator. The instance to be inspected is selected using the Instance selection Index from text box in the Transform panel. The transform components for this instance are shown but are read only.

- This mode is ideal for applications that require inspection of randomly oriented parts. A Model is used by the Locator to find and locate instances. An inspection tool positioned on the same Model can then be transposed automatically by specifying a selection index in the Instance Scene created by the Locator.

Positioning Mode Summary

The following table summarizes the similarities and differences between the different tool positioning modes.

Table 19 Positioning Modes Summary

	Image-based		Model-based	
	Uncalibrated	Calibrated	Manual	Automatic
Apply transform to a Model Defined Location	Disabled	Disabled	Enabled	Enabled
Lock Location Transform on a Selected Instance	N/A	N/A	Disabled	Enabled
Express Position, Size and Sampling Step using Calibrated Units	Disabled	Enabled	N/A	N/A
Positioning coordinate system	Image	World	Object	Object
Units used for positioning	pixels	Calibrated Units	Calibrated Units	Calibrated Units
Units used to express Results in the Image coordinate system	pixels	pixels	pixels	pixels
Units used to express Results in the World coordinate system	N/A	Calibrated Units	Calibrated Units	Calibrated Units
Units used to express Results in the Object coordinate system	N/A	N/A	Calibrated Units	Calibrated Units

Table 19 Positioning Modes Summary

	Image-based		Model-based	
	Uncalibrated	Calibrated	Manual	Automatic
Units used to express Results in the Tool coordinate system	pixels	Calibrated Units	Calibrated Units	Calibrated Units
Input Image	Required	Required	Required	Required
Input Model Scene	Not used	Not used	Required	Required
Input Instance Scene	Not used	Not used	Not used	Required

Sampling

Sampling refers to the procedure used by Inspection tools for gathering values within the portion of an image bounded by the Rectangle. Two sampling parameters, the sampling step and bilinear interpolation, can be used to establish a proper tradeoff between speed and precision.

Bilinear Interpolation

The Rectangle's axes are rarely aligned with the grid of pixels that constitute the Image, especially in Model-based modes. Without interpolation, any given pixel within the Rectangle is assigned the value of the Image pixel closest to the sampled pixel's center. This results in jaggedness and loss of precision. The bilinear interpolation function smoothes out the jaggedness within the sampled image by attributing each pixel a value interpolated from grey levels of neighboring pixels.

Detail from grey-scale Image with and without interpolation



Without bilinear interpolation



With bilinear interpolation

Figure 244 Bilinear Interpolation

When subpixel precision is required in an inspection application, bilinear interpolation should always be enabled for the sampling process. For applications where the speed requirements are more critical than precision, non-interpolated sampling can be used.

Sampling Step

In the Rectangle-sampled Image, all pixels are the same size and are square. The sampling step defines the height and width in calibrated units of each of the Rectangle's pixels.

- A default sampling step is computed by the inspection tool, based on the average size, in calibrated units, of a pixel in the Image. This default sampling step is usually recommended.
- For specific applications where a more appropriate tradeoff between speed and precision must be established, the sampling step can be modified by enabling the **Custom** check box in the Location panel and by modifying the sampling step in the text box.
- Increasing the sampling step value reduces the tool's precision and decreases the execution time.
- Reducing the sampling step can increase the tool's precision but can also increase the execution time. Undersampling can be useful in applications where an approximate measure is sufficient.

Using Inspection Tools

Inspection tools have a similar interface, or properties page, that is composed of 5 tab-selected panels. For the initial setup and configuration of an inspection tool, you will access these panels in order, from left to right.

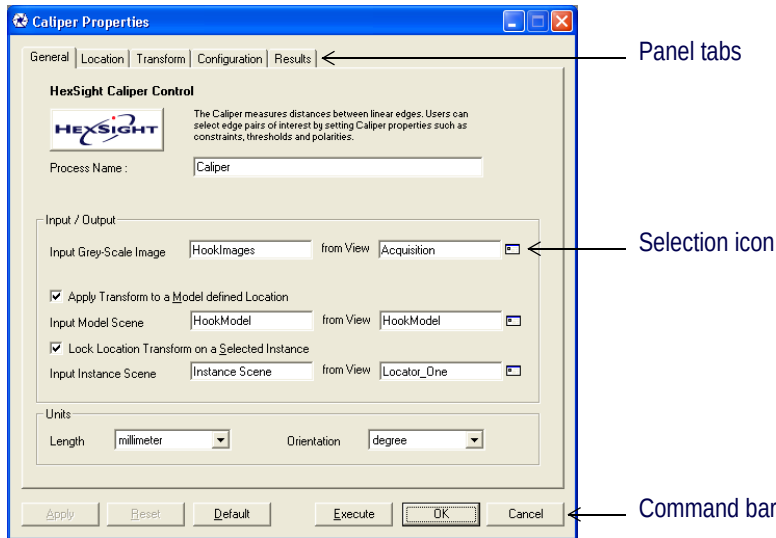


Figure 245 Inspection Tools — Panel Interface

The standard command buttons at the bottom of the panels are the same as with other HexSight tools.

Basic Method

Preliminary Step: Execute the sequence of processes in the application's Process Manager so that all tools have generated the necessary Views and Images in the runtime database.

1. Instantiate and name the inspection tool in the same manner as all other HexSight tools.
2. Enter the Input and Output Views and Images.
3. Enable the required positioning options.

4. Place the tool's Rectangle or Sector on the Image.
5. Configure the tool's settings.
6. Execute the tool process
7. Review the results then readjust settings as required.

General Panel

Inputs and outputs are set in the interface's General panel. You will also use this panel to configure a positioning mode.

Inputs

An Input Image is always required. Depending on the positioning mode, additional required inputs will include a **Model Scene** and an **Instance Scene**. The selection icon lets you rapidly select Input Views, Scenes and Images from a list of compatible items in the runtime database.

Positioning Mode

Image-based or Model-based positioning is configured with the **Apply Transform to a Model Defined Location** and **Lock Location Transform on a Selected Instance** check boxes in the General panel. Refer to the Tool positioning section for more details on the usage of Image and Model-based positioning modes.

In the Image-based mode, only an Input Image is required. In the Model-based Manual mode, an Input Model Scene is also required. To use the tool in the Model-based Automatic mode, an Input Instance Scene must also be provided, in addition to the Input Image and Input Model Scene.

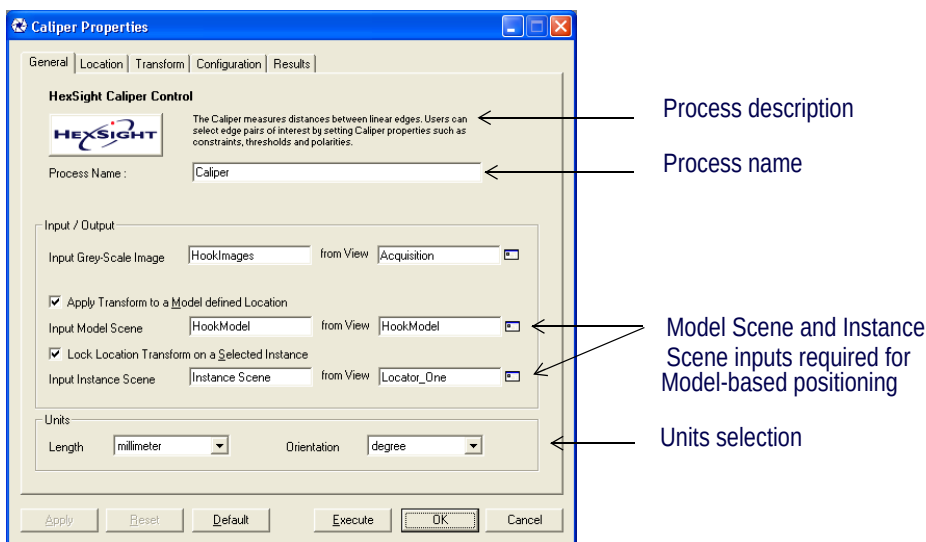


Figure 246 Inspection Tools — General Panel

Inspection Tool Units

The Units frame is used to select the Calibrated units, used to position the tool as well as to express the results computed by the inspection tool in the Image-based Calibrated positioning mode and in Model-based modes.

The **Length** combo box sets the units used to express dimensions and positions. The **Orientation** combo box sets the units used to express angles. Inspection tools incorporate unit conversion capabilities, which means that results can be output using any of the supported units, even though the Image is not calibrated with the same units.

Note:When you modify the units with either the **Length** or the **Orientation** combo boxes, changes are applied immediately. All changes made to parameters since the last clicking of the **Apply** button will be lost if one of the **Units** combo boxes is modified.

Location Panel

This panel presents a similar graphical interface for all Inspection tools. It provides an Image display, called the Location Display, used for placing a tool upon a specific area of the object. When the tool is Model-based, the Location display shows the Image of the Model, otherwise it shows the Input Image. Zoom features allow you to inspect the Image in fine detail for an optimal placement of the tool.

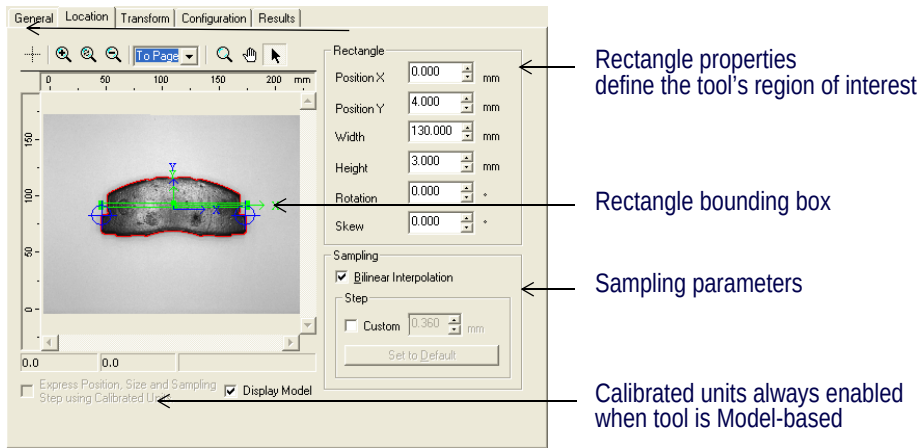


Figure 247 Inspection Tools — Location Panel

Location Display

Use the Location display to position a tool over the appropriate area of an object. Right-click in the display for zoom options as needed, or use the buttons in the tool bar. Refer to the Using HexSight Displays section of this guide for more details on using the display.

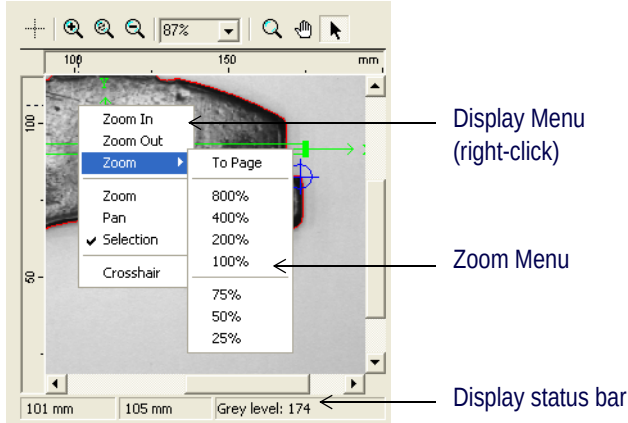


Figure 248 Inspection Tools — Location Display

The top and left sidebars show the X-Y coordinates. Coordinates are always expressed in calibrated units in the Image-based Calibrated and Model-based modes. In the Image-based Uncalibrated mode, pixels units are used instead of calibrated units.

The status bar at the bottom of the display shows the X-Y coordinates of the cursor at any given position, as well as additional information such as pixel values.

Location Display of a Model-Based Tool

- Shows the Image of the Model.
- X-Y coordinates are those of the World coordinate system, in calibrated units.
- Position, Rotation and Skew are defined with respect to the Object coordinate system.

Location Display of an Image-Based Tool

- Shows the Input Image.

- X-Y coordinates are those of the Image coordinate system or the World coordinate system, depending on the setting of the Calibrated Units check box.
- Position, Rotation and Skew are defined with respect to the World or Image coordinate systems, depending on the setting of the Calibrated Units check box.

Placing Rectangle within an Image

The Rectangle has eight side and corner handles that can be dragged with the mouse. The Rectangle's center point corresponds to the origin of the Tool coordinate system. This point also corresponds to the position of the Rectangle in the selected positioning coordinate system.

The following guidelines will help you to properly place the Rectangle used by HexSight Inspection tools.

- Adjust the position of the Rectangle by clicking and dragging either its center point or its border. You can also set and refine this position with the **Position X** and **Position Y** text boxes.
- Adjust the Width and Height of the Rectangle by clicking and dragging one of the resize handles. You can also set dimensions directly in the **Width** and **Height** text boxes. *The Rectangle is always resized with respect to its center point.*
- Set the angle of rotation by clicking and dragging the X-Axis arrow of the Tool coordinate system. You can also enter rotation values directly in the **Rotation** text box.
- Set the angle of skew for the Rectangle by clicking and dragging the Y-Axis arrow of the Tool coordinate system. You can also enter rotation values directly in the **Skew** text box.

Bilinear Interpolation

Bilinear interpolation is enabled by default and is recommended for applications requiring subpixel precision.

Sampling Step

The default sampling step is recommended for most applications. The sampling step corresponds to the width and height of one pixel in the sampled Rectangle, in calibrated units.

- To modify the default value, enable the **Custom** check box and enter the new value in the text box.
- To restore the custom sampling step to its default value for the current Image, click **Set to Default**.
- To return to the automatic sampling step computation mode, disable the **Custom** check box.
- To return to the default value, click the **Set to Default** command button.

Calibrated Units Check Box

Model-based tools always use calibrated units. Image-based tools, by default, use pixels as the length unit. To express dimensions and positions in calibrated units, for Image-based tools, you should enable the **Calibrated Units** check box.

Transform Panel

This panel is only enabled when using Model-based positioning modes. It contains transform components used to fit the inspection tool to object instances. The display shows the Input Image.

[Figure 249](#) illustrates the difference between the displays in the Transform panel and the Location panel.

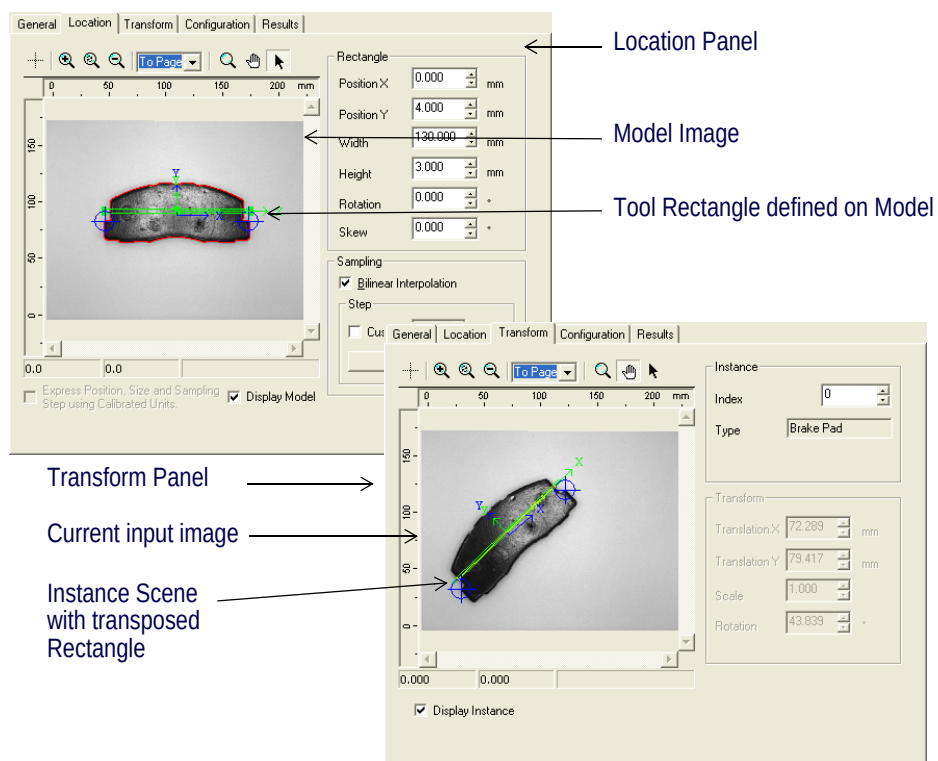


Figure 249 Location vs. Transform Display — Model-Based Tools

In the Manual mode, you must enter the components of the transform (Scale, Rotation and Translation) directly in the text boxes.

In the Automatic mode, the transform components in the text boxes are read-only. Select the instance to be inspected with the Index text box in the Instance frame.

The **Type** text box shows the name of the instance to which this instance corresponds. Graphic markers are also overlaid to the Image to illustrate the Object coordinate system transposed to fit the instance

Configuration Panel

The Configuration panel is specific to each inspection tool. It is used to configure the algorithm used for carrying out the inspection task.

Results Panel

The Results panel is specific to each Inspection tool. It shows the results of the inspection that was carried out by the tool.

Results can be expressed with respect to the Tool coordinate system, the Object coordinate system, the World coordinate system or the Image coordinate system.

Units used to express calibrated results are set with the Units combo boxes in the General panel.

Caliper

The Caliper tool finds and locates one or more edge pairs that meet user-defined constraints on a region of interest and then measures distances between the two edges within each pair. Edge pairs and their individual edges can be scored according to user-defined criteria.

Caliper Chapter Outline

[Understanding Caliper Concepts](#)

[Tool Position](#)

[Edge Detection](#)

[Edge Scoring](#)

[Results](#)

[Using the Caliper](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

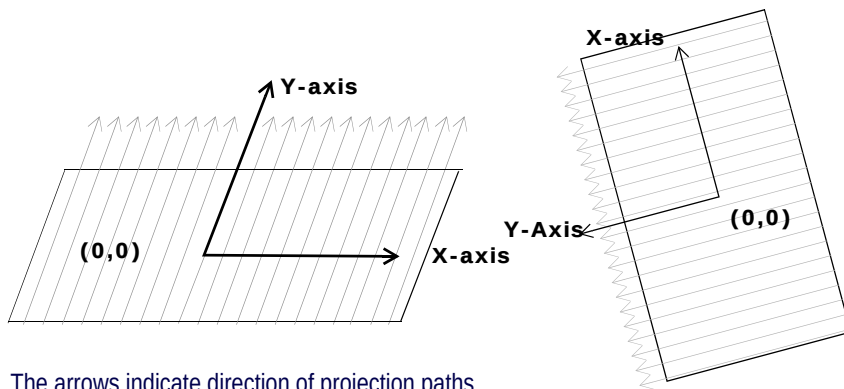
Understanding Caliper Concepts

The Caliper tool finds, locates and measures the gap between one or more edge pairs on an object within a bounded area called a Rectangle. The Caliper uses pixel values within the Rectangle to build projections needed for edge detection.

Once potential edges have been located, the Caliper applies the constraints that have been configured independently for each edge pair to determine which edge pairs are valid. The Caliper scores and measures valid edge pairs before displaying the results. The Caliper algorithm is presented in four sections: **Tool Position**, **Edge Detection**, **Edge Scoring** and **Results**.

Tool Position

The area processed by a tool, such as the Edge Locator, is bounded by a Rectangle such as the one illustrated in [Figure 250](#).



The arrows indicate direction of projection paths.

Figure 250 Projection Paths

The tool's Rectangle is characterized by its width, height, rotation, and skew. The position of the tool corresponds to the (X,Y) coordinates of its origin.

Edge Detection

Edges represent contours of an object, or physical boundaries between areas on an object. Edge enhancement projections, based on changes in pixel values, enable the Caliper to carry out an efficient edge detection.

Projection Path

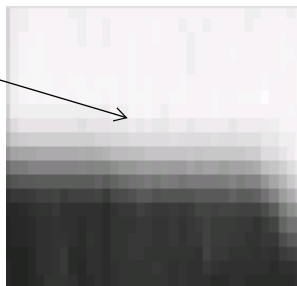
The Caliper processes data within the selection Rectangle along pixel-wide lines, called projection paths, parallel to the Rectangle's Y-axis (See [Figure 250](#)). The average pixel value for each projection path is stored in a one-dimensional signal that can be used in the edge location process.

Projection Image

The Caliper uses pixel values within each projection path to build a projection image, as shown in [Figure 251](#). This projection image is interpreted as a projection curve in which potential edges are indicated as strong positive or negative changes in greylevels.

Grey-Scale Image:

grid of individual pixels



Projection Image:

average grey-scale value attributed to each projection path

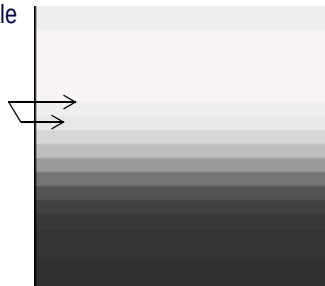


Figure 251 Image vs. Projection Image

As opposed to the grey-scale Image, which displays each pixel's individual greylevel value, the projection image displays the average greylevel value for each projection path within the physical area bounded by the Rectangle. This minimizes variations in pixel values caused by non-edge features or noise.

Projection Curve

The projection curve is a graphical representation of the projection image within the Caliper's Rectangle.

Sharp transitions in the curve intersect with potential edges: the steeper the rise or fall in a curve, the stronger the edge.

This curve also indicates changes in light values which characterize the polarity of the edge. Polarity is particularly important when setting constraints for the detection of valid edges.

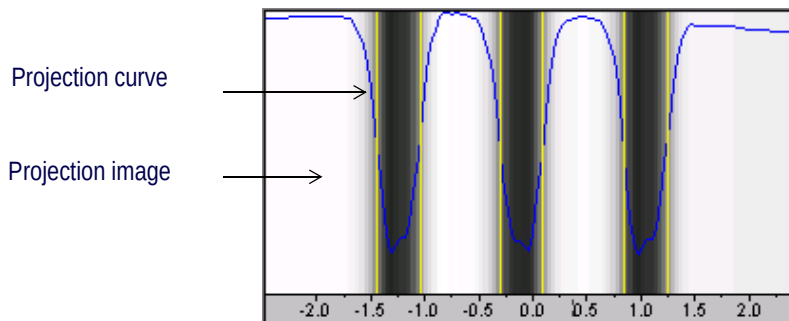


Figure 252 Projection Curve

Magnitude Curve

The magnitude curve is the first derivative of the projection curve. For a 256 greylevel image, it may range between -255 and 255. Positive and negative peaks in the curve indicate potential edges. Sharp peaks indicate strong, well-defined edges whereas dull peaks may indicate noise or poorly-defined edges.

Magnitude Threshold

The magnitude threshold is expressed as an absolute value. Peaks in the magnitude curve that are above and below the magnitude thresholds indicate potential edges.

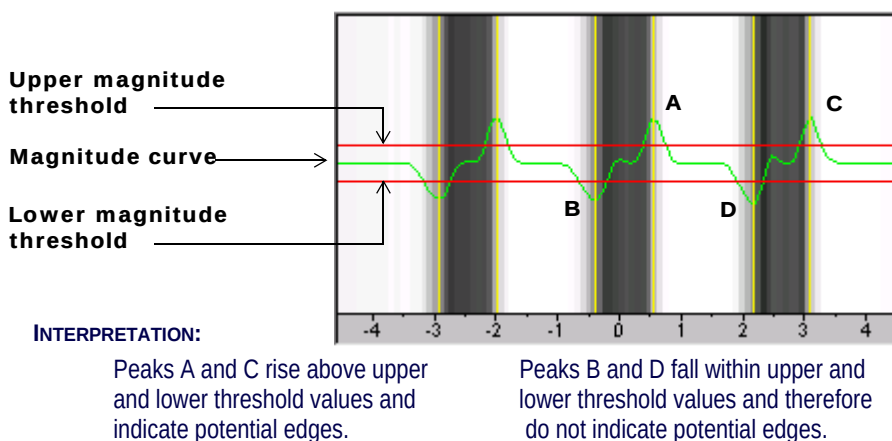


Figure 253 Magnitude Threshold

Filter

The filtering process attenuates peaks in the magnitude curve that are caused by noise. The filter half-width value corresponds to the half-width of the convolution kernel used to estimate the first derivative of the projection curve. It should be approximately equivalent to the width of the edge, in pixels, as it appears in the Image.

Edge Scoring

The Caliper scores potential edges according to the constraints set for edges, within the Rectangle. The scoring method restricts the Caliper's search so that only results for valid edge pairs are returned.

There are two basic types of constraints: edge position and edge magnitude. You can set the Caliper to use only one constraint type or both. A graphical function editor is provided for viewing and setting each type of constraint for each edge within a pair.

Magnitude Constraint

The magnitude constraint is based on edge values relative to the magnitude threshold. Edges having a magnitude equal to, or exceeding the magnitude

threshold, are attributed a score of **1**. Edges with values below the magnitude threshold receive a score ranging from **0** to **0.999**, according to a manually set magnitude constraint function.

The magnitude constraint can be applied to the first and second edges. [Figure 254](#) shows two different setups for a magnitude constraint function.

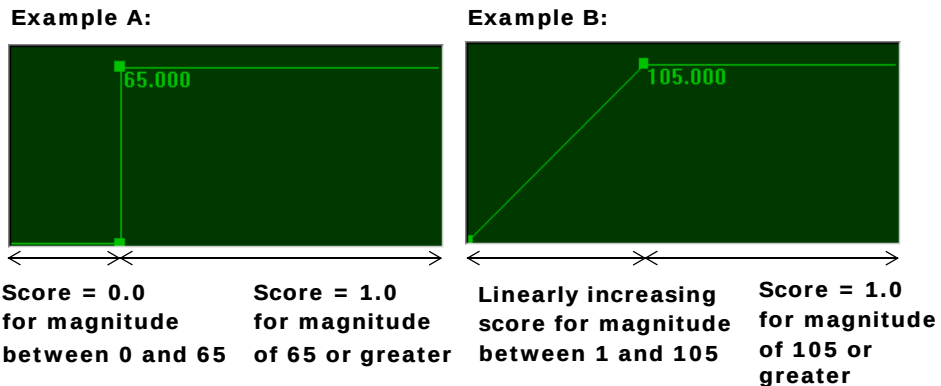


Figure 254 Magnitude Constraint Function

Position Constraint

Position constraints designate the physical location of an edge within the Rectangle image. This restricts the Caliper's search for an edge to a specific zone of an image.

- It is possible to graphically set a position constraint function in the projection image when the approximate position of an edge is known beforehand. This is useful for scoring an edge based on its offset from the expected position.

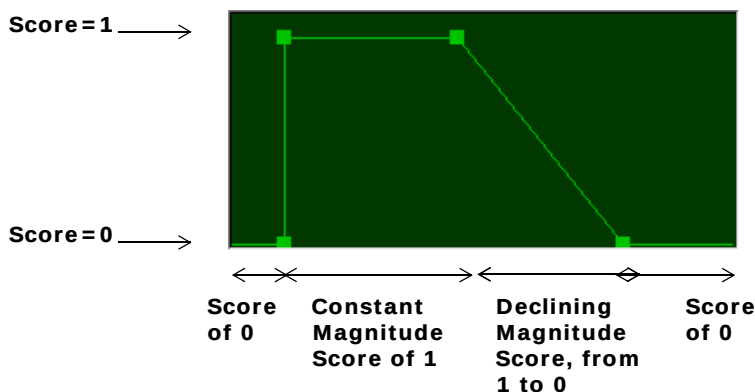


Figure 255 Position Constraint Function

Score Threshold

Scores attributed by the Caliper for constraints range from **0** to **1**. If both position and magnitude constraints are enabled, each constraint accounts for 50% of the total edge score. A user-defined threshold is applied to the score to only retain the strongest edges.

Polarity

Polarity is not taken into account for scoring edges but is particularly important when setting constraints for the detection of valid edge pairs. The Caliper applies this constraint before applying scored constraints. Edges are classified by their polarity, defined in the Caliper interface as either **Light to Dark** or **Dark to Light**, as it changes from left to right in the tool Rectangle. See [Figure 256](#).

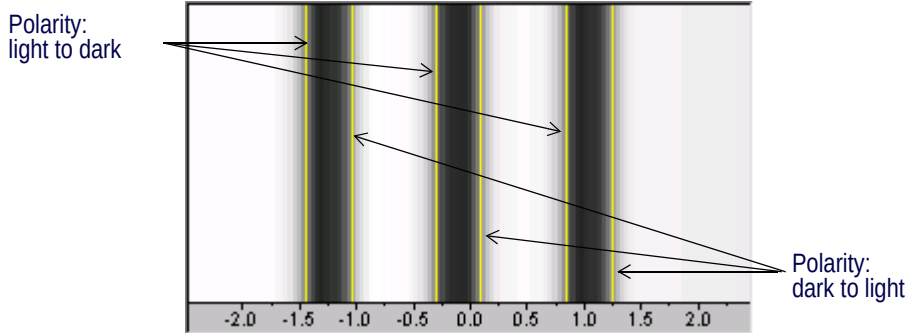


Figure 256 Polarity

Results

Caliper results are read only properties shown for each instance of a valid edge pair found and located. Some of the edge properties do not depend on a positioning mode (see **Inspection Tools** chapter) or on a specific coordinate system.

- **Magnitude:** Magnitude of an edge expressed as its corresponding peak value in the magnitude curve.
- **Magnitude Score:** Score between 0 and 1, calculated according to an edge magnitude function.
- **Position Score:** Score between 0 and 1, calculated according to an edge position function.
- **Score:** A score is returned for each edge as well as for each edge pair. For the edge pair score, each edge accounts for 50% of the score.
- **Size:** The actual Caliper measure. This measure is the length of the segment that connects the center point of each edge segment.

Some properties depend on the positioning mode or the coordinate system. These properties include the X and Y positions of an edge, the orientation (Rotation) of an edge and the distance between the Tool origin and the position of the edge.

All results appear in runtime for each current instance. Consult the ***Reference Manual*** for more information on using Caliper results.

Using the Caliper

The Caliper setup is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Caliper follow the left to right order of these panels.

For each panel you will find pointers to guide you through the setup.

General Panel

Use this panel to name the instance for the Caliper and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affects the required inputs.

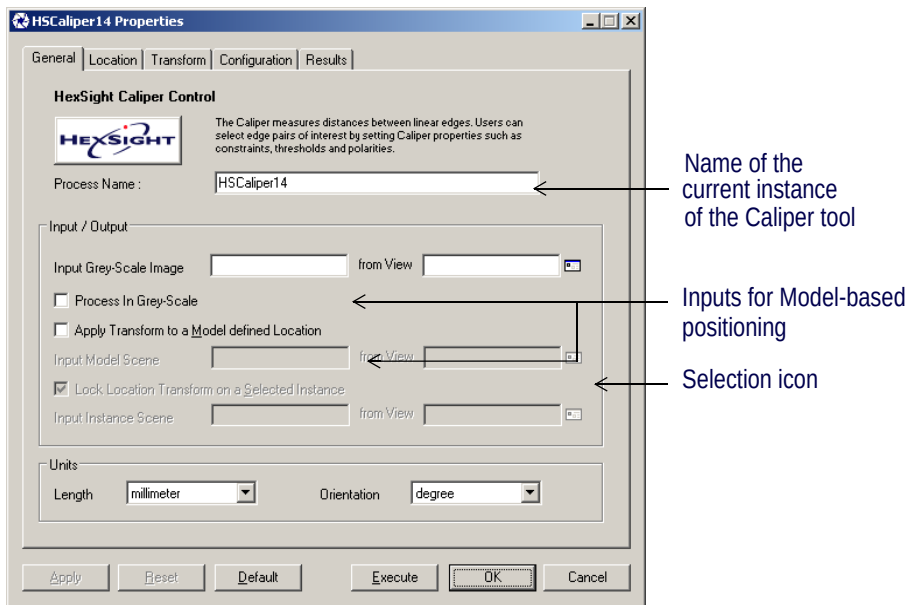


Figure 257 Caliper — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based

automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 20 Positioning Modes: Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 20 Positioning Modes: Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSCaliper#, where # is replaced by its rank number in the list of instantiated processes.

Inputs

Image

The **Image** field is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool.

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process is applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

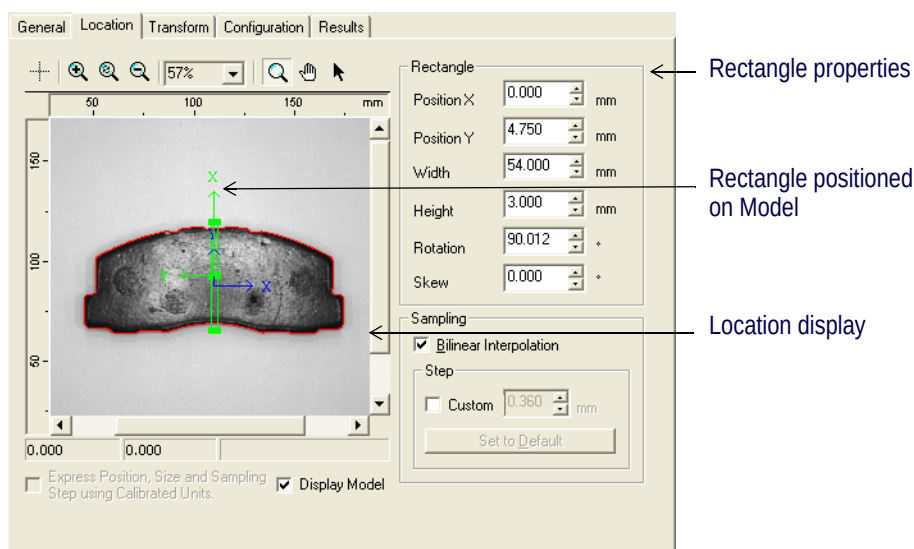


Figure 258 Caliper — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height**, **Rotation**, and **Skew**. See the ***Inspection Tools*** chapter of this guide for more details on the Rectangle.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation and Skew** are always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing the Rectangle

You can use the same Caliper tool to measure several edge pairs within the Rectangle PROVIDING they are parallel to each other.

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.

- The Caliper detects object edges that are parallel to its Y-axis: adjust the rotation to best match the inclination of the edges you want to measure.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.
- Skew by dragging the tip of the Y-axis Tool coordinate system.

Note: You must adjust the Rectangle so that the projection image exceeds the edges you need to locate by a number of pixels greater than the value of filter half-width.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that is applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

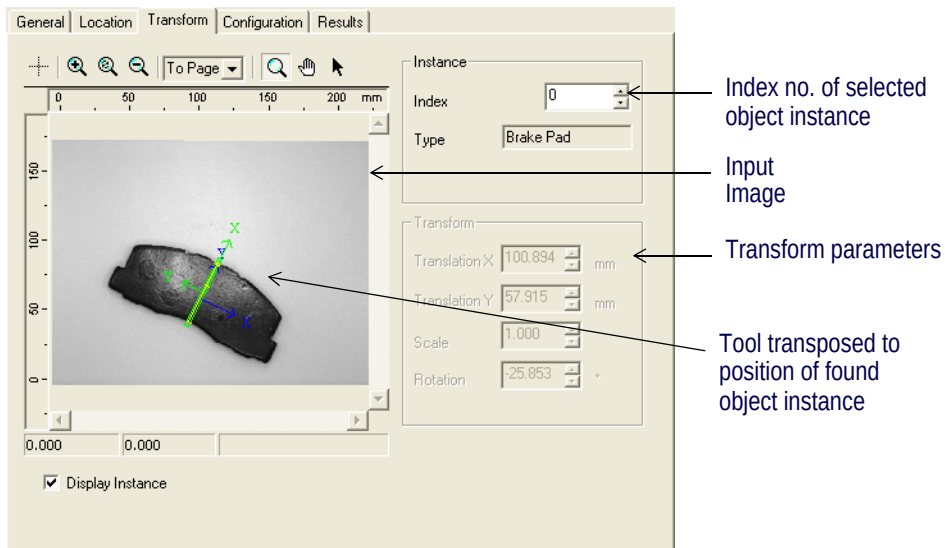


Figure 259 Caliper — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters. Located edges are shown as yellow lines within the Rectangle.

Display Instance

Enable the **Display Instance** check box to activate a visual representation of the object instance selected in the **Index** text box. This representation of the object is provided by the Instance Scene generated by the Locator. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled, the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure how edges are detected in the Rectangle. The Configuration panel is used to configure edge detection parameters as well as to set polarity and scoring constraints based on position and magnitude.

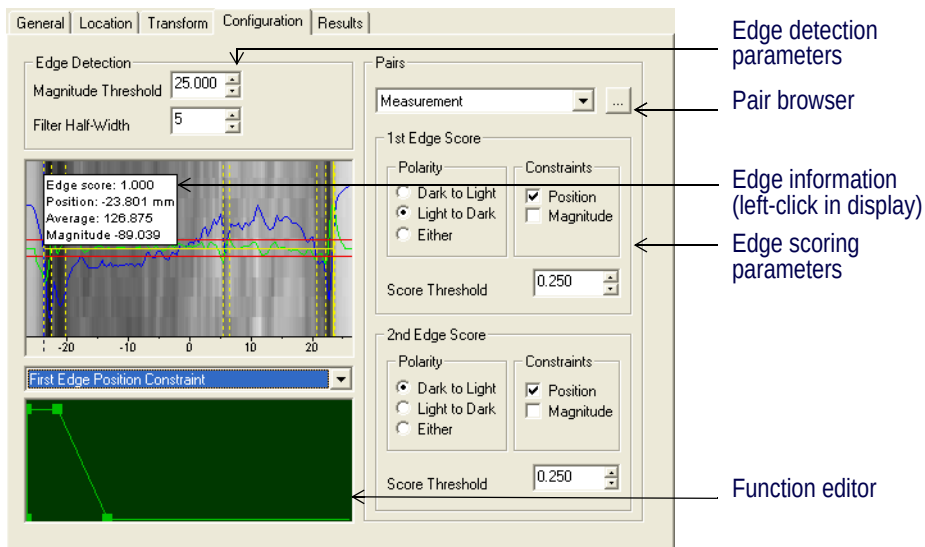


Figure 260 Caliper — Configuration Panel

Rectangle Display

Use the Rectangle display to visualize the configuration and the results of the edge detection process. By default, this display shows the Image sampled from the region of interest onto which are overlaid the projection and edge magnitude curves, the detection thresholds and the detected edges.

Display Options

Left-clicking on the Rectangle display opens a pop-up window displaying the projection value, edge magnitude, position within the region of interest, and score at the actual cursor position.

Right-clicking on the Rectangle display opens a context menu of the following display options.

- **Image** displays the Image sampled from the region of interest bounded by the Rectangle and mapped to a rectangle. The Image is useful for visualizing edges with respect to other features in the region of interest.
- **Projection Image** displays the projection Image used to detect edges in the region of interest. This projection is built by averaging each column of pixel in the sampled Image. Use the projection image to obtain a visual indication of the edge features in the region of interest.
- **Edges** displays the edges detected in the region of interest as vertical yellow lines.
- **Thresholds** displays the positive and negative thresholds used to detect peaks on the edge magnitude curve as two horizontal red lines.
- **Projection Curve** displays a plot of the projection as a blue curve. This one-dimensional signal is computed by averaging each column of pixel in the sampled Image. Edges occur at sharp transitions in the curve.
- **Magnitude Curve** displays a plot of the edge magnitude as a green curve. The edge magnitude signal is the first derivative of the projection curve. Edges in the region of interest appear as positive or negative peaks on this curve. Sharp peaks indicate the presence of well-defined edges. Dull peaks indicate weak edges, or variations caused by smooth transitions.

Edge Detection Parameters

Magnitude Threshold

Magnitude Threshold defines the threshold above which the absolute value of a peak on the edge magnitude curve is considered as a potential edge.

- All peaks on the edge magnitude curve that fall between the two red magnitude threshold lines are not considered.
- All peaks on the edge magnitude curve that are above the upper threshold or below the lower threshold are considered as potential edges.
- Left-click on a peak in the Rectangle display to view its magnitude in a pop-up window.
- Sometimes unwanted edges, such as those corresponding to shadows, have magnitude values CLOSE to those of valid edges. When possible, do NOT use the magnitude threshold to reject such edges because in some images, lighting variations may cause edges of interest to also fall below the threshold. Instead, use polarity and position constraints that are not affected by lighting variations from one image to another.

Filter Half-Width

Filter Half-Width defines the half-width of the filter used to compute the first derivative of the projection curve. When properly set, the filter enhances edge information. Peaks on the magnitude curve that correspond to noise can be minimized, while peaks corresponding to potential edges can be made sharper.

- Set the filter half-width to a value equivalent to the number of pixels which the greylevel transition associated with the edge to be detected spans on the projection curve.
- If filter half-width set is too small compared with the width of the greylevel transition, edges might not be detected due to very low peaks on the edge magnitude curve.

- If filter half-width is set too large, positioning accuracy of detected edges might be affected by neighboring edges.

Note: If you increase the filter half-width, return to the Location panel to make sure that Rectangle is still sufficiently oversized. You must adjust the Rectangle so that, in the projection image exceeds the edges that must be located by a number of pixels greater than the value of filter half-width.

Pairs

Unlike the Edge Locator, the Caliper looks only for the number of edge pairs that have been configured. By default, when a new Caliper is created, a single pair named *Pair0* is configured. To add a new pair or rename/delete an existing pair, access the **Pair Manager** dialog by clicking the (...) button.

- Click **Add Pair** to create a new pair.
- **Remove Pair** deletes the selected pair.
- Clicking **Rename Pair** prompts you for a new pair name.

Use the Pairs combo box to select the pair for which you want to configure the polarity and scoring functions.

- If the Caliper is configured to measure only one edge pair, it will only output results for one pair, even though other pairs within the Rectangle respect the constraints.
- If no scoring is set for the two edges of the pair, the Caliper scans the projection curve from left to right and uses the first two edges that meet the polarity criteria to build the pair.
- If position and/or magnitude scoring is selected, the Caliper uses the two edges with the highest score values to compute the properties of the pair.

Polarity

You can constrain the Caliper to build the pair only with edges of specific polarities using the first and second edge **Polarity** radio buttons.

- **Dark to Light** will only consider positive peaks on the edge magnitude curve. Positive peaks are caused by transitions, in the left to right direction, from darker to lighter greylevels on the projection curve.
- **Light to Dark** will only consider negative peaks on the edge magnitude curve.
- **Either** will consider any peak on the edge magnitude curve.

Scoring

You can configure the Caliper to individually score the potential first and second edges of the pair according to position and/or magnitude criteria.

When position and/or magnitude scoring is enabled for the first edge of the pair, the potential edge with the highest score according to the first edge scoring criteria is used as the first edge. Similarly, if position and/or magnitude scoring is enabled for the second edge of the pair, the potential edge with the highest score according to the second edge scoring criteria is used to complete the pair. The total score of an edge is computed as follows:

- Always 1 if **Position** and **Magnitude** check boxes are not enabled.
- Equal to position score if only the **Position** check box is enabled.
- Equal to magnitude score if only the **Magnitude** check box is enabled.
- Equal to the average of position and magnitude scores if both **Position** and **Magnitude** check boxes are enabled.

Position

You can use a position constraint function to score the potential first and/or second edges by enabling its associated **Position** check box. The position scoring function for the first edge of the pair is defined with the function editor when **First Edge Position Constraint** is selected in the function editor's selection combo box. Select **Second Edge Position Constraint** to define the position scoring function for the second edge of the pair.

- Drag the four handles in the function editor to define the position scoring function.

- The X-axis of the function editor corresponds to the X-axis used to plot the projection and edge magnitude curves.
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position is expressed as a fraction of the total width of the image or projection image displayed in the Rectangle display. So for instance if a handle is placed exactly at the center of the function editor, its X-position is 0.5.
- Y-position of the handles defines its associated score. The Y-position of the handles is fixed and is 0 for the two outer handles and 1 for the inner two.
- The position score for one edge is interpolated from the linear segment connecting the nearest handles on each side of this edge. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Magnitude

You can use a magnitude constraint function to score the potential first and/or second edges by enabling its associated **Magnitude** check box. The magnitude scoring function for the first edge of the pair is defined with the function editor when **First Edge Magnitude Constraint** is selected in the function editor's selection combo box. Select **Second Edge Magnitude Constraint** to define the magnitude scoring function for the second edge of the pair.

The magnitude constraint is typically used in conjunction with the position constraint, to refine edge detection or to eliminate a set of edges whose magnitude is below the threshold level.

- Drag the two handles in the function editor to define the position scoring function.
- The X-axis of the function editor corresponds to magnitude values ranging from 0 (no edge) to 255 (the highest magnitude).
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position corresponds directly to an edge

magnitude value on the X-axis. For instance, if a handle is placed exactly at the center of the function editor, its X-position corresponds to the edge magnitude 128.

- Y-position of the handles defines its associated score. The Y-position of the handles can be either 0 or 1.
- The magnitude score for one edge is interpolated from the linear segment connecting the nearest handles on each side of its magnitude. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Score Threshold

You can use **Score Threshold** to reject potential first and second edges with low score values. This threshold is applied to the total score of the edges and not on individual position and magnitude score components.

Results Panel

The Results panel provides a grid of results and a visual display of the results.

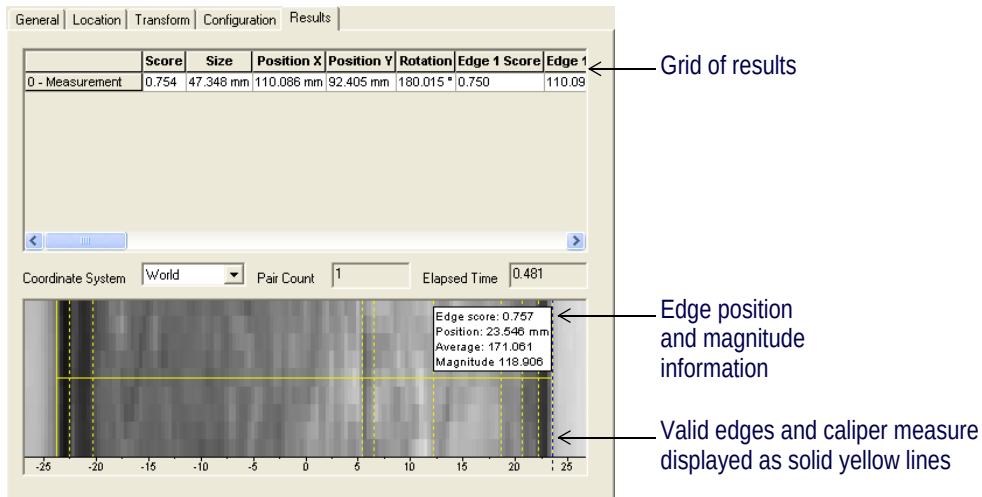


Figure 261 Caliper — Results Panel

Grid of Results

The grid of results displays properties for each edge pair and for each edge in an edge pair.

- **Score** shows the score for the edge pair.
- **Size** shows the actual Caliper measure of distance between the pair of edges.
- **Position X** and **Position Y** are the X and Y coordinates of the center points for each edge pair.
- **Rotation** is the angle of rotation angle for the edge pair.
- **Edge 1 / 2 Score** are the score for the first/second edges of the pair. If both position and magnitude constraints are used, each constraint accounts for 50% of the edge score.
- **Edge 1 / 2 Position X** and **Edge 1 / 2 Position Y** are the X and Y coordinates of the center points for the first/second edge segment of the pair.
- The **Edge 1 / 2 Rotation** are the angles of rotation the first/second edge of the pair.
- The **Position Score** is computed for both the first and second edge of each pair, according to the position constraint function.
- The **Magnitude**, for both the first and second edge of each pair, is the peak value of the edge in the magnitude curve.
- The **Magnitude Score** is computed for both the first and second edge of each pair, according to the magnitude constraint function.

Pair Count

The pair count indicates the number of pairs that have been set up within the **Configuration panel**. If an edge pair is not found, results for this edge pair appear as zero. The pair count is not affected.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: Image, World, Object, Tool. The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Results Display

To help visualize the located edges, the Results panel includes a visual display of the Sector.

- Right-click in the display to access the context menu for display options: Image, projection image, projection curve (blue), edge magnitude curve (green), threshold lines (red), and detected edges (yellow).
- Left-click on the display to open a pop-up window describing the position, projection value, edge magnitude and edge scores.

Elapsed Time

- **Elapsed Time** shows the total execution time of the Caliper. This includes sampling and projection, edge detection, validating, sorting and returning results.

Edge Locator

The Edge Locator finds and locates an edge or a set of edges that meet user-defined constraints. The results of located edges can be scored and sorted according to user-defined criteria.

Edge Locator Chapter Outline

[Understanding Edge Locator Concepts](#)

[Tool Position](#)

[Edge Detection](#)

[Edge Scoring](#)

[Results](#)

[Using the Edge Locator](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

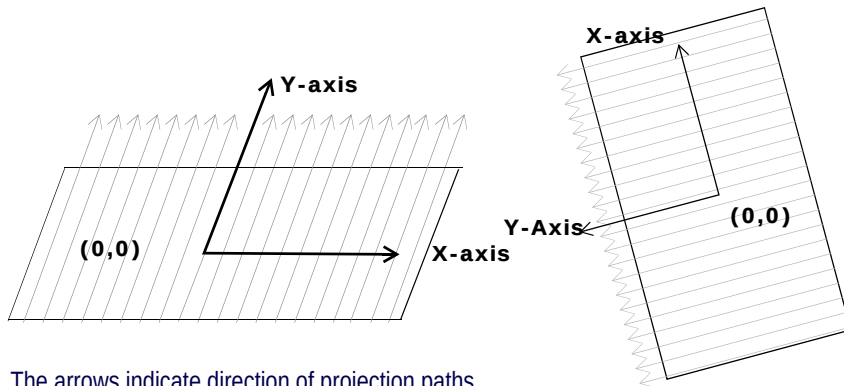
Understanding Edge Locator Concepts

The Edge Locator tool finds and locates one or more edges on an object within a bounded area called a Rectangle. The Edge Locator uses pixel greylevel values within the Rectangle to build projections, needed for edge detection.

Once potential edges have been located, the Edge Locator scores these edges according to user-defined criteria and then returns results for valid edges it has located. The Edge Locator process is presented in four sections: **Tool Position**, **Edge Detection**, **Edge Scoring** and **Results**.

Tool Position

The area processed by a tool, such as the Edge Locator, is bounded by a Rectangle such as the one illustrated in [Figure 262](#).



The arrows indicate direction of projection paths.

Figure 262 Projection Paths

The tool's Rectangle is characterized by its width, height, rotation, and skew. The position of the tool corresponds to the (X,Y) coordinates of its origin.

Edge Detection

Edges represent contours of an object, or physical boundaries between areas on an object. Edge enhancement projections, based on changes in greylevel values, enable the Edge Locator to carry out an efficient edge detection.

Projection Path

The Edge Locator processes data within the selection Rectangle along pixel-wide lines, called projection paths, parallel to the Rectangle's Y-axis. The average greylevel value for each projection path is stored in a one-dimensional signal that can be used in the edge location process.

Projection Image

The Edge Locator uses pixel values within each projection path to build a projection image, as shown in [Figure 263](#). This projection image is interpreted as a projection curve in which potential edges are indicated as strong positive or negative changes in greylevels.

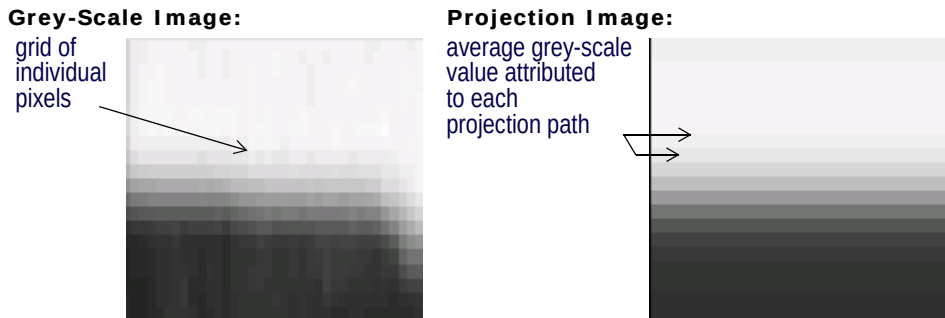


Figure 263 Image vs. Projection Image

As opposed to the grey-scale Image, which displays each pixel's individual greylevel value, the projection image displays the average greylevel value for each projection path within the physical area bounded by the Rectangle. This minimizes variations in pixel values caused by non-edge features or noise.

Projection Curve

The projection curve is a graphical representation of the projection image within the Edge Locator's *Rectangle*.

Sharp transitions in the curve intersect with potential edges; the steeper the rise or fall in a curve, the stronger the edge.

This curve also indicates changes in light values which characterize the polarity of the edge. Polarity is particularly important when setting constraints for the detection of valid edges.

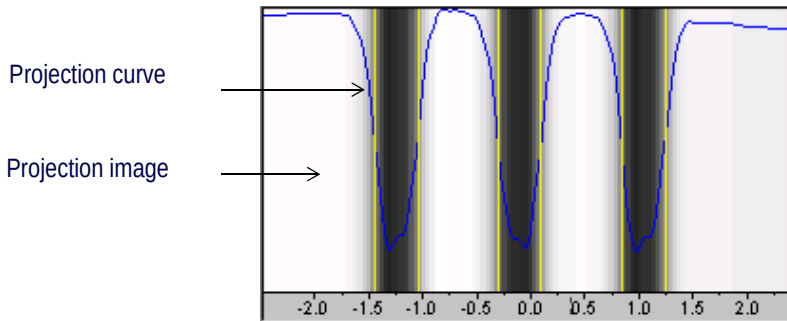


Figure 264 Projection Curve

Magnitude Curve

The magnitude curve is the first derivative of the projection curve. For a 256 greylevel image, it may range between -255 and 255. Positive and negative peaks in the curve indicate potential edges. Sharp peaks indicate strong, well-defined edges whereas dull peaks may indicate noise or poorly-defined edges.

Magnitude Threshold

The magnitude threshold is expressed as an absolute value. Peaks in the magnitude curve that are above and below the magnitude thresholds indicate potential edges.

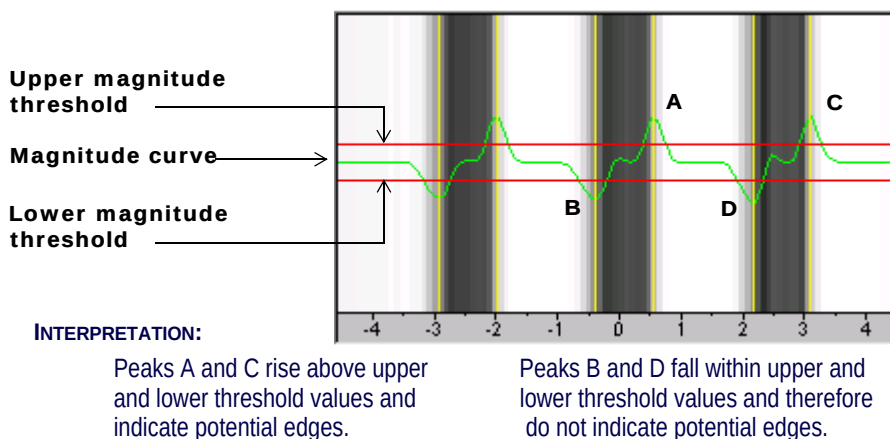


Figure 265 Magnitude Threshold

Filter

The filtering process attenuates peaks in the magnitude curve that are caused by noise. The filter *half-width* value corresponds to the half-width of the convolution kernel used to estimate the first derivative of the projection curve. It should be approximately equivalent to the width of the edge, in pixels, as it appears in the Image.

Edge Scoring

The Edge Locator scores potential edges according to the constraints set for the edge, or set of edges, within the Rectangle. The scoring method restricts the Edge Locator's search so that only results for valid edges are returned.

There are two basic types of constraints: edge position and edge magnitude. You can set the Edge Locator to use only one constraint type or both. A graphical function editor is provided for viewing and setting each type of constraint.

Magnitude Constraint

The *magnitude constraint* is based on edge values relative to the magnitude threshold. Edges having a magnitude equal to, or exceeding the magnitude

threshold, are attributed a score of **1**. Edges with values below the magnitude threshold receive a score ranging from **0** to **0.999**, according to a manually set *magnitude constraint function*.

[Figure 266](#) shows two different setups for a magnitude constraint function.

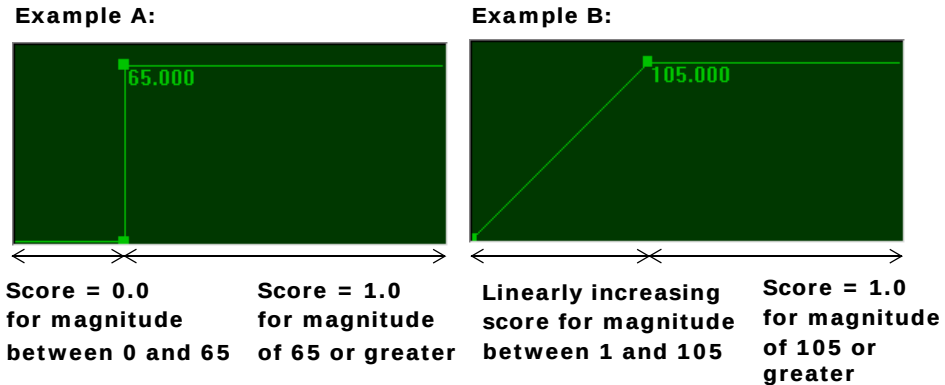


Figure 266 Magnitude Constraint Function

Position Constraint

Position constraints designate the physical location of an edge within the Rectangle image. This restricts the Edge Locator's search for an edge to a specific zone of an Image.

- It is possible to graphically set a position constraint function in the projection image when the approximate position of an edge is known beforehand. This is useful for scoring an edge based on its offset from the expected position.

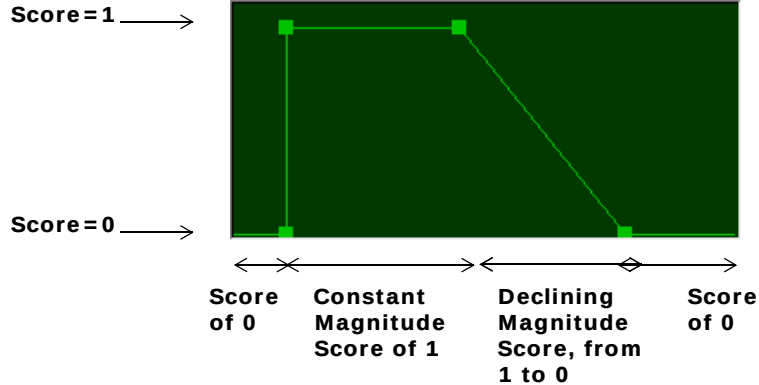


Figure 267 Position Constraint Function

Score Threshold

Scores attributed by the Edge Locator for constraints, range from **0** to **1**. If both position and magnitude constraints are enabled, each constraint accounts for 50% of the total edge score. A user-defined threshold is applied to the score to only retain the strongest edges. You can enable the Edge Locator to sort edge results according to the total score received for each edge.

Polarity

Polarity is not taken into account for scoring edges but it is particularly important when setting constraints for the detection of valid edges. The Edge Locator applies this constraint before applying scored constraints. Edges are classified by their polarity, defined in the interface as either **Light to Dark** or **Dark to Light**, as it changes from left to right in the tool Rectangle. See [Figure 268](#)

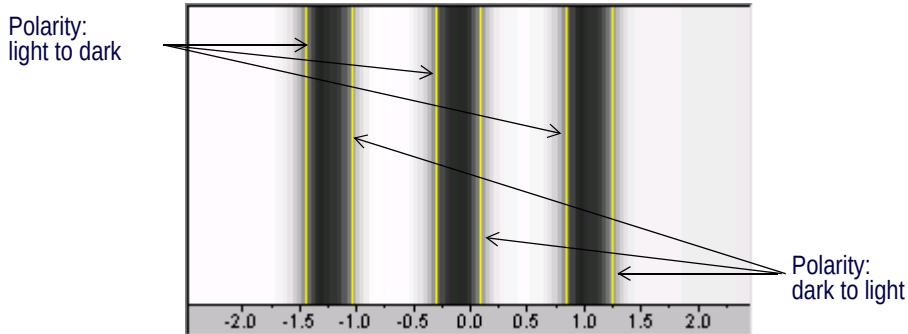


Figure 268 Polarity

Results

Edge Locator results are read only properties shown for each instance of a valid edge. Some of the edge properties do not depend on a positioning mode (see ***Inspection Tools*** chapter) or on a specific coordinate system. These properties are:

- **Magnitude:** Magnitude of the edge expressed as its corresponding peak value in the magnitude curve.
- **Magnitude Score:** Score between 0 and 1, calculated according to the edge magnitude function.
- **Position Score:** Score between 0 and 1, calculated according to the edge position function.
- **Score:** The score for the edge.

Some properties depend on the positioning mode or the coordinate system. These properties include the X and Y positions of an edge, the orientation (Rotation) of an edge and the distance between the Tool origin and the position of the edge.

All results appear in runtime for each current instance. Consult the ***Reference Manual*** for more information on using Edge Locator results.

Using the Edge Locator

The Edge Locator setup is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Edge Locator follow the left to right order of these panels.

For each panel you will find pointers to guide you through the setup.

General Panel

Use this panel to name the instance of the Edge Locator and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affects the required inputs.

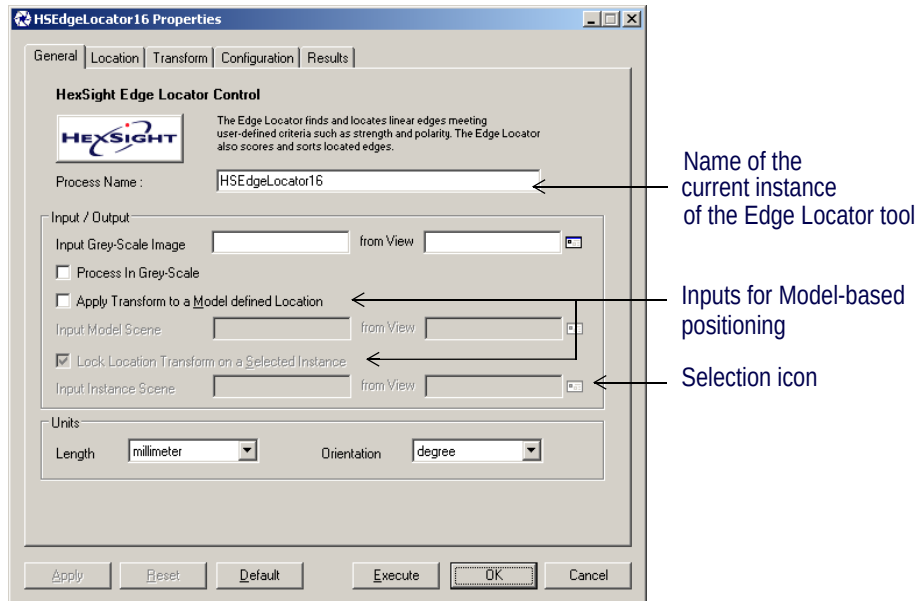


Figure 269 Edge Locator — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based

automatic. See the **Inspection Tools** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 21 Positioning Modes: Required Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 21 Positioning Modes: Required Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSEdgeLocator#, where # is replaced by its rank number in the list of instantiated processes.

Inputs

Image

The **Image** field is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool.

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's **Rectangle**, which defines the region of interest in the Image to which the process is applied.

Location Display

In addition to the Image used to position the tool, the Location display also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the Location display shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the Location display.

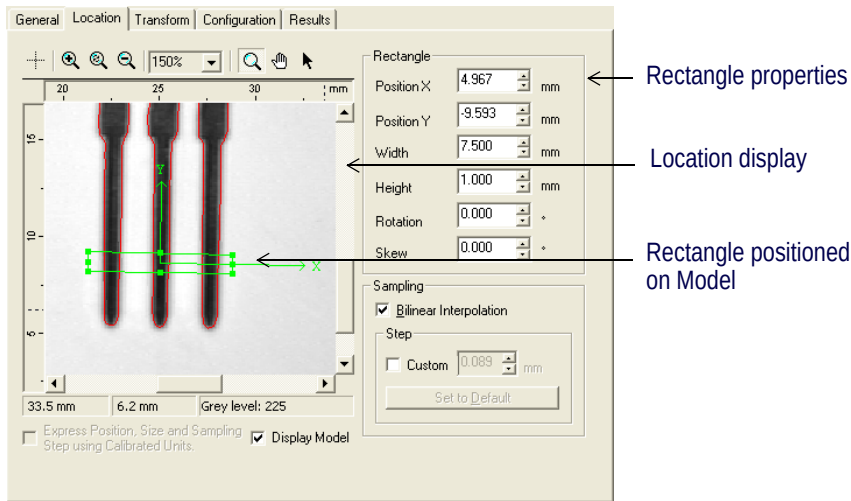


Figure 270 Edge Locator — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height**, **Rotation**, and **Skew**. See the ***Inspection Tools*** chapter of this guide for more details on the Rectangle.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation and Skew** are always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing the Rectangle

You can use the same Edge Locator tool to measure several edges within the Rectangle PROVIDING they are parallel to each other.

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- The Edge Locator detects object edges that are parallel to its Y-axis: adjust the rotation to best match the inclination of the edges you want to measure.

- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.
- Skew by dragging the tip of the Y-axis Tool coordinate system.

Note: You must adjust the Rectangle so that the projection image exceeds the edges you need to locate by a number of pixels greater than the value of filter half-width.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that is applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

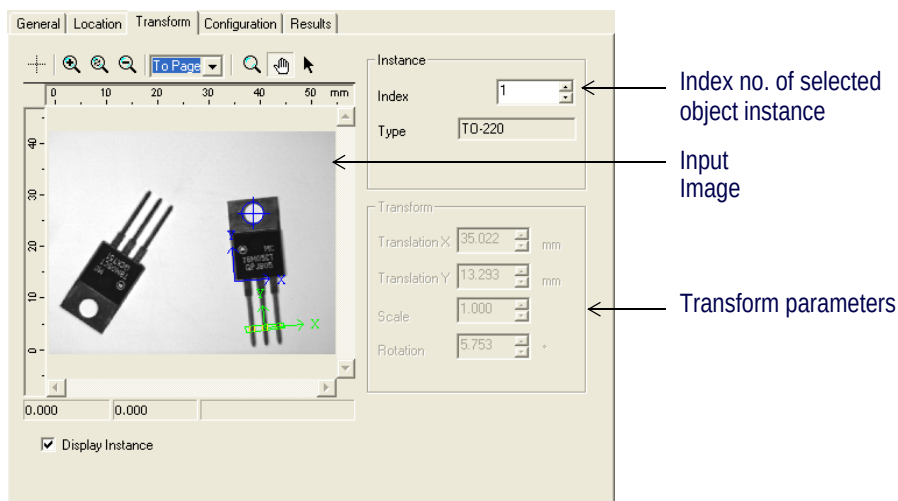


Figure 271 Edge Locator — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters. Located edges are shown as yellow lines within the Rectangle.

Display Instance

Enable the **Display Instance** check box to activate a visual representation of the instance selected in the **Index** text box. This representation of the object is provided by the Instance Scene generated by the Locator. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled, the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure how edges are detected in the Rectangle. The Configuration panel is used to configure edge detection parameters as well as to set polarity and scoring constraints based on position and magnitude.

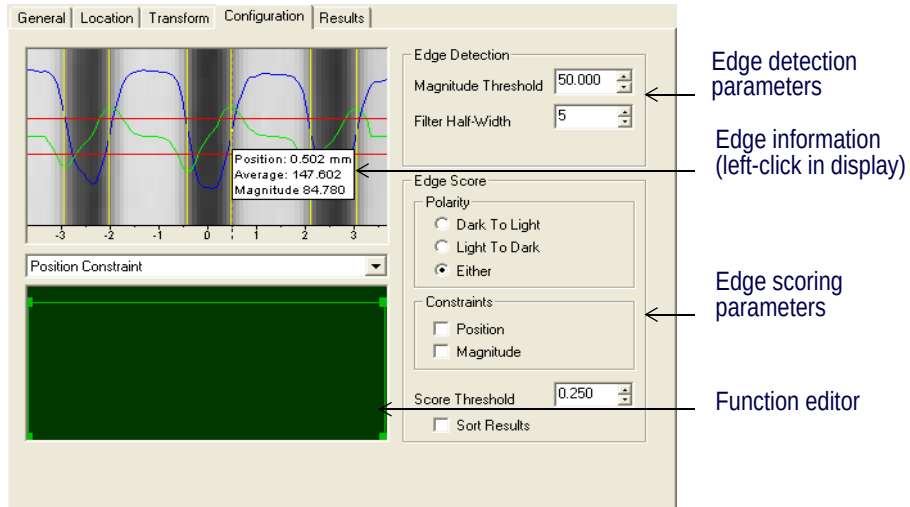


Figure 272 Edge Locator — Configuration Panel

Rectangle Display

Use the Rectangle display to visualize the configuration and the results of the edge detection process. By default, this display shows the Image sampled from the region of interest onto which are overlaid the projection and edge magnitude curves, the detection thresholds and the detected edges

Rectangle Display Options

Left-clicking on the Rectangle display opens a pop-up window displaying the projection value, edge magnitude, position within the region of interest, and score at the actual cursor position.

Right-clicking on the Rectangle display opens a context menu of the following display options.

- **Image** displays the Image sampled from the region of interest bounded by the Rectangle and mapped to a rectangle. The Image is useful for visualizing edges with respect to other features in the region of interest.
- **Projection Image** displays the projection Image used to detect edges in the region of interest. This projection is built by averaging each column of pixel in the sampled Image. Use the projection image to obtain a visual indication of the edge features in the region of interest.
- **Edges** displays the edges detected in the region of interest as vertical yellow lines.
- **Thresholds** displays the positive and negative thresholds used to detect peaks on the edge magnitude curve as two horizontal red lines.
- **Projection Curve** displays a plot of the projection as a blue curve. This one-dimensional signal is computed by averaging each column of pixel in the sampled Image. Edges occur at sharp transitions in the curve.
- **Magnitude Curve** displays a plot of the edge magnitude as a green curve. The edge magnitude signal is the first derivative of the projection curve. Edges in the region of interest appear as positive or negative peaks on this curve. Sharp peaks indicate the presence of well-defined edges. Dull peaks indicate weak edges, or variations caused by smooth transitions.

Edge Detection Parameters

Magnitude Threshold

Magnitude Threshold defines the threshold above which the absolute value of a peak on the edge magnitude curve is considered as a potential edge.

- All peaks on the edge magnitude curve that fall between the two red magnitude threshold lines are not considered.
- All peaks on the edge magnitude curve that are above the upper threshold or below the lower threshold are considered as potential edges.

- Left-click on a peak in the Rectangle display to view its magnitude in a pop-up window.
- Sometimes unwanted edges, such as those corresponding to shadows, have magnitude values CLOSE to those of valid edges. When possible, do NOT use the magnitude threshold to reject such edges because in some images, lighting variations may cause edges of interest to also fall below the threshold. Instead, use polarity and position constraints that are not affected by lighting variations from one image to another.

Filter Half-Width

Filter Half-Width defines the half-width of the filter used to compute the first derivative of the projection curve. When properly set, the filter enhances edge information. Peaks on the magnitude curve that correspond to noise can be minimized, while peaks corresponding to potential edges can be made sharper.

- Set the filter half-width to a value equivalent to the number of pixels which the greylevel transition associated with the edge to be detected spans on the projection curve.
- If filter half-width set is too small compared with the width of the greylevel transition, edges might not be detected due to very low peaks on the edge magnitude curve.
- If filter half-width is set too large, positioning accuracy of detected edges might be affected by neighboring edges.

Note: If you increase the filter half-width, return to the Location panel to make sure that Rectangle is still sufficiently oversized. You must adjust the Rectangle so that, in the projection image exceeds the edges that must be located by a number of pixels greater than the value of filter half-width.

Polarity

You can constrain the Edge Locator to accept only edges of specific polarities using the **Polarity** radio buttons.

- **Dark to Light** will only consider positive peaks on the edge magnitude curve. Positive peaks are caused by transitions, in the left to right direction, from darker to lighter greylevels on the projection curve.
- **Light to Dark** will only consider negative peaks on the edge magnitude curve.
- **Either** will consider any peak on the edge magnitude curve.

Scoring

You can configure the Edge Locator to score the potential edges according to position and/or magnitude criteria. The total score of an edge is computed as follows:

- Always 1 if **Position** and **Magnitude** check boxes are not enabled.
- Equal to position score if only the **Position** check box is enabled.
- Equal to magnitude score if only the **Magnitude** check box is enabled.
- Equal to the average of position and magnitude scores if both **Position** and **Magnitude** check boxes are enabled.

Position

You can use a position function editor to score the potential edges by enabling the **Position** check box. The position scoring function is defined with the function editor when **Position Constraint** is selected in the function editor's selection combo box.

- Drag the four handles in the function editor to define the position scoring function.
- The X-axis of the function editor corresponds to the X-axis used to plot the projection and edge magnitude curves.
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position is expressed as a fraction of the total width of the image or projection image displayed in the Rectangle display. So for instance if a handle is placed exactly at the center of the function editor, its X-position is 0.5.

- Y-position of the handles defines its associated score. The Y-position of the handles is fixed and is 0 for the two outer handles and 1 for the inner two.
- The position score for one edge is interpolated from the linear segment connecting the nearest handles on each side of this edge. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Magnitude

You can use a magnitude constraint function to score the potential edges by enabling the **Magnitude** check box. The magnitude scoring function is defined with the function editor when **Magnitude Constraint** is selected in the function editor's selection combo box.

The magnitude constraint is typically used in conjunction with the position constraint, to refine edge detection or to eliminate a set of edges whose magnitude is below the threshold level.

- Drag the two handles in the function editor to define the position scoring function.
- The X-axis of the function editor corresponds to magnitude values ranging from 0 (no edge) to 255 (the highest magnitude).
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position corresponds directly to an edge magnitude value on the X-axis. For instance, if a handle is placed exactly at the center of the function editor, its X-position corresponds to the edge magnitude 128.
- Y-position of the handles defines its associated score. The Y-position of the handles can be either 0 or 1.
- The magnitude score for one edge is interpolated from the linear segment connecting the nearest handles on each side of its magnitude. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Score Threshold

You can use **Score Threshold** to reject edges with low score values. This threshold is applied to the total score of the edges and not on individual position and magnitude score components.

Sort Results

You can enable the **Sort Results** check box to sort the located edges in descending order of score values. By default, **Sort Results** is not enabled and edges are output in the same left to right order as they appear on the projection curve.

Results Panel

The Results panel provides a grid of results and a visual display of these results.

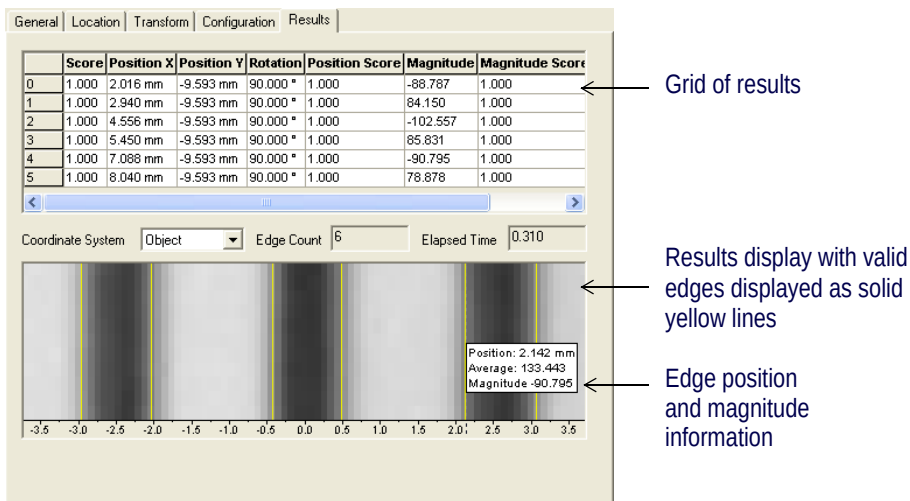


Figure 273 Edge Locator — Results Panel

Grid of Results

Every valid edge is numbered and assigned a set of properties that can be viewed in the grid of results.

- **Score** shows the score for the edge. If both position and magnitude constraints are used, each constraint accounts for 50% of the edge score.
- **Position X** is the X coordinate of the center point for each edge segment.
- **Position Y** is the Y coordinate of the center point for each edge segment,.
- **Rotation** shows the angle for the edge, in orientation units.
- The **Position Score** for the edge is computed according to the position constraint function.
- The **Magnitude** of the edge indicates its peak value in the magnitude curve.
- The **Magnitude Score** for the edge is computed according to the magnitude constraint function.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: Image, World, Object, Tool.

The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Results Display

To help visualize the located edges, the Results panel includes a visual display of the Sector.

- Use display's context menu (right-click) to display either the Image, mapped from the Sector or the projection image.
- Context menu options also include the projection curve (blue), the edge magnitude curve (green), the threshold lines (red), and the detected edges (yellow).

- Left-click on the display to open a pop-up window describing the position, projection value, edge magnitude and edge scores anywhere along the projection curve.

Elapsed Time

Elapsed Time shows the total execution time of the Edge Locator. This includes sampling and projection, edge detection, validating, sorting and returning results.

Edge Count

Edge Count indicates the number of valid edges that have been found within the Rectangle.

Arc Caliper

The Arc Caliper tool finds and locates one or more edge pairs that meet user-defined constraints on an arc shaped region of interest and then measures distances between the two edges within each pair. Edge pairs and their individual edges can be scored according to user-defined criteria.

Arc Caliper Chapter Outline

[Understanding Arc Caliper Concepts](#)

[Tool Position](#)

[Edge Detection](#)

[Edge Scoring](#)

[Results](#)

[Using the Arc Caliper](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Arc Caliper Concepts

The Arc Caliper tool finds, locates and measures the gap between one or more edge pairs on an object within a bounded area called a Sector. The Arc Caliper uses pixel greylevel values within the Sector to build projections, either annular or radial, needed for edge detection.

Once potential edges have been located, the Arc Caliper applies the constraints that have been configured independently for each individual edge and pair to determine which edge pairs are valid. The Arc Caliper scores and measures valid edge pairs before displaying the results. The Arc Caliper algorithm is presented in four sections: [Tool Position](#), [Edge Detection](#), [Edge Scoring](#) and [Results](#).

Tool Position

The area processed by an arc tool, such as the Arc Caliper, is bounded by a region of interest such as the one illustrated in [Figure 274](#).

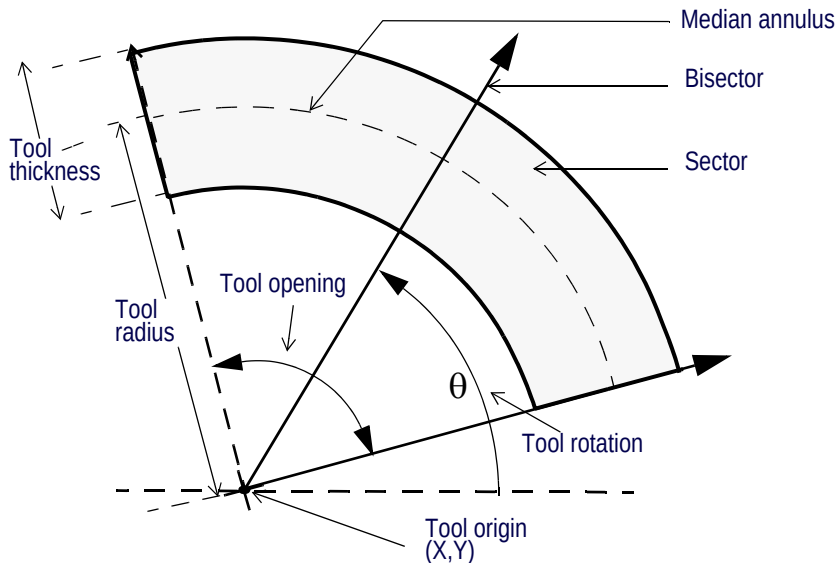


Figure 274 Arc Tool Sector

The tool's Sector is characterized by its opening, radius, thickness, position and orientation. The position of the tool corresponds to the (X,Y) coordinates of its origin. The tool orientation corresponds to the angle between a horizontal axis and the bisector.

Edge Detection

Edges represent contours of an object, or physical boundaries between areas on an object. Edge enhancement projections, based on changes in greylevel values, enable the Arc Caliper to carry out an efficient edge detection.

Projection Path

The Arc Caliper processes data within the Sector along pixel-wide lines, called projection paths. Projection paths can be either annular or radial. Annular projection is used to find edges that are aligned with the median annulus, such as arcs on concentric circles. Radial projection is used to find edges aligned along radial projections, much like the spokes of a wheel. These projection paths are illustrated in [Figure 275](#).

The average greylevel value for each projection path is stored in a one-dimensional signal that can be used in the edge location process.

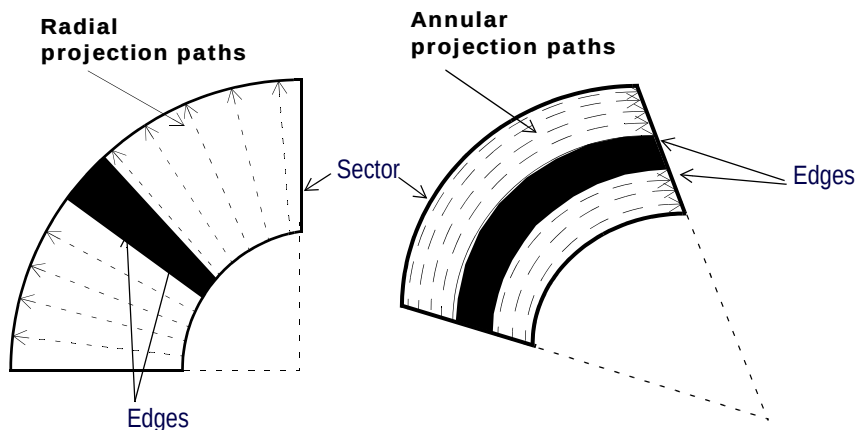


Figure 275 Radial vs. Annular Projection Paths

Projection Image

The Arc Caliper uses pixel values within each projection path to build a projection image, as shown in [Figure 276](#). This is a mapping from a Sector to a rectangle. Whereas for radial projection, the X-axis of the rectangle depicts an angular step, it represents an annular step (in length units) in the case of annular projection. The projection image is interpreted as a projection curve in which potential edges are indicated as strong positive or negative changes in greylevels.

Input Grey-Scale Image

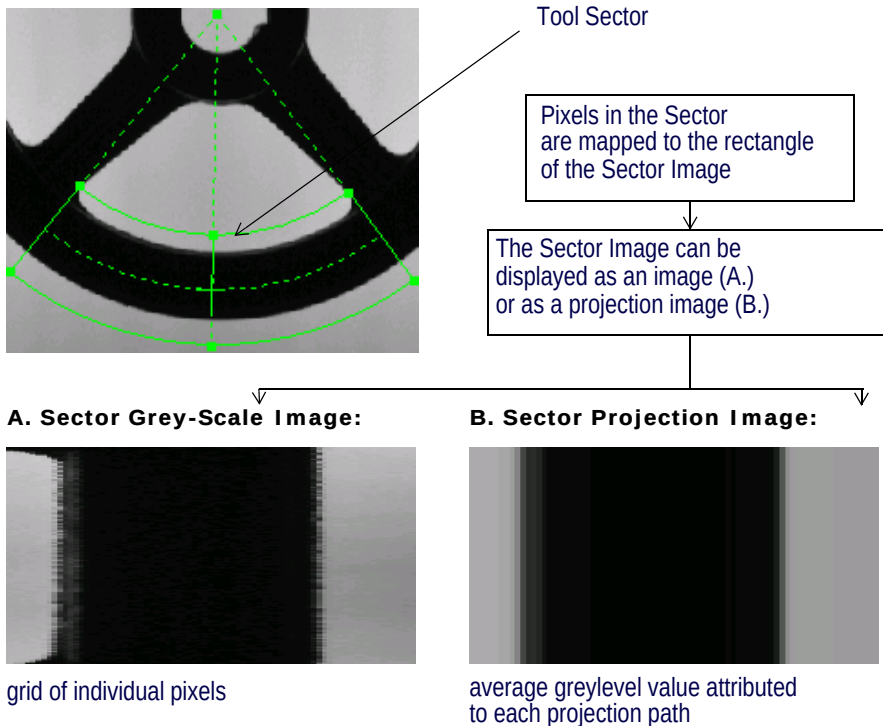


Figure 276 Image vs. Projection Image

As opposed to the Image, which displays each pixel's individual greylevel value, the projection image displays the average greylevel value for each projection path within the physical area bounded by the Sector. This minimizes variations in pixel values caused by non-edge features or noise.

Projection Curve

The projection curve is a graphical representation of the projection image within the Arc Caliper's Sector.

Sharp transitions in the curve intersect with potential edges; the steeper the rise or fall in a curve, the stronger the edge.

This curve also indicates changes in light values which characterize the polarity of the edge. Polarity is particularly important when setting constraints for the detection of valid edges.

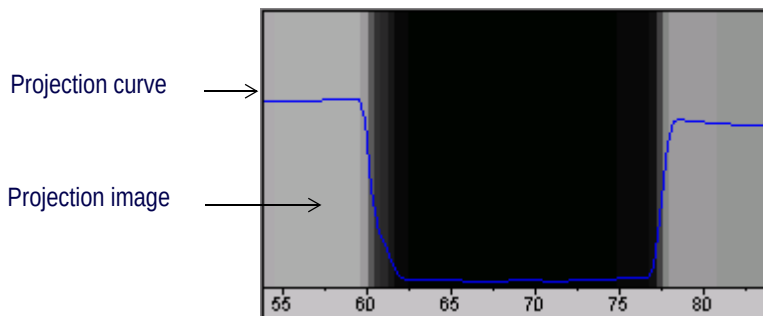


Figure 277 Projection Curve

Magnitude Curve

The magnitude curve is the first derivative of the projection curve. For a 256 greylevel image, it may range between -255 and 255. Positive and negative peaks in the curve indicate potential edges. Sharp peaks indicate strong, well-defined edges whereas dull peaks may indicate noise or poorly-defined edges.

Magnitude Threshold

The magnitude threshold indicates an absolute value, above which peaks in the magnitude curve indicate potential edges.

Filter

The filtering process attenuates peaks in the magnitude curve that are caused by noise. The filter half-width value corresponds to the half-width of the convolution kernel used to estimate the first derivative of the projection

curve. It should be approximately equivalent to the width of the edge, in pixels, as it appears in the Image.

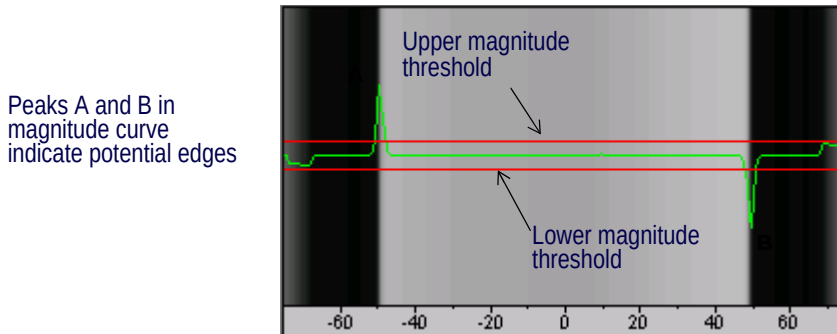


Figure 278 Magnitude Threshold

Edge Scoring

The Arc Caliper scores potential edges according to the constraints set for edges, within the Sector. The scoring method restricts the Caliper's search so that only results for valid edge pairs are returned.

There are two basic types of constraints: edge position and edge magnitude. You can set the Arc Caliper to use only one constraint type or both. A graphical function editor is provided for viewing and setting each type of constraint for each edge within a pair.

Magnitude Constraint

The magnitude constraint is based on edge values relative to the magnitude threshold. Edges having a magnitude equal to, or exceeding the magnitude threshold, are attributed a score of **1**. Edges with values below the magnitude threshold receive a score ranging from **0** to **0.999**, according to a manually set magnitude constraint function.

The magnitude constraint can be applied to the first and second edges. [Figure 279](#) shows two different setups for a magnitude constraint function.

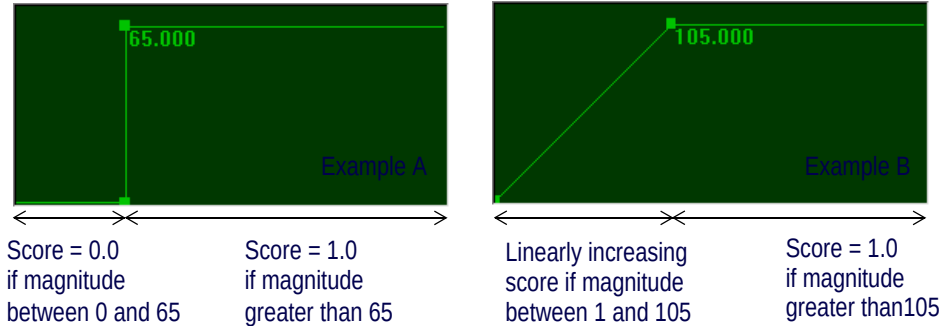


Figure 279 Magnitude Constraint Function

Position Constraint

Position constraints designate the physical location of an edge within the Sector Image. This restricts the Arc Caliper's search for an edge to a specific zone of an Image.

- It is possible to graphically set a position constraint function in the projection image when the approximate position of an edge is known beforehand. This is useful for scoring an edge based on its offset from the expected position.

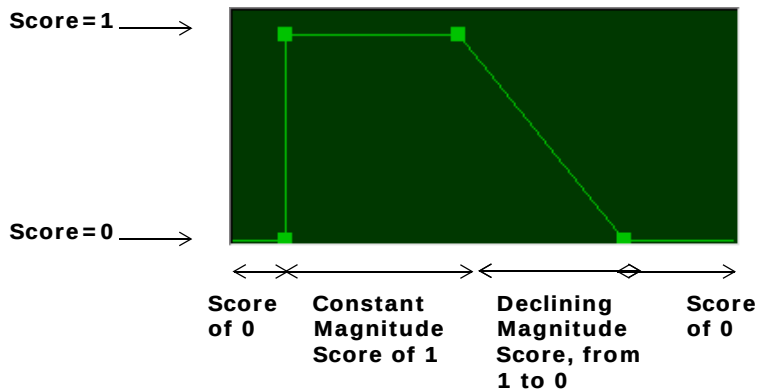


Figure 280 Position Constraint Function

Score Threshold

Scores attributed by the Arc Caliper for constraints range from **0** to **1**. If both position and magnitude constraints are enabled, each constraint accounts for 50% of the total edge score. A user-defined threshold is applied to the score to only retain the strongest edges.

Polarity

Polarity is not taken into account for scoring edges but is particularly important when setting constraints for the detection of valid edge pairs. The Arc Caliper applies this constraint before applying scored constraints. Edges are classified by their polarity, defined in the interface as either **Light to Dark** or **Dark to Light**, as it changes from left to right in the tool sector. See [Figure 281](#).

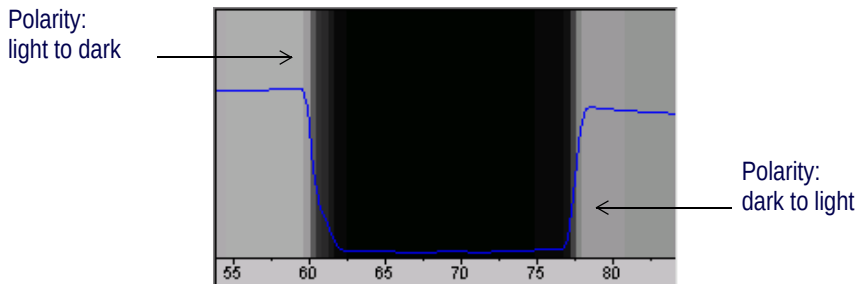


Figure 281 Polarity

Results

Arc Caliper results are read only properties shown for each instance of a valid edge pair found and located. Some of the edge properties do not depend on a positioning mode (see **Inspection Tools** chapter) or on a specific coordinate system.

- **Magnitude:** Magnitude of an edge expressed as its corresponding peak value in the magnitude curve.
- **Magnitude Score:** Score between 0 and 1, calculated according to an edge magnitude function.

- **Position Score:** Score between 0 and 1, calculated according to an edge position function.
- **Score:** A score is returned for each edge as well as for each edge pair. For the edge pair score, each edge accounts for 50% of the score.
- **Size:** The actual Arc Caliper measure. This measure is the length of the segment that connects the center point of each edge segment.

Some properties depend on the positioning mode or the coordinate system. These properties include, for each individual edge of a pair as well as for the pair, the X and Y positions, the orientation (Rotation) and the distance (Radius) between the center of the arc (Tool origin) and the position of an edge or an edge pair. [Figure 282](#) illustrates the position and orientation of an edge pair in both the radial and the annular projection modes.

- In the radial projection mode, the orientation (rotation) of an edge in the Tool coordinate system is calculated as the angle between the edge pair and the bisector. The position of the edge pair is represented located at the intersection of the edge pair center and the tool radius. The output radius is thus the tool radius.
- In the annular projection mode, the position is set to the intersection between the detected average arc edge pair and the bisector. This leads to a constant rotation of 90 degrees with respect to the tool. The output radius is variable in this case and corresponds to the radius where the edge pair has been detected.

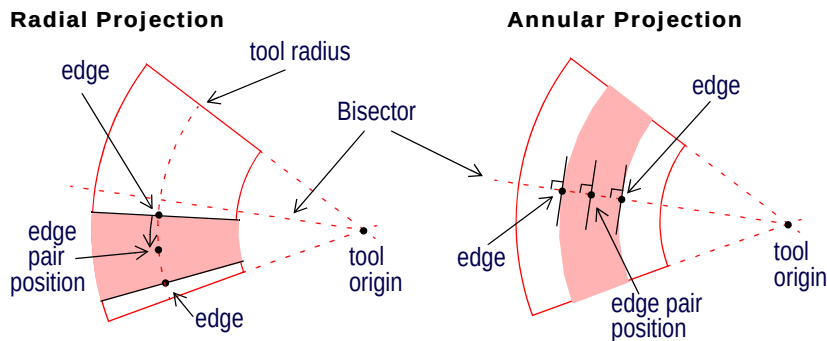


Figure 282 Edge Position and Orientation

Specific results that depend on the positioning mode and the selection of a coordinate system are presented in the following pages, regrouped by positioning mode.

Results in Model-Based Modes

In the Model-based mode, any of the four coordinate systems: Tool, Object, World, Image, can be selected (see [Figure 283](#)).

Position

X and Y positions are not available in the Tool coordinate system since this coordinate system is polar. X and Y positions are returned in the World, Object, and Image coordinate systems. In this last coordinate system, position is always expressed in pixels.

Rotation and Radius

In the Tool, Object and World coordinate systems, the angle of rotation is the angle of the edge pair, as defined in [Figure 282](#) for both radial and annular projection modes, expressed in the target coordinate system. The radius is the same in the above mentioned coordinate systems.

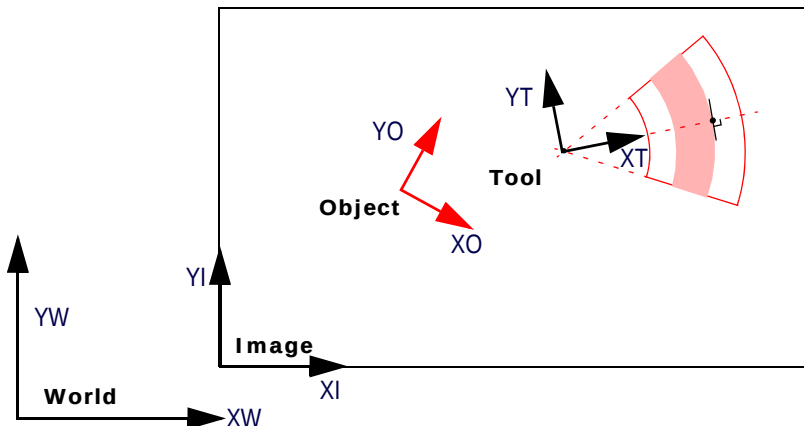


Figure 283 Coordinate Systems in Model-Based Positioning Modes

Since, in the Image coordinate system, edge pair positions are expressed in pixels, an estimation of the rotation angle is provided for radial projections.

This estimation is obtained by computing the arctangent of the angle defined by the back projection of the edge pair position and the tool origin in the Image. For both the annular and the radial projections, the radius is estimated following the same principle of back-projection. summarizes these cases.

Size

In radial projection mode, the size of an edge pair is equal to the length of the median axis portion between the two edges. When the projection is annular, it is equal to the distance between the two edges that form the pair. Nevertheless, size is not available in the image coordinate system.

Table 22 Image Coordinate Systems

Coordinate System	Properties	Radial Projection	Annular Projection
Tool	Position X, Y	N/A	N/A
	Rotation	relative to bisector	90 degrees
	Radius	tool radius	detected radius
	Size	Radius * angular difference	annular width
Object	Position X, Y	edge pair center	edge pair center
	Rotation	angle of bisector + Rotation in Tool coord. system	90 degrees + angle of bisector
	Radius	tool radius	tool radius
	Size	Radius * angular difference	annular width
World	Position X, Y	edge pair center	edge pair center
	Rotation	angle of bisector + Rotation in Tool coord. system	90 degrees + angle of bisector
	Radius	tool radius	tool radius
	Size	Radius * angular difference	annular width

Table 22 Image Coordinate Systems

Coordinate System	Properties	Radial Projection	Annular Projection
Image	Position X, Y	edge pair center in pixels	edge pair center in pixels
	Rotation	estimated locally	estimated locally
	Radius	estimated locally in pixels	estimated locally in pixels
	Size	N/A	annular width in pixels

Results in Image-Based Positioning Mode

In the Image-based positioning mode, there are two possible cases: calibrated and uncalibrated Images. While in the former case, the results can be output in the World, Tool and Image coordinate systems, results are only available in the Tool as well as the Image coordinate systems for uncalibrated Images. For both calibrated and uncalibrated images the Object coordinate system is not defined. [Figure 283](#) can also illustrate this case, except that the Object coordinate system is not present.

For calibrated Images, the results are output following the same general principle explained in the previous section for **Results in Model-Based Modes**.

[Table 23](#) summarizes the case of uncalibrated Images.

- X and Y positions are available only in the Image coordinate system.
- The angle of rotation is estimated and made available in both the Tool and the Image coordinate system only for the radial projection. In both of these coordinate systems, the radius is estimated in pixels.
- The size estimated from edge positions is available for all coordinate systems in non-calibrated modes.

Table 23 Image-Based Uncalibrated Properties

Coordinate System	Properties	Radial Projection	Annular Projection
Tool	Position X, Y	N/A	N/A
	Rotation	estimated	N/A
	Radius	estimated in pixels	estimated in pixels
	Size	Radius * angular difference in pixels	annular width in pixels
Image	Position X, Y	edge pair center estimated in pixels	edge pair center estimated in pixels
	Rotation	estimated	N/A
	Radius	estimated in pixels	estimated in pixels
	Size	Radius * angular difference in pixels	annular width in pixels
World	Position X, Y	edge pair center	edge pair center
	Rotation	angle of bisector + Rotation in Tool coord. system	90 degrees + angle of bisector
	Radius	tool radius	tool radius
	Size	Radius * angular difference in pixels	annular width in pixels

All results appear in runtime for each current instance. Consult the **Reference Manual** for more information on using Arc Caliper results.

Using the Arc Caliper

Setup of the Arc Caliper is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Arc Caliper follow the left to right order of these panels.

For each panel you will find pointers to guide you through the setup.

General Panel

Use this panel to name the instance of the Arc Caliper and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affects the required inputs.

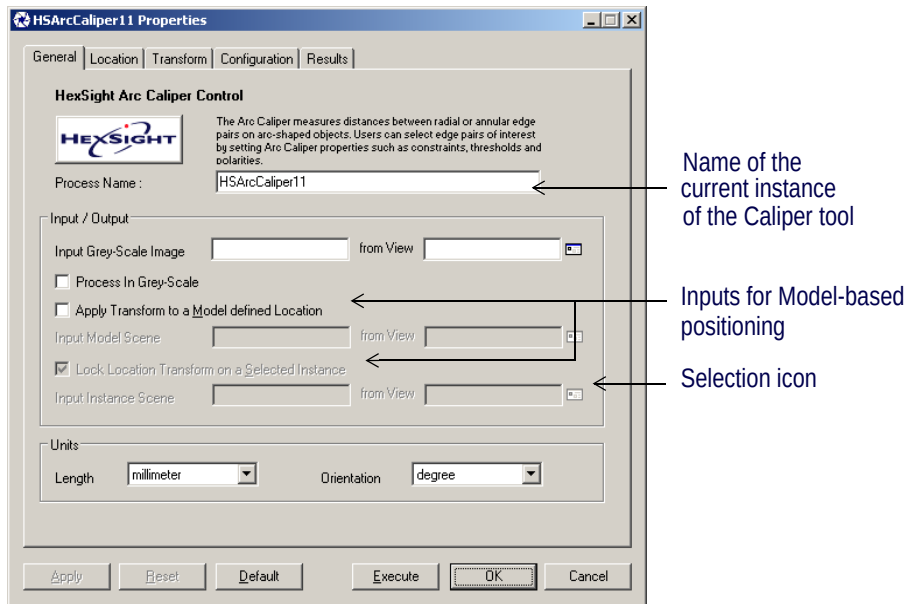


Figure 284 Arc Caliper — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based

automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 24 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 24 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSArcCaliper#, where # is replaced by its rank number in the list of instantiated processes.

Inputs

Image

The **Image** is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool.

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Sector, which defines the region of interest in the Image to which the process is applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Sector that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

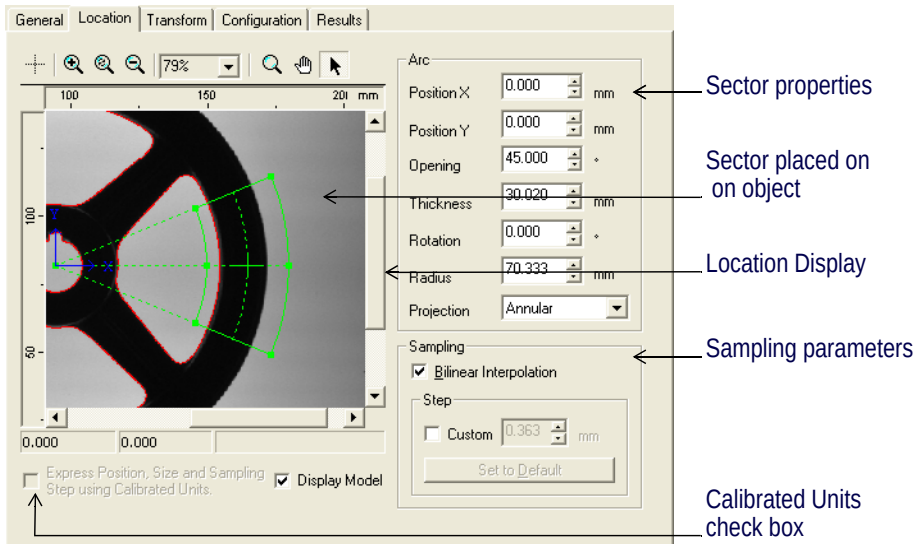


Figure 285 Arc Caliper — Location Pane

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Sector Parameters

The tool's region of interest is bounded by a Sector defined by the parameters **Position X**, **Position Y**, **Opening**, **Thickness**, **Rotation**, and **Radius**. See the *Inspection Tools* chapter of this guide for more details on the Sector.

- **Position X**, **Position Y**, **Thickness**, and **Radius** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Opening** and **Rotation** are always expressed in the selected orientation unit.
- With Image-based positioning, the Sector is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Sector is positioned with respect to the Object coordinate system.

Placing and Sizing the Sector

- Type the Sector parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Sector by dragging its border or its origin located at the intersection of its two bounding radii shown with dotted lines. Handle *A* in [Figure 286](#).
- Adjust the Sector's radius by dragging its center located at the intersection between the bisector and the median annulus. Handle *F* in [Figure 286](#).

- Set the Sector's thickness by dragging any of its four resizing handles located at intersections between its two bounding radii and annuli or by dragging the two resizing handles located at the intersections of the bisector and the two bounding annuli. Handles *B*, *D*, *E*, *G*, *H*, and *J* in [Figure 286](#).

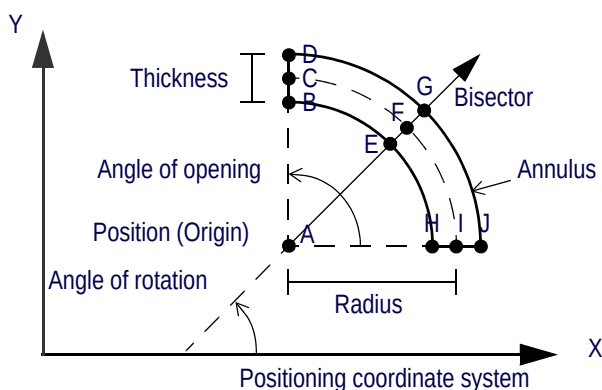


Figure 286 Arc Caliper — Adjusting the Sector

- Set the Sector's opening by dragging any of its four resizing handles located at intersections between its two bounding radii and annuli. Handles *B*, *D*, *H*, and *J* in [Figure 286](#).
- Rotate the Sector by dragging the two intersections between the median annulus and the two bounding radii. Handles *C*, and *I* in [Figure 286](#).

Note: You must adjust the Rectangle so that the projection image exceeds the edges you need to locate by a number of pixels greater than the value of filter half-width.

Projection

Use the **Projection** combo box to select between Radial and Annular projection modes. See the **Understanding Arc Caliper Concepts** section of this chapter for more details on projection.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that is applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

Transform Display

In addition to the input Image, the **Transform Display** also shows the Sector region of interest, transposed according to the transform parameters. Located edge pairs are shown as yellow lines within the Sector.

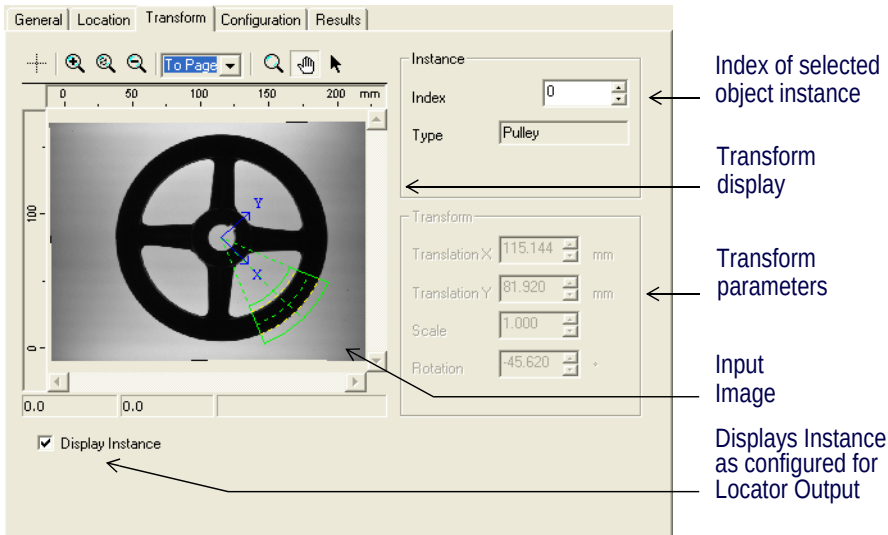


Figure 287 Arc Caliper — Transform Panel

Display Instance

Enable the **Display Instance** check box to activate a visual representation of the instance selected in the **Index** text box. This representation of the object is provided by the Instance Scene generated by the Locator. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model

corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled, the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure how edge pairs are detected in the Sector. The Configuration panel is used to configure edge detection parameters as well as to set polarity and scoring constraints based on position and magnitude.

Sector Display

Use the Sector display to visualize the configuration and the results of the edge detection process. By default, this display shows the Image sampled from the region of interest onto which are overlaid the projection and edge magnitude curves, the detection thresholds and the detected edges.

Display Options

Left-clicking on the Rectangle display opens a pop-up window displaying the projection value, edge magnitude, position within the region of interest, and score at the actual cursor position.

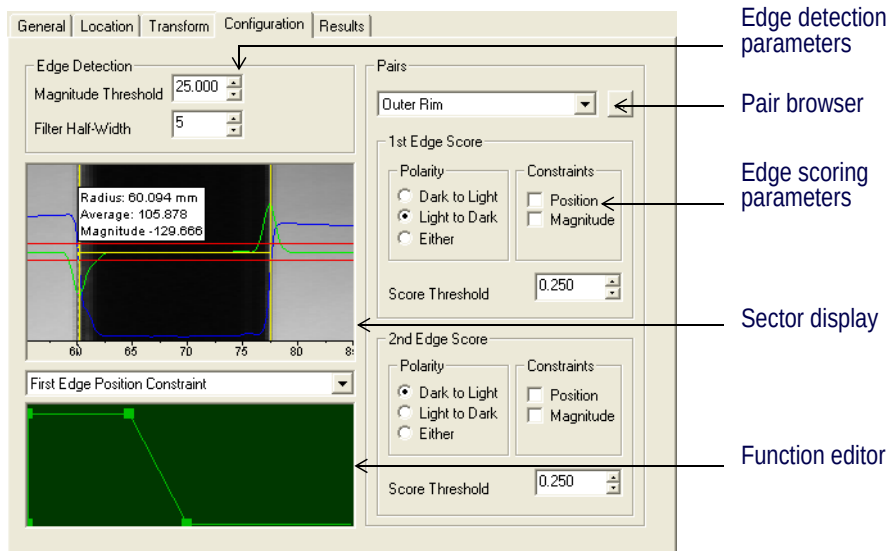


Figure 288 Arc Caliper — Configuration Panel

Right-clicking on the Rectangle display opens a context menu of the following display options.

- **Image** displays the Image sampled from the region of interest bounded by the Sector and mapped to a rectangle. The Image is useful for visualizing edges with respect to other features in the region of interest.

If radial projection is used, the horizontal and vertical axes of the mapped Image respectively represent angular and annular steps in the Sector.

If annular projection is used, the horizontal and vertical axes of the mapped Image then represent annular and angular steps in the Sector.

- **Projection Image** displays the projection Image used to detect edges in the region of interest. This projection is built by averaging each column of pixel in the sampled Image. Use the projection image to obtain a visual indication of the edge features in the region of interest.

If radial projection is used, the horizontal axis represents angular steps in the Sector.

If annular projection is used, the horizontal axis then represents annular steps in the Sector.

- **Edges** displays the edges detected in the region of interest as vertical yellow lines.
- **Thresholds** displays the positive and negative thresholds used to detect peaks on the edge magnitude curve as two horizontal red lines.
- **Projection Curve** displays a plot of the projection as a blue curve. This one-dimensional signal is computed by averaging each column of pixel in the sampled Image. Edges occur at sharp transitions in the curve.
- **Magnitude Curve** displays a plot of the edge magnitude as a green curve. The edge magnitude signal is the first derivative of the projection curve. Edges in the region of interest appear as positive or negative peaks on this curve. Sharp peaks indicate the presence of well-defined edges. Dull peaks indicate weak edges, or variations caused by smooth transitions.

Edge Detection Parameters

Magnitude Threshold

Magnitude Threshold defines the threshold above which the absolute value of a peak on the edge magnitude curve is considered as a potential edge.

- All peaks on the edge magnitude curve that fall between the two red magnitude threshold lines are not considered.
- All peaks on the edge magnitude curve that are above the upper threshold or below the lower threshold are considered as potential edges.
- Left-click on a peak in the Sector display to view its magnitude in a pop-up window.
- Sometimes unwanted edges, such as those corresponding to shadows, have magnitude values CLOSE to those of valid edges. When possible, do NOT use the magnitude threshold to reject such edges because in some images, lighting variations may cause edges of interest to also fall below the threshold. Instead, use polarity and position constraints that are not affected by lighting variations from one image to another.

Filter Half-Width

Filter **Half-Width** defines the half-width of the filter used to compute the first derivative of the projection curve. When properly set, the filter enhances edge information. Peaks on the magnitude curve that correspond to noise can be minimized, while peaks corresponding to potential edges can be made sharper.

- Set the filter half-width to a value equivalent to the number of pixels which the greylevel transition associated with the edge to be detected spans on the projection curve.
- If filter half-width set is too small compared with the width of the greylevel transition, edges might not be detected due to very low peaks on the edge magnitude curve.

- If filter half-width is set too large, positioning accuracy of detected edges might be affected by neighboring edges.

Note: If you increase the filter half-width, return to the Location panel to make sure that Sector is still sufficiently oversized. You must adjust the Sector so that, in the projection image exceeds the edges that must be located by a number of pixels greater than the value of filter half-width.

Pairs

Unlike the Arc Edge Locator, the Arc Caliper looks only for the number of edge pairs that have been configured. By default, when a new Arc Caliper is created, a single pair named *Pair0* is configured. To add a new pair or rename/delete an existing pair, access the **Pair Manager** dialog by clicking the (...) button.

- Click **Add Pair** to create a new pair.
- **Remove Pair** deletes the selected pair.
- Clicking **Rename Pair** prompts you for a new pair name.

Use the Pairs combo box to select the pair for which you want to configure the polarity and scoring functions.

- If the Arc Caliper is configured to measure only one edge pair, it will only output results for one pair, even though other pairs within the Sector respect the constraints.
- If no scoring is set for the two edges of the pair, the Arc Caliper scans the projection curve from left to right and uses the first two edges that meet the polarity criteria to build the pair.
- If position and/or magnitude scoring is selected, the Arc Caliper uses the two edges with the highest score values to compute the properties of the pair.

Polarity

You can constrain the Arc Caliper to build the pair only with edges of specific polarities using the first and second edge **Polarity** radio buttons.

- **Dark to Light** will only consider positive peaks on the edge magnitude curve. Positive peaks are caused by transitions, in the left to right direction, from darker to lighter greylevels on the projection curve.
- **Light to Dark** will only consider negative peaks on the edge magnitude curve.
- **Either** will consider any peak on the edge magnitude curve.

Scoring

You can configure the Arc Caliper to individually score the potential first and second edges of the pair according to position and/or magnitude criteria.

When position and/or magnitude scoring is enabled for the first edge of the pair, the potential edge with the highest score according to the first edge scoring criteria is used as the first edge. Similarly, if position and/or magnitude scoring is enabled for the second edge of the pair, the potential edge with the highest score according to the second edge scoring criteria is used to complete the pair. The total score of an edge is computed as follows:

- Always 1 if **Position** and **Magnitude** check boxes are not enabled.
- Equal to position score if only the **Position** check box is enabled.
- Equal to magnitude score if only the **Magnitude** check box is enabled.
- Equal to the average of position and magnitude scores if both **Position** and **Magnitude** check boxes are enabled.

Position

You can use a position constraint function to score the potential first and/or second edges by enabling its associated **Position** check box. The position scoring function for the first edge of the pair is defined with the function editor when **First Edge Position Constraint** is selected in the function editor's selection combo box. Select **Second Edge Position Constraint** to define the position scoring function for the second edge of the pair.

- Drag the four handles in the function editor to define the position scoring function.

- The X-axis of the function editor corresponds to the X-axis used to plot the projection and edge magnitude curves.
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position is expressed as a fraction of the total width of the image or projection image displayed in the Sector display. So for instance if a handle is placed exactly at the center of the function editor, its X-position is 0.5.
- Y-position of the handles defines its associated score. The Y-position of the handles is fixed and is 0 for the two outer handles and 1 for the inner two.
- The position score for one edge is interpolated from the linear segment connecting the nearest handles on each side of this edge. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Magnitude

You can use a magnitude constraint function to score the potential first and/or second edges by enabling its associated **Magnitude** check box. The magnitude scoring function for the first edge of the pair is defined with the function editor when **First Edge Magnitude Constraint** is selected in the function editor's selection combo box. Select **Second Edge Magnitude Constraint** to define the magnitude scoring function for the second edge of the pair.

The magnitude constraint is typically used in conjunction with the position constraint, to refine edge detection or to eliminate a set of edges whose magnitude is below the threshold level.

- Drag the two handles in the function editor to define the position scoring function.
- The X-axis of the function editor corresponds to magnitude values ranging from 0 (no edge) to 255 (the highest magnitude).
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position corresponds directly to an edge

magnitude value on the X-axis. For instance, if a handle is placed exactly at the center of the function editor, its X-position corresponds to the edge magnitude 128.

- Y-position of the handles defines its associated score. The Y-position of the handles can be either 0 or 1.
- The magnitude score for one edge is interpolated from the linear segment connecting the nearest handles on each side of its magnitude. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Score Threshold

You can use **Score Threshold** to reject potential first and second edges with low score values. This threshold is applied to the total score of the edges and not on individual position and magnitude score components.

Results Panel

The Results panel provides a grid of results and a visual display of the results.

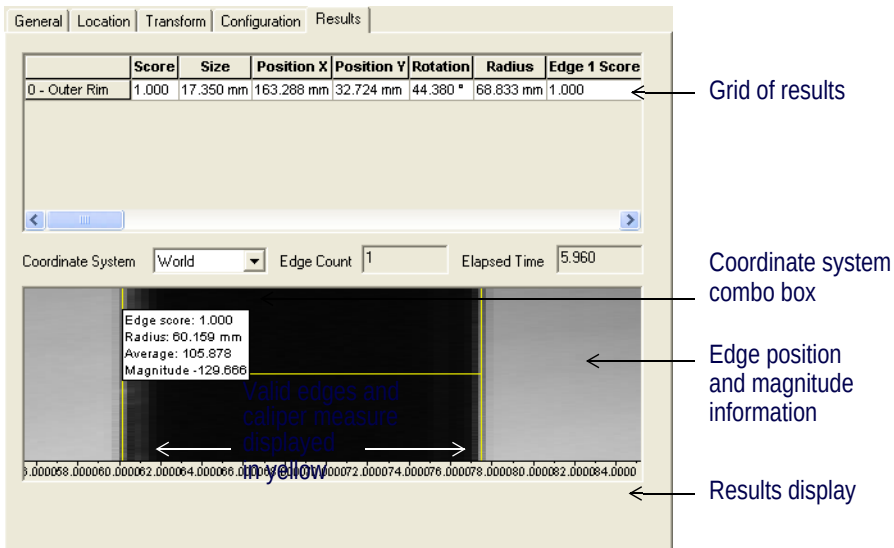


Figure 289 Arc Caliper — Results Panel

Grid of Results

The grid of results displays properties for each edge pair and for each edge in an edge pair.

- **Score** is the score computed for the edge pair.
- **Size** is the actual measure of the distance between the edges of the pair.
- **Position X** is the X coordinate of the center point for each edge pair, referenced in selected units in the selected coordinate system.
- **Position Y** is the Y coordinate of the center point for each edge pair, referenced in selected units in the selected coordinate system.
- **Rotation** is the angle for the edge pair, in orientation units.
- **Radius** is the distance between the edge pair and the tool origin.
- **Edge 1 / 2 Score** are the scores for the first/second edge of the pair. If both position and magnitude constraints are used, each constraint accounts for 50% of the edge score.
- **Edge 1 / 2 Position X** are the X coordinates of the center point for the first/second edge segments of the pair.
- **The Edge 1 / 2 Position Y** are the Y coordinates of the center point for the first/second edge segments of the pair, referenced in selected units in the selected coordinate system.
- The **Edge 1 / 2 Rotation** are the angles of rotation for the first/second edges of the pair, in orientation units.
- The **Position Score**, is computed for both the first and second edges of each pair, according to the position constraint function.
- The **Magnitude**, for both the first and second edges of each pair, is the peak value of the edge in the magnitude curve.
- The **Magnitude Score**, is computed for both the first and second edges of each pair, according to the magnitude constraint function.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: Image, World, Object, Tool.

The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Results Display

The Results panel includes a visual display of the Sector.

- Use display's context menu (right-click) to display either the Image, mapped from the Sector or the projection image.
- Context menu options also include the projection curve (blue), the edge magnitude curve (green), the threshold lines (red), and the detected edges (yellow).
- Left-click on the display to open a pop-up window describing the position, projection value, edge magnitude and edge scores.

Elapsed Time

Elapsed Time shows the total execution time of the Arc Caliper. This includes sampling and projection, edge detection, validating, sorting and returning results.

Pair Count

The pair count indicates the number of pairs that have been set up within the **Configuration panel**. If an edge pair is not found, results for this edge pair appear as zero. The pair count is not affected.

Arc Edge Locator

The Arc Edge Locator finds and locates an edge or a set of edges that meet user-defined constraints on an arc-shaped region of interest. The located edges can be scored and sorted according to user-defined criteria.

Arc Edge Locator Chapter Outline

[Understanding Arc Edge Locator Concepts](#)

[Tool Position](#)

[Edge Detection](#)

[Edge Scoring](#)

[Results](#)

[Using the Arc Edge Locator](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Arc Edge Locator Concepts

The Arc Edge Locator tool finds and locates one or more edges on an object within a bounded area called a Sector. The Arc Edge Locator uses pixel greylevel values within the Sector to build projections, either annular or radial, needed for edge detection.

Once potential edges have been located, the Arc Edge Locator scores these edges according to user-defined criteria and then returns results for valid edges it has located. The Arc Edge Locator process is presented in four sections: **Tool Position**, **Edge Detection**, **Edge Scoring** and **Results**.

Tool Position

The area processed by an arc tool, such as the Arc Edge Locator, is bounded by a Sector such as the one illustrated in [Figure 290](#).

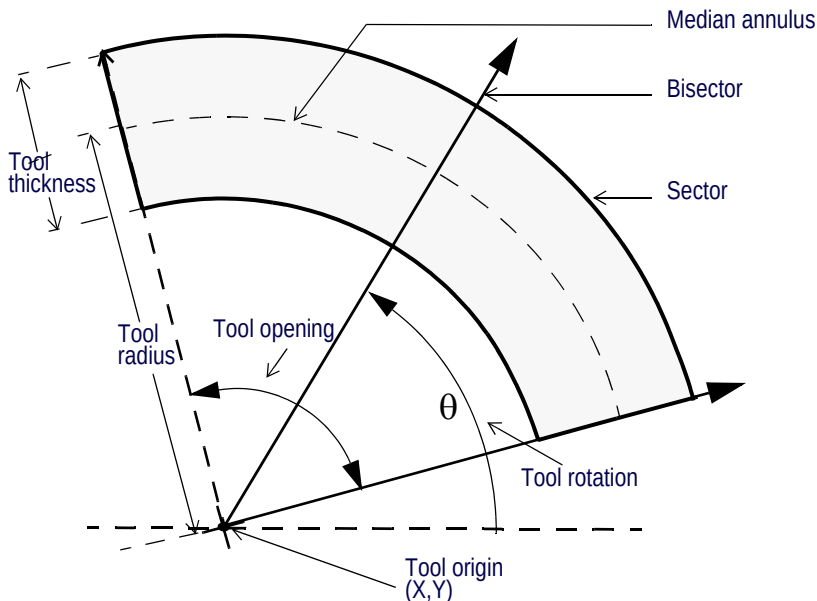


Figure 290 Arc Tool Sector

The tool's Sector is characterized by its opening, radius, thickness, position and orientation. The position of the tool corresponds to the (X,Y) coordinates

of its origin. The tool orientation corresponds to the angle between a horizontal axis and the bisector.

Edge Detection

Edges represent contours of an object, or physical boundaries between areas on an object. Edge enhancement projections, based on changes in greylevel values, enable the Arc Edge Locator to carry out an efficient edge detection.

Projection Path

The Arc Edge Locator processes data within the Sector along pixel-wide lines, called projection paths. Projection paths can be either annular or radial. Annular projection is used to find edges that are aligned with the median annulus (see [Figure 290](#)) such as arcs on concentric circles. Radial projection is used to find edges aligned along radial projections, much like the spokes of a wheel. These projection paths are illustrated in [Figure 291](#).

The average greylevel value for each projection path is stored in a one-dimensional signal that can be used in the edge location process.

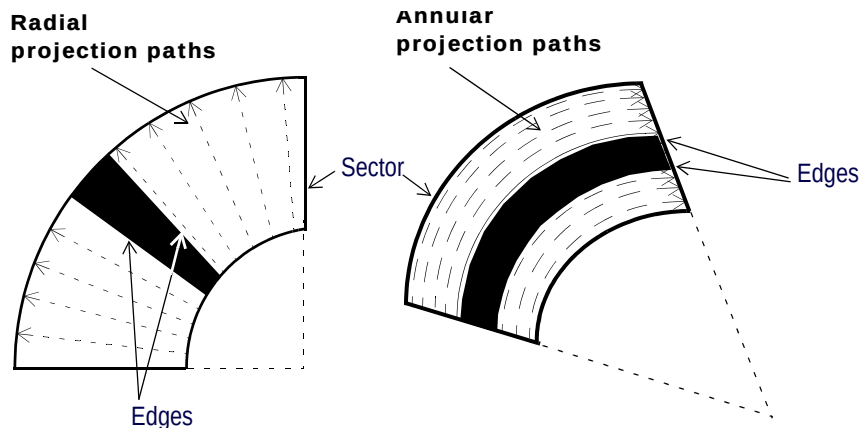


Figure 291 Radial vs. Annular Projection Paths

Projection Image

The Arc Edge Locator uses pixel values within each projection path to build a projection image, as shown in [Figure 292](#). This is a mapping from a Sector to a rectangle. Whereas for radial projection, the X-axis of the rectangle depicts an angular step, it represents an annular step (in length units) in the case of annular projection. The projection image is interpreted as a projection curve in which potential edges are indicated as strong positive or negative changes in greylevels.

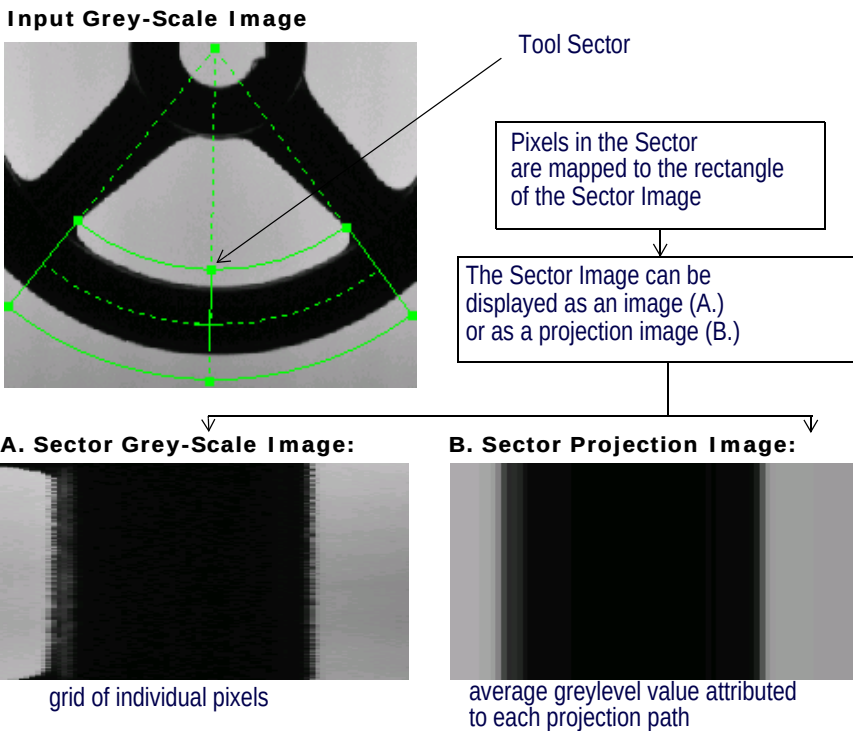


Figure 292 Image vs. Projection Image

As opposed to the Image, which displays each pixel's individual greylevel value, the projection image displays the average greylevel value for each projection path within the physical area bounded by the Sector. This minimizes variations in pixel values caused by non-edge features or noise.

Projection Curve

The projection curve is a graphical representation of the projection image within the Arc Edge Locator's Sector.

Sharp transitions in the curve intersect with potential edges; the steeper the rise or fall in a curve, the stronger the edge.

This curve also indicates changes in light values which characterize the polarity of the edge. Polarity is particularly important when setting constraints for the detection of valid edges.

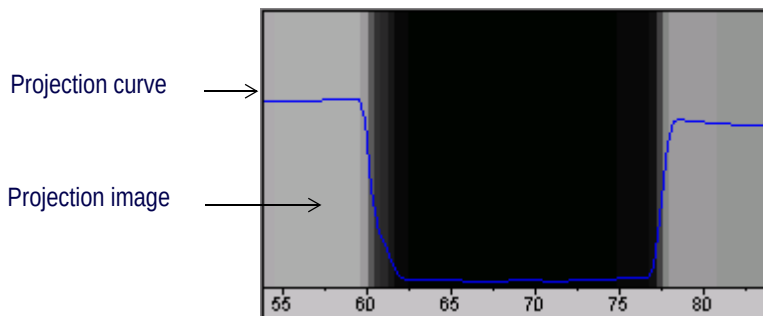


Figure 293 Projection Curve

Magnitude Curve

The magnitude curve is the first derivative of the projection curve. For a 256 greylevel image, it may range between -255 and 255. Positive and negative peaks in the curve indicate potential edges. Sharp peaks indicate strong, well-defined edges whereas dull peaks may indicate noise or poorly-defined edges.

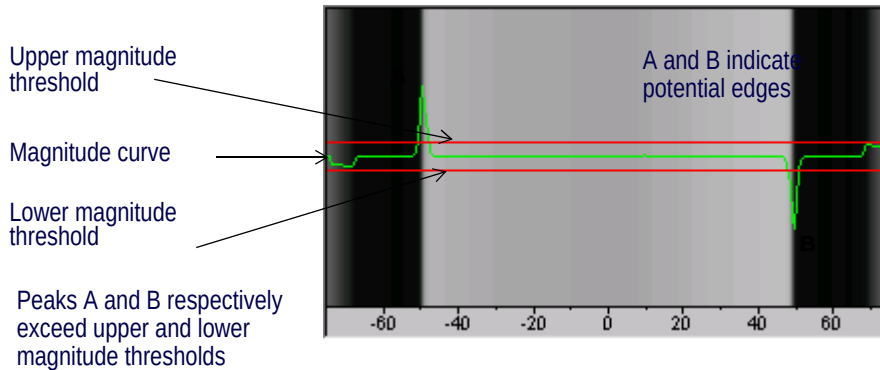


Figure 294 Magnitude Threshold

Magnitude Threshold

The magnitude threshold indicates an absolute value, above which peaks in the magnitude curve indicate potential edges.

Filter

The filtering process attenuates peaks in the magnitude curve that are caused by noise. The filter half-width value corresponds to the half-width of the convolution kernel used to estimate the first derivative of the projection curve. It should be approximately equivalent to the width of the edge, in pixels, as it appears in the Image.

Edge Scoring

The Arc Edge Locator scores potential edges according to the constraints set for the edge, or set of edges, within the Sector. The scoring method restricts the Arc Edge Locator's search so that only results for valid edges are returned.

There are two basic types of constraints: edge position and edge magnitude. You can set the Arc Edge Locator to use only one constraint type or both. A graphical function editor is provided for viewing and setting each type of constraint.

Magnitude Constraint

The magnitude constraint is based on edge values relative to the magnitude threshold. Edges having a magnitude equal to, or exceeding the magnitude threshold, are attributed a score of **1**. Edges with values below the magnitude threshold receive a score ranging from **0** to **0.999**, according to a manually set magnitude constraint function.

[Figure 295](#) shows two different setups for a magnitude constraint function.

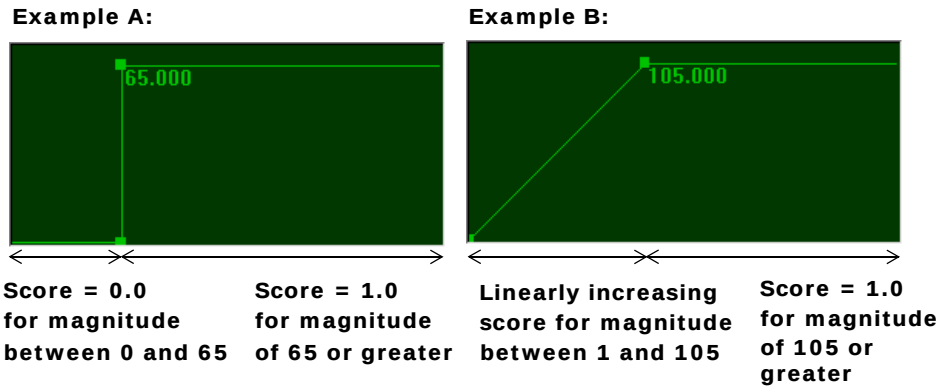


Figure 295 Magnitude Constraint Function

Position Constraint

Position constraints designate the physical location of an edge within the Sector Image. This restricts the Arc Edge Locator's search for an edge to a specific zone of an Image.

- It is possible to graphically set a position constraint function in the projection image when the approximate position of an edge is known beforehand. This is useful for scoring an edge based on its offset from the expected position.

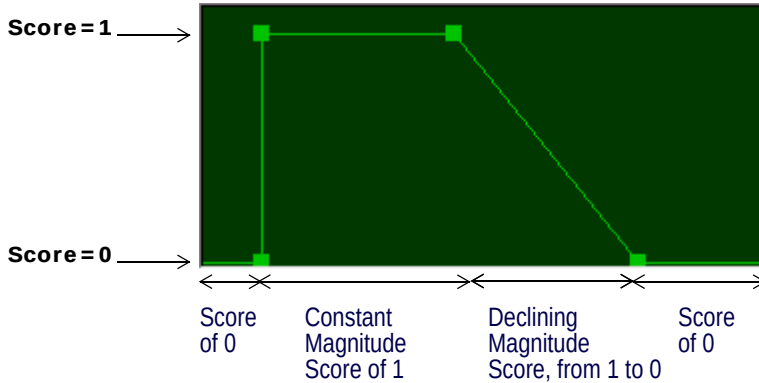


Figure 296 Position Constraint Function

Score Threshold

Scores attributed by the Arc Edge Locator for constraints, range from **0** to **1**. If both position and magnitude constraints are enabled, each constraint accounts for 50% of the total edge score. A user-defined threshold is applied to the score to only retain the strongest edges. You can enable the Arc Edge Locator to sort edge results according to the total score received for each edge.

Polarity

Polarity is not taken into account for scoring edges but it is particularly important when setting constraints for the detection of valid edges. The Arc Edge Locator applies this constraint before applying scored constraints. Edges are classified by their polarity, defined in the interface as either **Light to Dark** or **Dark to Light**, as it changes from left to right in the tool sector. See Figure 297.

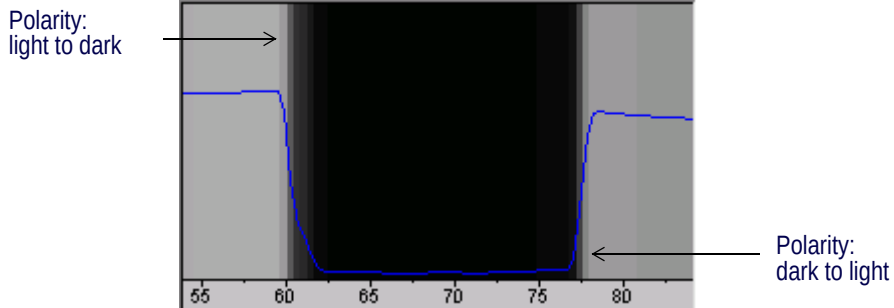


Figure 297 Polarity

Results

Arc Edge Locator results are read only properties shown for each instance of a valid edge. Some of the edge properties do not depend on a positioning mode (see **Inspection Tools** chapter) or on a specific coordinate system. These properties are:

- **Magnitude:** Magnitude of the edge expressed as its corresponding peak value in the magnitude curve.
- **Magnitude Score:** Score between 0 and 1, calculated according to the edge magnitude function.
- **Position Score:** Score between 0 and 1, calculated according to the edge position function.
- **Score:** The score for the edge.

Some properties depend on the positioning mode or the coordinate system. These properties include the X and Y positions of an edge, the orientation (Rotation) of an edge and the distance (Radius) between the center of the arc (Tool origin) and the position of the edge. [Figure 298](#) illustrates the position and orientation of an edge in both the radial and the annular projection modes.

- In the radial projection mode, the orientation (rotation) of an edge in the Tool coordinate system is calculated as the angle between the edge and the bisector. The position of the edge is located at the intersection

of the edge and the tool radius. The output radius is thus the tool radius.

- In the annular projection mode, the position is set to the intersection between the detected arc edge and the bisector. This leads to a constant rotation of 90 degrees with respect to the tool. The output radius is variable in this case and corresponds to the radius where the edge has been detected.

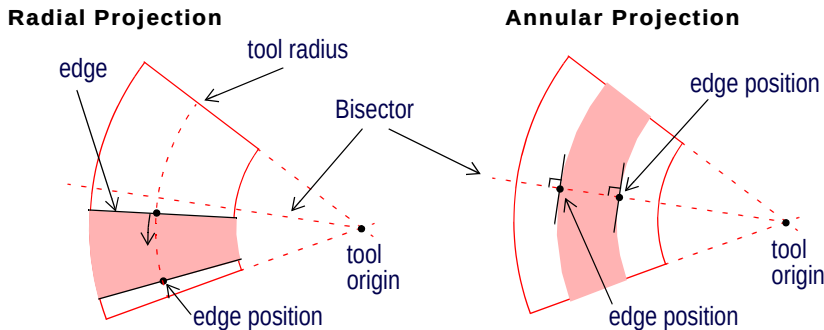


Figure 298 Edge Position and Orientation

Specific results that depend on the positioning mode and the selection of a coordinate system are presented in the following pages, regrouped by positioning mode.

Results in Model-Based Modes

In the Model-based mode, any of the four coordinate systems: Tool, Object, World, Image, can be selected (see [Figure 299](#)).

Position

X and Y positions are not available in the Tool coordinate system since this coordinate system is polar. X and Y positions are returned in the World, Object and Image coordinate systems. In this last coordinate system, position is always expressed in pixels.

Rotation and Radius

In the Tool, Object and World coordinate systems, the angle of rotation is the angle of the edge, as defined in [Figure 298](#) for both radial and annular projection modes, expressed in the target coordinate system. The radius is the same in the above mentioned coordinate systems.

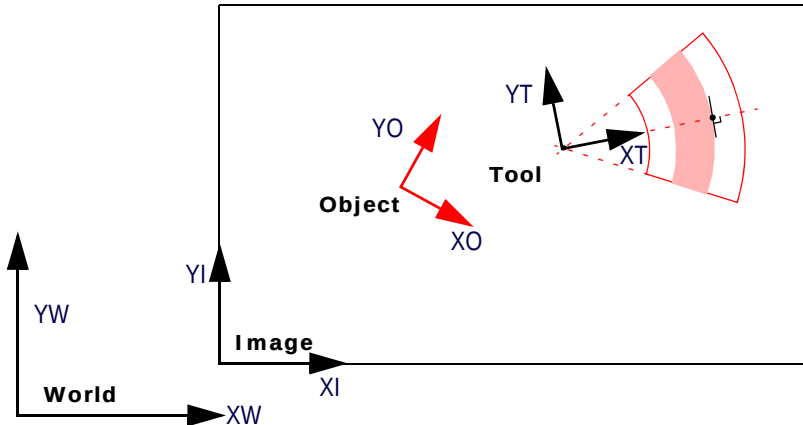


Figure 299 Coordinate Systems in Model-Based Positioning Modes

Since, in the Image coordinate system, edge positions are expressed in pixels, an estimation of the rotation angle is provided for radial projections. This estimation is obtained by computing the arctangent of the angle defined by the back projection of the edge position and the tool origin in the Image. For both the annular and the radial projections, the radius is estimated following the same principle of back-projection. Figure 299 summarizes these cases.

Results in Image-Based Positioning Mode

In the Image-based positioning mode, there are two possible cases: calibrated and uncalibrated Images. While in the former case, the results can be output in the World, Tool and Image coordinate systems, results are only available in the Tool as well as the Image coordinate systems for uncalibrated Images. For both calibrated and uncalibrated images, the Object coordinate

system is not defined. [Figure 299](#) can also illustrate this case, except that the Object coordinate system is not present.

For calibrated Images, the results are output following the same general principle explained in the previous section for **Results in Model-Based Modes**.

summarizes the case of uncalibrated Images.

- X and Y positions are available only in the Image coordinate system.
- The angle of rotation is estimated and made available in both the Tool and the Image coordinate system only for the radial projection. In both of these coordinate systems, the radius is estimated in pixels.

Table 25 Image Coordinate Systems

Coordinate System	Properties	Radial Projection	Annular Projection
Tool	Position X, Y	N/A	N/A
	Rotation	estimated	N/A
	Radius	estimated in pixels	estimated in pixels
Image	Position X, Y	estimated in pixels	estimated in pixels
	Rotation	estimated	N/A
	Radius	estimated in pixels	estimated in pixels
World	Position X, Y	edge center	edge center
	Rotation	angle of bisector + Rotation in Tool coord. system	90 degrees + angle of bisector
	Radius	tool radius	tool radius
Image	Position X, Y	in pixels	in pixels
	Rotation	estimated locally	estimated locally
	Radius	estimated locally	estimated locally

All results appear in runtime for each current instance. Consult the **Reference Manual** for more information on using Edge Locator results.

Using the Arc Edge Locator

Setup of the Arc Edge Locator is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Arc Edge Locator follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Arc Edge Locator and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affects the required inputs.

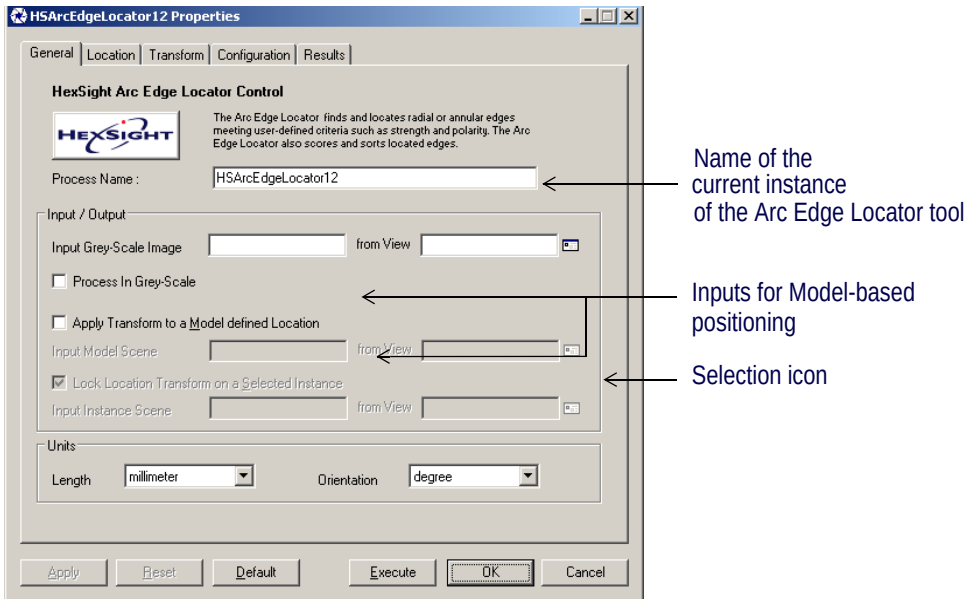


Figure 300 Arc Edge Locator — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based

automatic. See the **Inspection Tools** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 26 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 26 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSArcEdgeLocator#, where # is replaced by its rank number in the list of instantiated processes.

Inputs

Image

The **Image** is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Sector, which defines the region of interest in the Image to which the process is applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Sector that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

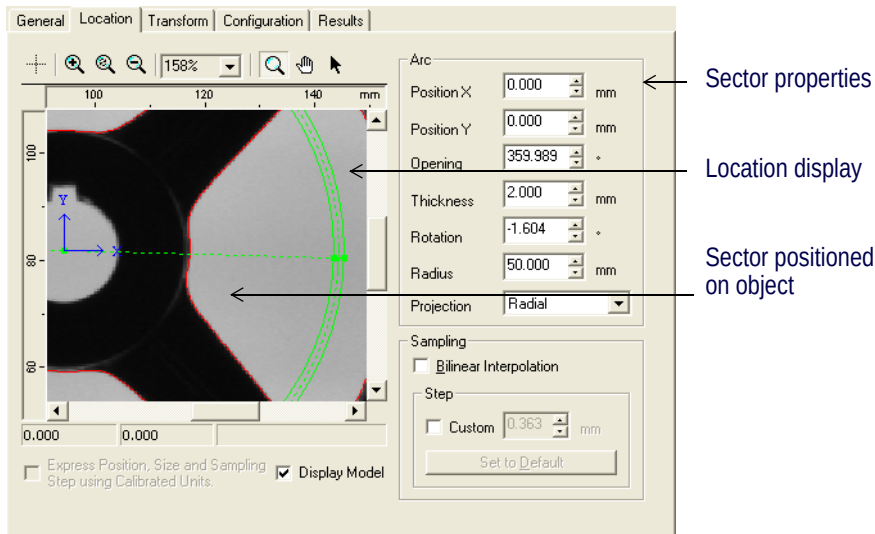


Figure 301 Arc Edge Locator — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Sector Parameters

The tool's region of interest is bounded by a Sector defined by the parameters **Position X**, **Position Y**, **Opening**, **Thickness**, **Rotation**, and **Radius**. See the *Inspection Tools* chapter of this guide for more details on the Sector.

- **Position X**, **Position Y**, **Thickness**, and **Radius** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Opening** and **Rotation** are always expressed in the selected orientation unit.
- With Image-based positioning, the Sector is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Sector is positioned with respect to the Object coordinate system.

Placing and Sizing the Sector

- Type the Sector parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Sector by dragging its border or its origin located at the intersection of its two bounding radii shown with dotted lines. Handle *A* in [Figure 302](#).
- Adjust the Sector's radius by dragging its center located at the intersection between the bisector and the median annulus. Handle *F* in [Figure 302](#).

- Set the Sector's thickness by dragging any of its four resizing handles located at intersections between its two bounding radii and annuli or by dragging the two resizing handles located at the intersections of the bisector and the two bounding annuli. Handles *B*, *D*, *E*, *G*, *H*, and *J* in [Figure 302](#).

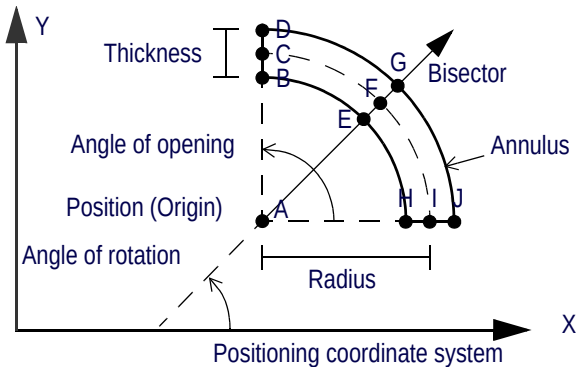


Figure 302 Arc Edge Locator — Adjusting the Sector

- Set the Sector's opening by dragging any of its four resizing handles located at intersections between its two bounding radii and annuli. Handles *B*, *D*, *H*, and *J* in [Figure 302](#).
- Rotate the Sector by dragging the two intersections between the median annulus and the two bounding radii. Handles *C*, and *I* in [Figure 302](#).

Note: You must adjust the Rectangle so that the projection image exceeds the edges you need to locate by a number of pixels greater than the value of filter half-width.

Projection

Use the **Projection** combo box to select between Radial and Annular projection modes. See the **Understanding Arc Edge Locator Concepts** section of this chapter for more details on projection.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that is applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

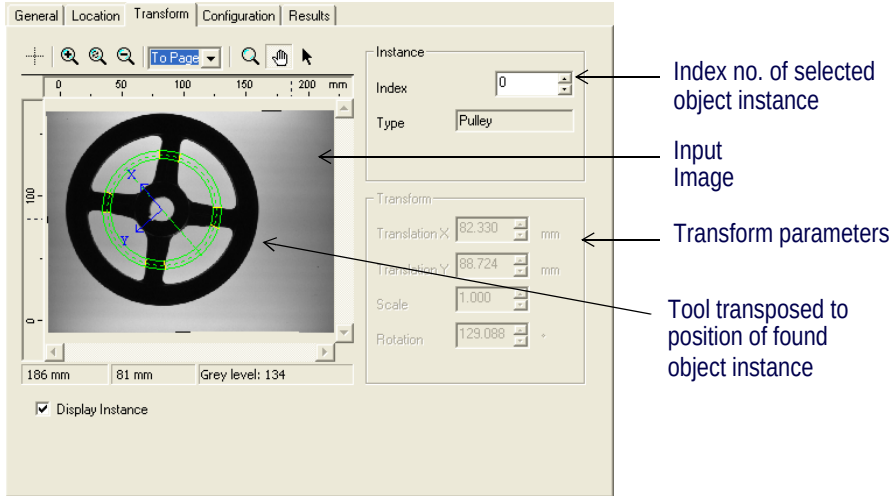


Figure 303 Arc Edge Locator — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Sector region of interest, transposed according to the transform parameters. Located edges are shown as yellow lines within the Sector.

Display Instance

Enable the **Display Instance** check box to activate a visual representation of the instance selected in the **Index** text box. This representation of the object is provided by the Instance Scene generated by the Locator. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled, the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure how edges are detected in the Sector. The Configuration panel is used to configure edge detection parameters as well as to set polarity and scoring constraints based on position and magnitude.

Sector Display

Use the Sector display to visualize the configuration and the results of the edge detection process. By default, this display shows the Image sampled from the region of interest onto which are overlaid the projection and edge magnitude curves, the detection thresholds and the detected edges.

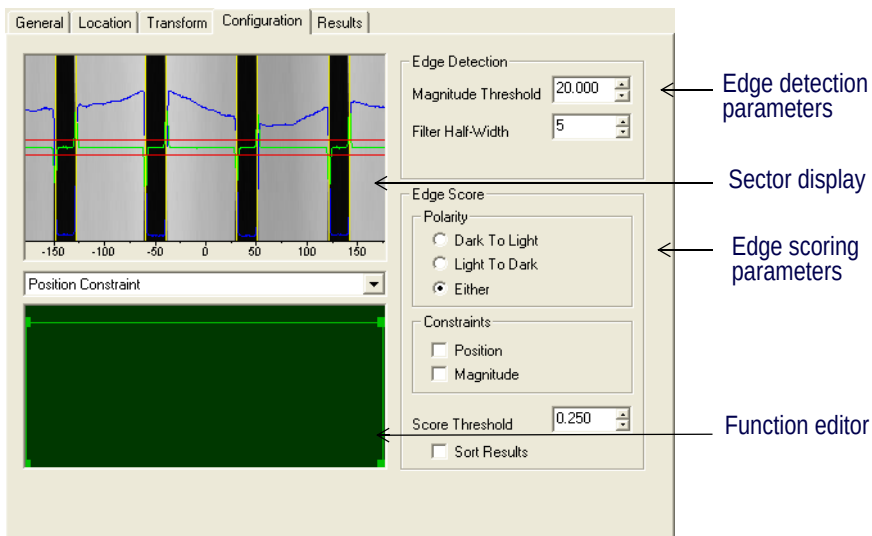


Figure 304 Arc Edge Locator — Configuration Panel

Display Options

Left-clicking on the Rectangle display opens a pop-up window displaying the projection value, edge magnitude, position within the region of interest, and score at the actual cursor position.

Right-clicking on the Rectangle display opens a context menu of the following display options.

- **Image** displays the Image sampled from the region of interest bounded by the Sector and mapped to a rectangle. The Image is useful for visualizing edges with respect to other features in the region of interest.

If radial projection is used, the horizontal and vertical axes of the mapped Image respectively represent angular and annular steps in the Sector.

If annular projection is used, the horizontal and vertical axes of the mapped Image then represent annular and angular steps in the Sector.

- **Projection Image** displays the projection Image used to detect edges in the region of interest. This projection is built by averaging each column of pixel in the sampled Image. Use the projection image to obtain a visual indication of the edge features in the region of interest.

If radial projection is used, the horizontal axis represents angular steps in the Sector.

If annular projection is used, the horizontal axis then represents annular steps in the Sector.

- **Edges** displays the edges detected in the region of interest as vertical yellow lines.
- **Thresholds** displays the positive and negative thresholds used to detect peaks on the edge magnitude curve as two horizontal red lines.
- **Projection Curve** displays a plot of the projection as a blue curve. This one-dimensional signal is computed by averaging each column of pixel in the sampled Image. Edges occur at sharp transitions in the curve.

- **Magnitude Curve** displays a plot of the edge magnitude as a green curve. The edge magnitude signal is the first derivative of the projection curve. Edges in the region of interest appear as positive or negative peaks on this curve. Sharp peaks indicate the presence of well-defined edges. Dull peaks indicate weak edges, or variations caused by smooth transitions.

Edge Detection Parameters

Magnitude Threshold

Magnitude Threshold defines the threshold above which the absolute value of a peak on the edge magnitude curve is considered as a potential edge.

- All peaks on the edge magnitude curve that fall between the two red magnitude threshold lines are not considered.
- All peaks on the edge magnitude curve that are above the upper threshold or below the lower threshold are considered as potential edges.
- Left-click on a peak in the Sector display to view its magnitude in a pop-up window.
- Sometimes unwanted edges, such as those corresponding to shadows, have magnitude values CLOSE to those of valid edges. When possible, do NOT use the magnitude threshold to reject such edges because in some images, lighting variations may cause edges of interest to also fall below the threshold. Instead, use polarity and position constraints that are not affected by lighting variations from one image to another.

Filter Half-Width

Filter Half-Width defines the half-width of the filter used to compute the first derivative of the projection curve. When properly set, the filter enhances edge information. Peaks on the magnitude curve that correspond to noise can be minimized, while peaks corresponding to potential edges can be made sharper.

- Set the filter half-width to a value equivalent to the number of pixels which the greylevel transition associated with the edge to be detected spans on the projection curve.
- If filter half-width set is too small compared with the width of the greylevel transition, edges might not be detected due to very low peaks on the edge magnitude curve.
- If filter half-width is set too large, positioning accuracy of detected edges might be affected by neighboring edges.

Note: If you increase the filter half-width, return to the Location panel to make sure that Sector is still sufficiently oversized. You must adjust the Sector so that, in the projection image exceeds the edges that must be located by a number of pixels greater than the value of filter half-width.

Polarity

You can constrain the Arc Edge Locator to accept only edges of specific polarities using the **Polarity** radio buttons.

- **Dark to Light** will only consider positive peaks on the edge magnitude curve. Positive peaks are caused by transitions, in the left to right direction, from darker to lighter greylevels on the projection curve.
- **Light to Dark** will only consider negative peaks on the edge magnitude curve.
- **Either** will consider any peak on the edge magnitude curve.

Scoring

You can configure the Arc Edge Locator to score the potential edges according to position and/or magnitude criteria. The total score of an edge is computed as follows:

- Always 1 if **Position** and **Magnitude** check boxes are not enabled.
- Equal to position score if only the **Position** check box is enabled.
- Equal to magnitude score if only the **Magnitude** check box is enabled.

- Equal to the average of position and magnitude scores if both **Position** and **Magnitude** check boxes are enabled.

Position

You can use a position constraint function to score the potential edges by enabling the **Position** check box. The position scoring function is defined with the function editor when **Position Constraint** is selected in the function editor's selection combo box.

- Drag the four handles in the function editor to define the position scoring function.
- The X-axis of the function editor corresponds to the X-axis used to plot the projection and edge magnitude curves.
- When a handle is dragged or when the mouse pointer stops above it, is X-position appears. This position is expressed as a fraction of the total width of the image or projection image displayed in the Sector display. So for instance if a handle is placed exactly at the center of the function editor, its X-position is 0.5.
- Y-position of the handles defines its associated score. The Y-position of the handles is fixed and is 0 for the two outer handles and 1 for the inner two.
- The position score for one edge is interpolated from the linear segment connecting the nearest handles on each side of this edge. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Magnitude

You can use a magnitude constraint function to score the potential edges by enabling the **Magnitude** check box. The magnitude scoring function is defined with the function editor when **Magnitude Constraint** is selected in the function editor's selection combo box.

The magnitude constraint is typically used in conjunction with the position constraint, to refine edge detection or to eliminate a set of edges whose magnitude is below the threshold level.

- Drag the two handles in the function editor to define the position scoring function.
- The X-axis of the function editor corresponds to magnitude values ranging from 0 (no edge) to 255 (the highest magnitude).
- When a handle is dragged or when the mouse pointer stops above it, its X-position appears. This position corresponds directly to an edge magnitude value on the X-axis. For instance, if a handle is placed exactly at the center of the function editor, its X-position corresponds to the edge magnitude 128.
- Y-position of the handles defines its associated score. The Y-position of the handles can be either 0 or 1.
- The magnitude score for one edge is interpolated from the linear segment connecting the nearest handles on each side of its magnitude. Two *invisible* handles are located at both ends of the scoring function with Y-position values of 0.

Score Threshold

You can use **Score Threshold** to reject edges with low score values. This threshold is applied to the total score of the edges and not on individual position and magnitude score components.

Sort Results

You can enable the **Sort Results** check box to sort the located edges in descending order of score values. By default, **Sort Results** is not enabled and edges are output in the same left to right order as they appear on the projection curve.

Results Panel

The Results panel provides a grid of results and a visual display of the results.

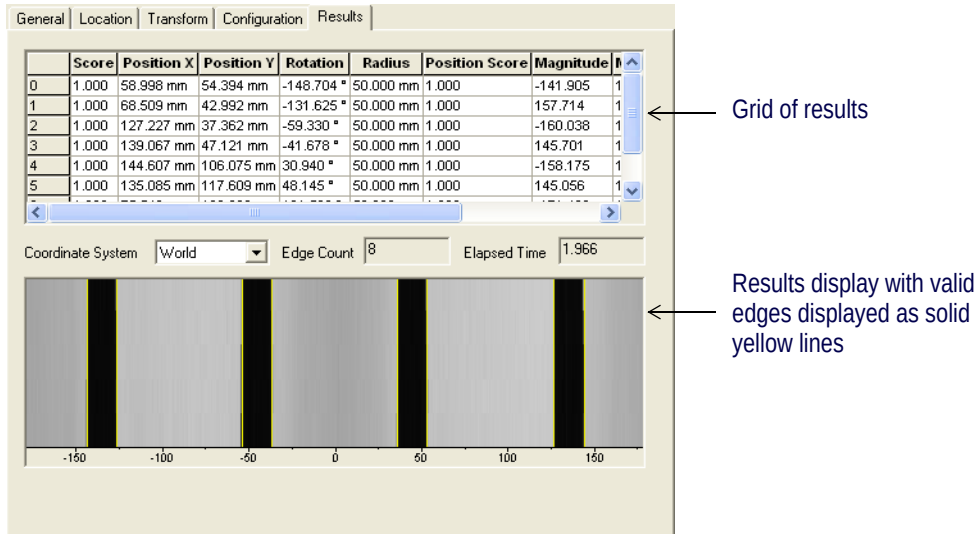


Figure 305 Arc Edge Locator — Results Panel

Grid of Results

Every valid edge is numbered and assigned a set of properties that can be viewed in the grid of results.

- **Score** is the total score for the edge. If both position and magnitude constraints are used, each constraint accounts for 50% of the edge score.
- **Position X** is the X coordinate of the center point for each edge segment, referenced in selected units in the selected coordinate system.
- **Position Y** is the Y coordinate of the center point for each edge segment.
- **Rotation** is the angle of rotation of the edge, in orientation units.
- **Radius** is the distance between the edge and the tool origin.
- The **Position Score** for the edge is computed according to the position constraint function.

- The **Magnitude** of the edge indicates its peak value in the magnitude curve.
- The **Magnitude Score** for the edge is computed according to the magnitude constraint function.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: Image, World, Object, Tool.

The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Sector Display

To help visualize the located edges, the Results panel includes a visual display of the Sector.

- Use display's context menu (right-click) to display either the Image, mapped from the Sector or the projection image.
- Context menu options also include the projection curve (blue), the edge magnitude curve (green), the threshold lines (red), and the detected edges (yellow).
- Left-click on the display to open a pop-up window describing the position, projection value, edge magnitude and edge scores anywhere along the projection curve.

Elapsed Time

Elapsed Time shows the total execution time of the Arc Edge Locator. This includes sampling and projection, edge detection, validating, sorting and returning results.

Edge Count

Edge Count indicates the number of valid edges that have been found within the Sector.

Blob Analyzer

The Blob Analyzer finds, labels and analyzes irregular shaped objects. This tool detects and computes intrinsic/extrinsic geometric and greylevel properties of blobs that meet user-defined criteria.

Blob Analyzer Chapter Outline

[Understanding Blob Analyzer Concepts](#)

[Selection Rectangle](#)

[Bilinear Interpolation](#)

[Image Segmentation](#)

[Area Constraints](#)

[Hole Filling](#)

[Blob Display](#)

[Results](#)

[Using the Blob Analyzer](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Blob Analyzer Concepts

The Blob Analyzer tool processes pixel information within an area bounded by a selection Rectangle. It uses pixel values within the Rectangle to apply image segmentation algorithms used for blob detection. User-defined criteria then restrict the Blob Analyzer's search for valid blobs.

The Blob Analyzer returns an array of numerical results for each valid blob that has been found and located. Blob results include geometric, topological and greylevel blob properties.

Selection Rectangle

From a user's point of view, the selection Rectangle can contain three types of data: all or part of the image background, all or part of an object and one or more blobs. In the context of blob analysis there are only two types of data: blob and background.

Once the Blob Analyzer has applied an image segmentation algorithm to the information in the Rectangle, it outputs a Blob Image in which all pixels are either part of a blob or part of the background.

Bilinear Interpolation

Bilinear interpolation is crucial for obtaining accurate Blob Analyzer results. If the Blob Analyzer is used in a Model-based mode, the selection Rectangle, and contained blobs, will rarely be aligned with the pixel grid, resulting in jagged edges on blob borders. Thus, interpolated pixel values provide a more true-to-life representation of blob contours. As illustrated in [Figure 306](#), a detail from a non-interpolated image shows a blob's contour as being very jagged and irregular.

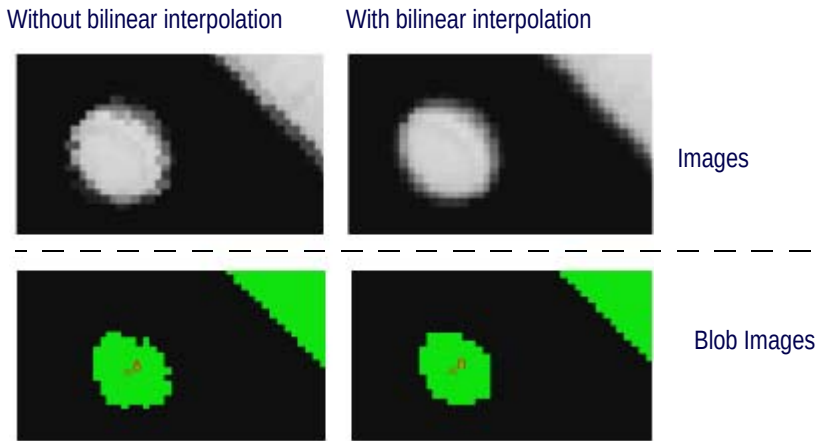


Figure 306 Effect of Bilinear Interpolation on Blob Detection

Image Segmentation

Blobs represent a contiguous region within an image characterized by a specific range of greylevel values.

For every new Image, the Blob Analyzer generates a histogram representing the distribution of the pixel values in the Rectangle. Image segmentation algorithms then provide thresholding functions allowing the Blob Analyzer to separate regions of the Image into two categories: blobs and background.

Histogram

The histogram provides the distribution of all the pixel values contained in the Blob Analyzer's selection Rectangle. Since HexSight uses 8-bit grey-scale images, pixel values in the histogram range from 0 (black) to 255 (white).

Images well suited to Blob detection typically present a bimodal histogram, i.e. with two cusps about which are grouped a greater part of the pixel distribution. Such a histogram is illustrated in [Figure 307](#). One of the masses of pixels corresponds to background pixels and the other to the object, or objects, in the Rectangle region of interest.

- In some cases, multimodal histograms with well-defined zones may indicate suitable images for blob detection.

- Images with poor contrast between blobs and background are not suited to blob applications. Unsuitable images present a histogram with a single mode or fairly even pixel distribution.

Good contrast between
object and background



Poor contrast between
object and background

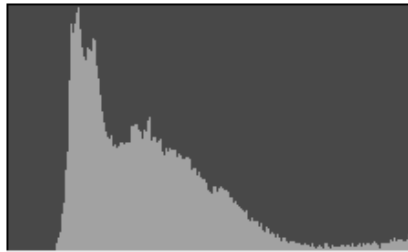


Figure 307 Blob Analyzer Histograms

Thresholding

Thresholds are used to segment the image into two types of pixels: blob pixels and background pixels. Depending on the segmentation mode selected, either one or two thresholds can be used. Moreover there are two types of thresholding functions: hard or soft.

Hard Thresholding

Hard thresholding is also referred to as binary thresholding since it segregates pixels into one of two states: 0 for background pixels, and 1 for blob pixels. The result is a binary image: for example a white blob on a black background. Hard thresholding assumes that changes in data values occur at the boundary between pixels, without allowing for a variation in greylevel values across blob boundaries. Since this is rarely the case, soft thresholding is more often used for applications.

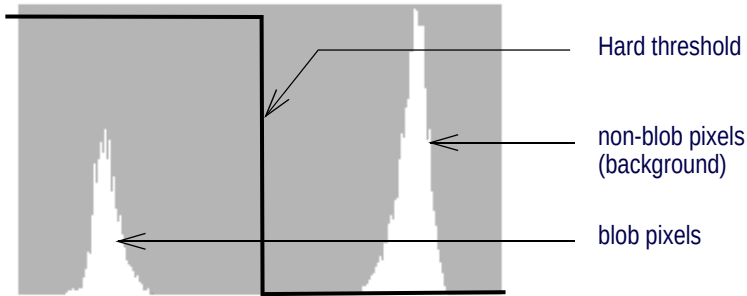


Figure 308 Image Segmentation - Hard Threshold

Soft Thresholding

Soft thresholding provides flexibility in treating pixels that border blob regions. A soft threshold is sloped and covers a range of pixel values. Once processed, pixels within the threshold range are output as weighted pixels. Weighted pixels are used to calculate blob results in a proportion corresponding to their value in the threshold range.

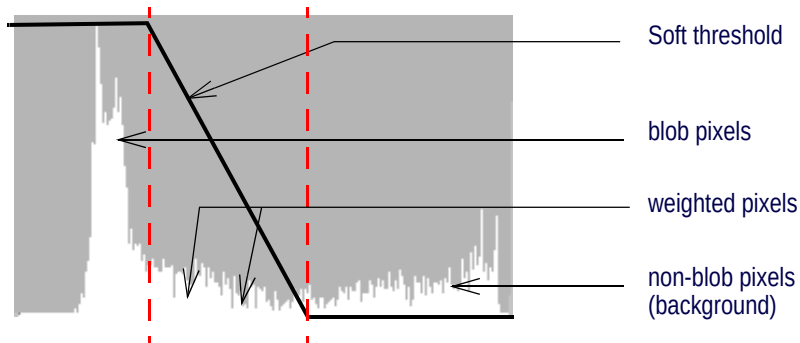


Figure 309 Image Segmentation - Soft Thresholding

- The range of values within a soft threshold is user-defined and corresponds to the difference between a maximum and a minimum threshold value.
- All weighted pixels appear in the **Blob Image**, but in fact they contribute to the property results in proportion to their weight in the threshold range.

- The **Pixel Weight Image** displays the weighted pixels in corresponding levels of grey.

Dynamic Thresholding

Dynamic thresholding provides an adaptive thresholding mode, particularly useful when there are lighting variations from one instance to another. A dynamic threshold sets a percentage of the pixel distribution (histogram) to be considered as blob pixels.

- A dynamic threshold can be either a soft or hard threshold and can be set to apply light dark, inside or outside segmentation.

Dynamic threshold for two instances of the same type of object:

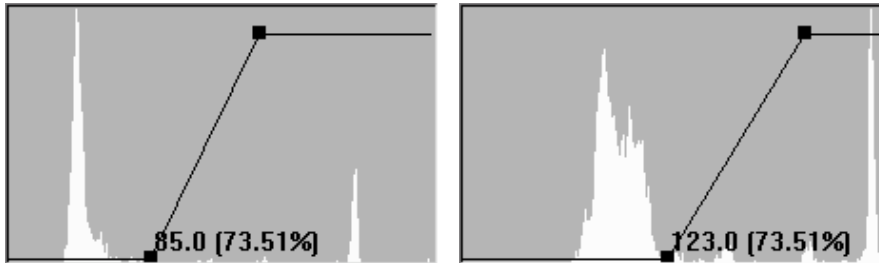


Figure 310 Image Segmentation - Dynamic Thresholding

Light Segmentation

The **Light** segmentation algorithm outputs all pixels with values higher than the threshold value as blob pixels. Pixels with values below the threshold become background pixels. This mode is used for light blobs on a dark background.

Dark Segmentation

The **Dark** segmentation algorithm is the inverse function of the Light segmentation algorithm. The Blob Analyzer outputs all pixels with values below the threshold value as blob pixels. Pixels with values above the threshold become background pixels. This mode is used for dark blobs on a light background.

Inside Segmentation

The **Inside** segmentation algorithm uses a dual threshold function. Only pixels with greylevel values within the threshold range are output as blob pixels. This mode is used when the pixel values of blobs fall somewhere between the values of the object and the Image background.

Outside Segmentation

The **Outside** segmentation operates as an inverse function of the **Inside** Segmentation Mode. Pixel values that fall outside the threshold range are output as blob pixels. This is useful for situations where blobs can be either very dark or very light as opposed to the background.

Area Constraints

User-set area constraints, minimum and maximum, are used to restrict the Blob Analyzer's search for blobs. The Blob area is the sum of the areas of all the pixels in the blob. When soft thresholding is used, the area of each pixel is weighted using the threshold function.

Hole Filling

When enabled, the Hole Filling parameter is applied to all detected blobs. All background within the perimeter of a given blob becomes included in the blob. All smaller blobs within a larger blob are also included in the "filled" larger blob. Both the background and smaller blobs are then considered as part of the filled blobs.

Blob Display

The Blob Display is a binary type image in which blobs are displayed in green and the background is displayed in black. The blob display can show blob markers, consisting, for each blob, of its index number and a graphical marker positioned at the center of mass.

Blob Image

The Blob Analyzer can be enabled to generate an Overlay Image; the Blob Image displayed in the Blob Analyzer's Results panel and in the Data Explorer. The Overlay Image provides useful visual information when developing applications, especially for verifying the efficiency of image segmentation configuration. Since it creates a significant increase in processing time, it should be disabled once the application is put into use.

Pixel Weight Image

The pixel weight image displays the relative greylevel value of weighted pixels when soft thresholding has been applied. All pixels outside the soft threshold range have a value of 0 or 1. The value of a weighted pixel varies from 0 to 1 in possible increments corresponding to the number of pixel values spanned by the soft threshold.

Blob Marker

The position of a detected blob is visually identified by a crosshair at the blob's center of mass. See [X-Y Position](#) further in this chapter. When soft thresholding is used, this center of mass takes into account the proportional contribution of weighted pixels in the blob's center of mass.

Blob Chain Code

The chain code of a detected blob is displayed as red path around the outside edges of the blob. The chain code is in fact a sequence of directions (up, down, left, right) that describes the contour or a blob, in Tool coordinate system only. See ***Chain Code Properties***.

Results

Blob Analyzer results are described here under the following eight categories of blob properties: general properties, perimeter properties, intrinsic inertia properties, extrinsic inertia properties, intrinsic bounding box properties, extrinsic bounding box properties, chain code properties, greylevel properties and topological properties.

General Properties

Area

Area results are returned in squared calibrated-units or pixels depending on the positioning mode applied to the Blob Analyzer.

X-Y Position

X-Y coordinates locate each blob according to its center of mass. The center of mass is defined by the average position of the pixels in the blob. This takes into account the effect of weighted pixel values, in the case of soft thresholding. In the Blob Results display, the center of mass is displayed as a crosshair and an index number. Position coordinates are referenced in the user-selected coordinate system: Tool, Object, Image or World. See the ***Coordinate Systems*** chapter.

Perimeter Properties

Raw Perimeter

Since blobs are grey-scale or color objects, their contour is constituted of pixels. The raw perimeter of a blob is defined as the sum of the pixel edge lengths on the contour of the blob. Although useful in many situations, the raw perimeter is sensitive to the orientation of the blob with respect to the pixel grid.

Convex Perimeter

The convex perimeter is more stable and accurate than the raw perimeter for convex shapes. Convex perimeter is calculated using an approximation of Crofton's theorem. The blob's diameter is determined by projections made at four different orientations: 0, 45, 90 and 135 degrees. The average diameter calculated from these projections is multiplied by PI to obtain the convex perimeter.

Roundness

The roundness of the blob expresses the degree of similarity between the blob and a circle. The roundness is 1 for a perfectly circular blob.

$$Roundness = \frac{4 \times \pi \times Area}{ConvexPerimeter^2}$$

Intrinsic Inertia Properties

Intrinsic Moments of Inertia

The intrinsic moments of inertia measure the inertial resistance of the blob to be rotated about its principal axes. Since their orientation depends on the coordinate system in which the blob is represented, the principal axes, major and minor, are defined in the section on extrinsic blob properties.

Elongation

The elongation expresses the degree of dispersion of all pixels belonging to the blob around its center of mass. The elongation of the blob is calculated as the square root of the ratio of the moment of inertia about the minor axis to the moment of inertia about the major axis.

$$Elongation = \sqrt{\frac{InertiaMaximum}{InertiaMinimum}}$$

Intrinsic Bounding Box Properties

The intrinsic bounding box defines the smallest rectangle, aligned with the principal axes, that can enclose the blob. The principal axes consist of an orthogonal XY coordinate system where X is the major axis and Y, the minor. The bounding box parameters are designated as Bottom (minimum Y coordinate), Left (minimum X coordinate), Top (maximum Y coordinate) and Right (maximum X coordinate).

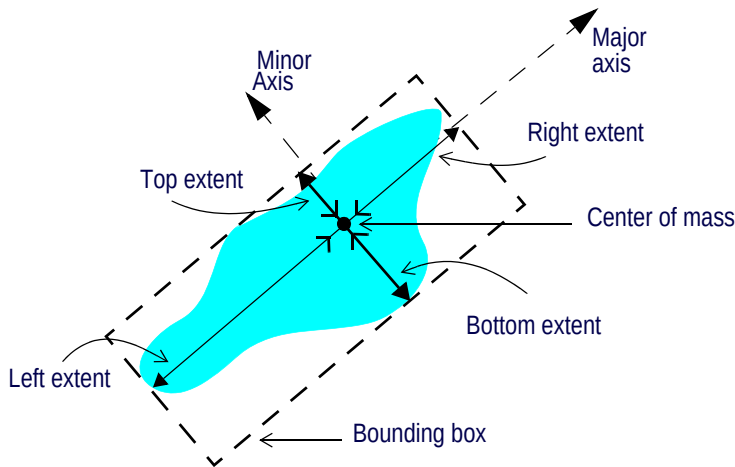


Figure 311 Intrinsic Bounding Box and Extents

Intrinsic Extents

Intrinsic extents are the distances between a blob's center of mass and the four sides of the intrinsic bounding box.

Rotation of the Intrinsic Bounding Box

The rotation of the intrinsic bounding box corresponds to the counter-clockwise angle between the X-axis of the bounding box (major axis) and the X-axis of the selected coordinate system.

Extrinsic Inertia Properties

Extrinsic Moments of Inertia

A moment of inertia of the blob is a measure of the inertial resistance of the blob to be rotated about a given axis. *Extrinsic* moments of inertia measure the moment of inertia about the x-y axes of the Tool coordinate system.

Principal Axes

Principal Axes designates a reference system that is constituted of the major axis and the minor axis. The major axis (X) is the axis about which the

moment of inertia is smallest. Conversely, the minor axis (Y) is the axis about which the moment of inertia of the blob is the greatest. See [Figure 312](#).

The principal axes are orthogonal and are identified by the angle between the selection Rectangle's X-axis and the blob's major axis.

Rotation of the Principal Axes

The rotation of the Principal Axes reference system is the counterclockwise angle between the X-axis of a selected coordinate system and the major axis. See [Figure 312](#).

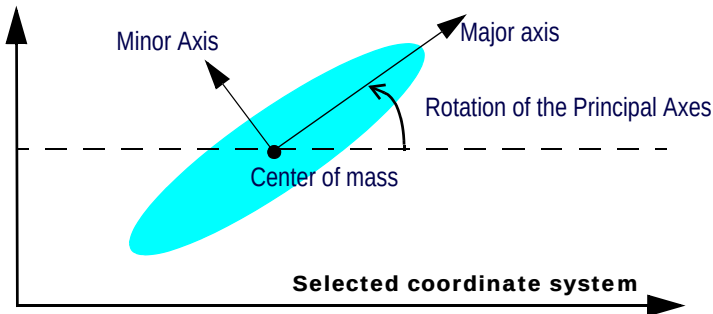


Figure 312 Rotation of the Principal Axes

Extrinsic Bounding Box Properties

The bounding box always defines the smallest rectangle that can enclose the blob. This bounding box is always aligned with the Tool coordinate system.

[Figure 313](#) presents an example of an extrinsic bounding box within its corresponding Tool coordinate system.

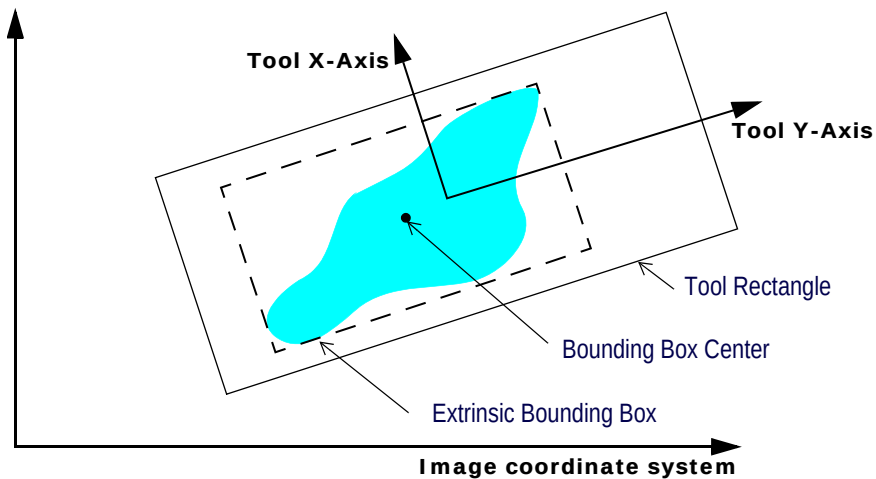


Figure 313 Extrinsic Bounding Box

Dimensions of the bounding box are given by its corresponding values in the selected coordinate system for:

- **Height:** The bounding box height
- **Width:** The bounding box width
- **Top:** The topmost Y value
- **Bottom:** The bottommost Y value
- **Left:** The leftmost X value
- **Right:** The rightmost X value

The position of the bounding box in the selected coordinate system is given by the X and Y coordinates of its center.

Rotation of the Extrinsic Bounding Box

The rotation of the extrinsic bounding box corresponds to the counter-clockwise angle between the X-axis of the bounding box and the X-axis of the selected coordinate system.

Extrinsic Extents

Extents of a blob are the distances between the center of mass and the four sides of the extrinsic bounding box. [Figure 314](#) illustrates a set of extrinsic extents.

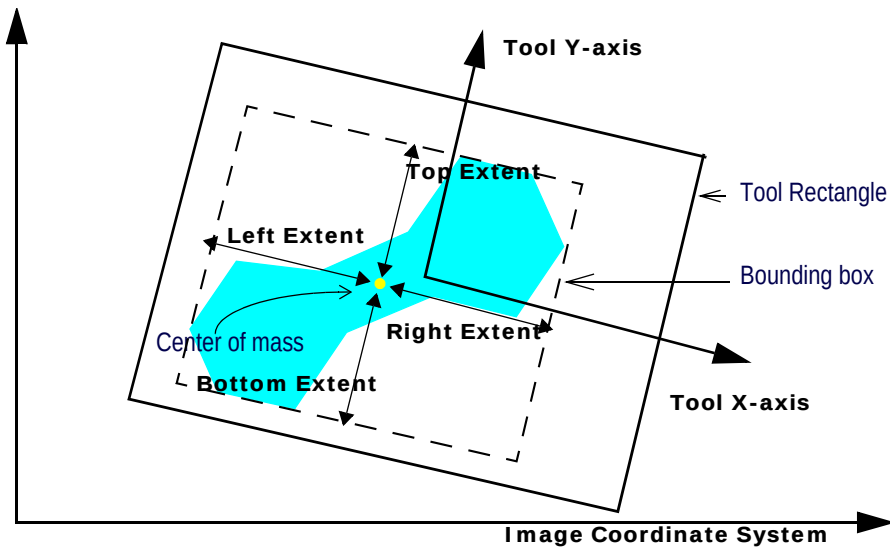


Figure 314 Extrinsic Extents

Chain Code Properties

A chain code is a sequence of direction codes that describes the boundary of a blob according to the Tool coordinate system only.

A chain code is characterized by the following elements.

- The start position of the chain code, which corresponds to the position of the first pixel associated to the chain code.
- Delta X and Delta Y designate the horizontal and vertical length of a boundary element in the chain code.
- The length of the chain code corresponds to the number of boundary elements in the chain code.

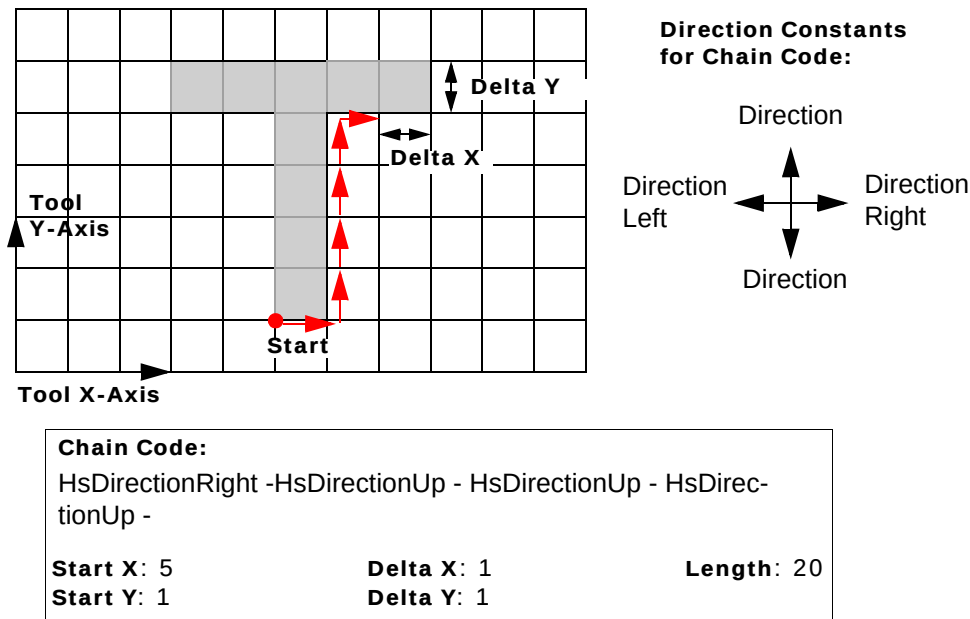


Figure 315 Chain Code Example

Greylevel Properties

In all cases, greylevel properties apply to pixels included in the blob regardless of weight values attributed by soft thresholding.

Mean Grey Level

The mean greylevel is the average greylevel of the pixels belonging to the blob.

Minimum Grey Level

The minimum greylevel is the lowest greylevel pixel found in the blob.

Maximum Grey Level

The maximum greylevel is the highest greylevel pixel found in the blob.

Grey Level Range

The greylevel range is the difference between the highest and the lowest greylevel found in the blob.

Standard Deviation Grey Level

The standard deviation of greylevels for the pixels belonging to the blob.

Topological Properties

Hole Count

The Hole Count property returns the number of holes found in each blob. The holes in smaller blobs that are contained within a larger blob are not included in the Hole Count. In other words, the Hole Count does not take into account the hierarchical relationship between blobs. [Figure 316](#) illustrates a case where Blob #1 returns a Hole Count of three, not four.

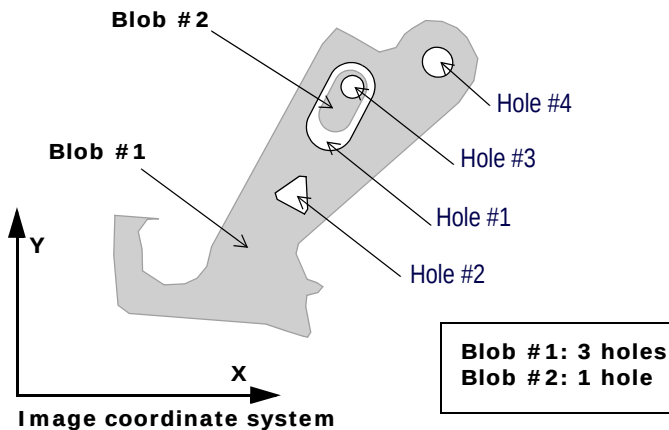


Figure 316 Blob Topology

Using the Blob Analyzer

Setup of the Blob Analyzer is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Blob Analyzer follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Blob Analyzer and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affects the required inputs.

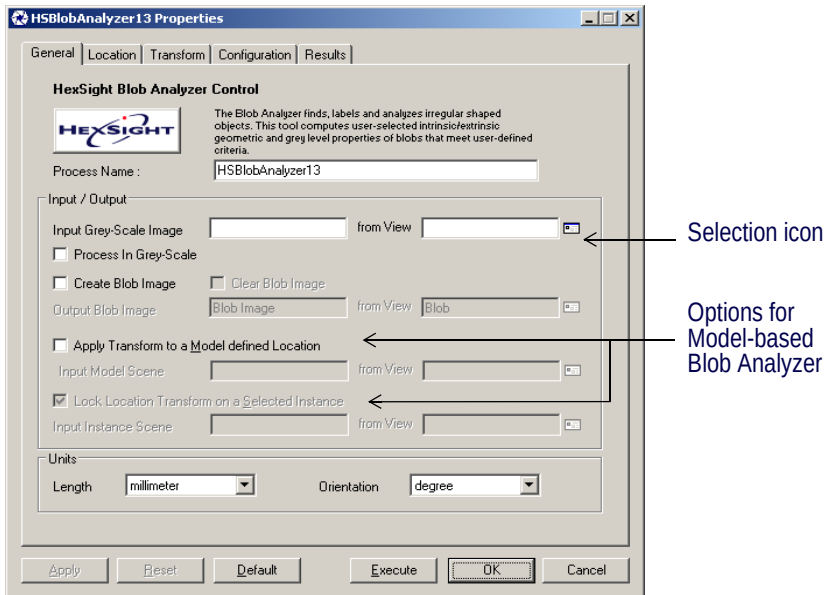


Figure 317 Blob Analyzer — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the ***Inspection Tools*** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 27 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 27 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSBlobAnalyzer#, where # is replaced by its rank number in the list of instantiated processes.

Inputs

Image

The **Image** is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Create Blob Image

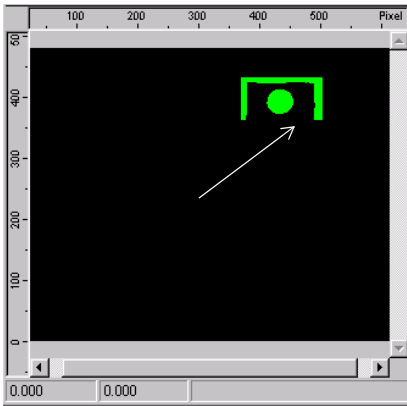
The **Create Blob Image** enables the Blob Analyzer to generate an output Image. Since outputting this image incurs a strong increase in the execution time; it should only be used as a visual aid when configuring the application .

- The Blob Image displays blob pixels in green and background pixels in black.
- The Blob Image will appear in the **Explorer** panel of the Application Control Interface as well as an overlay in the Blob Results display.

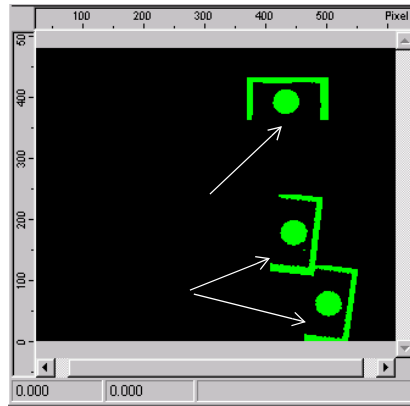
Clear Blob Image

This is enabled by default and ensures that only the last blob Image remains in the Blob Image output. When **Clear Blob Image** is disabled, the resulting blob Images for each execution of the application remain in the Blob Output Image. [Figure 318](#) illustrates the effect of the **Clear Blob Image** parameter on a blob output Image after three executions of the blob tool on different instances.

Clear Blob Image enabled



Clear Blob Image disabled

**Figure 318** Clear Blob Image Parameter

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

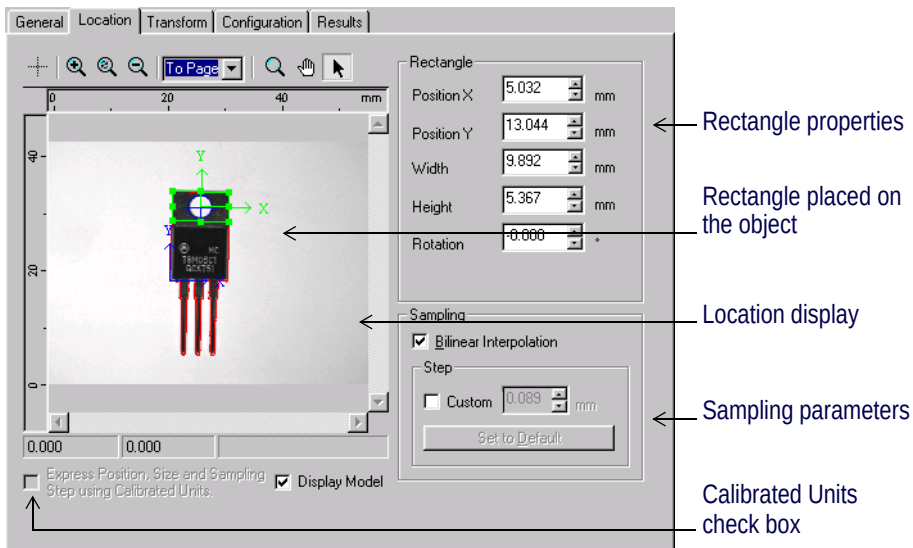


Figure 319 Blob Analyzer — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used

with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**. See the ***Inspection Tools*** chapter of this guide for more details on the Rectangle.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging the coordinate system marker at the intersection of the X-Y axes.

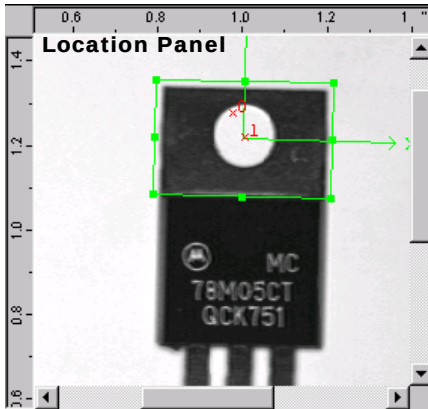
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Optimizing the Rectangle for Blob Analysis

For optimum blob analysis you should exclude as much background (non-blob areas) from the Rectangle as possible.

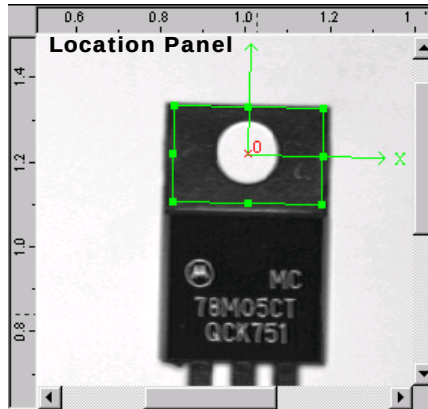
- Set the Rectangle so that it includes as little background as possible. Note that background areas can be difficult to differentiate in size from the blobs you want to find. See [Figure 320](#).
- The Rectangle must sufficiently exceed the area covered by the blob to allow for variations in the physical location of the blob as well as for possible positioning errors (object location).
- If the Rectangle area is too large, you will needlessly increase the tool's execution time.

Example A:
Rectangle slightly oversized



**Two blobs of similar size
detected**

Example B:
Rectangle properly sized



Only one blob detected

Figure 320 Sizing the Rectangle for Optimal Blob Analysis

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Bilinear interpolation provides a more realistic representation of pixel values that border and constitute blob contours.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

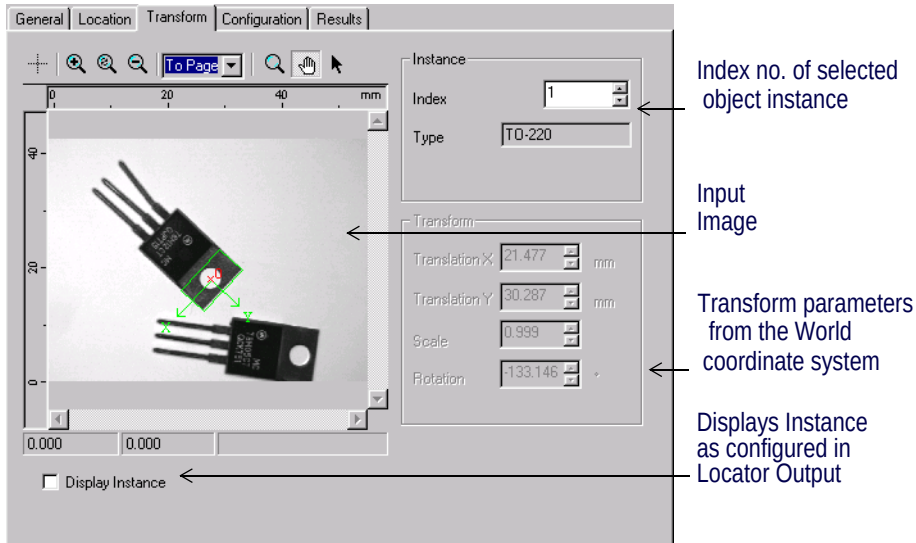


Figure 321 Blob Analyzer — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Each identified blob is identified by a crosshair marker at its center of mass and an instance number.

The status bar below the Image displays individual pixel greylevel values. These pixel values are useful in determining image segmentation thresholds in the **Configuration** panel.

Display Instance

Enable the **Display Instance** check box to activate a visual representation of the instance selected in the **Index** text box. This representation of the object is provided by the Instance Scene generated by the Locator. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.

- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest. When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- Type the transform parameters directly in the text boxes. You can also use the associated spin up/down buttons to fine-tune the values.

Configuration Panel

Use this panel to configure the parameters for determining blobs in the detected in the Rectangle region of interest.

Since the Blob Analyzer relies on differences in pixel greylevel values to divide the region of interest into blob and non-blob areas, efficient blob detection depends on the appropriate choice of a segmentation mode.

To configure a Blob Analyzer application, you will most often have to work back and forth between the thresholding function, the observable results in the **Blob Display** and the information in the **Results Panel**. The **Blob Display** and **Pixel Weight Display** provide important visual aids for verifying the effect of the selected segmentation algorithm.

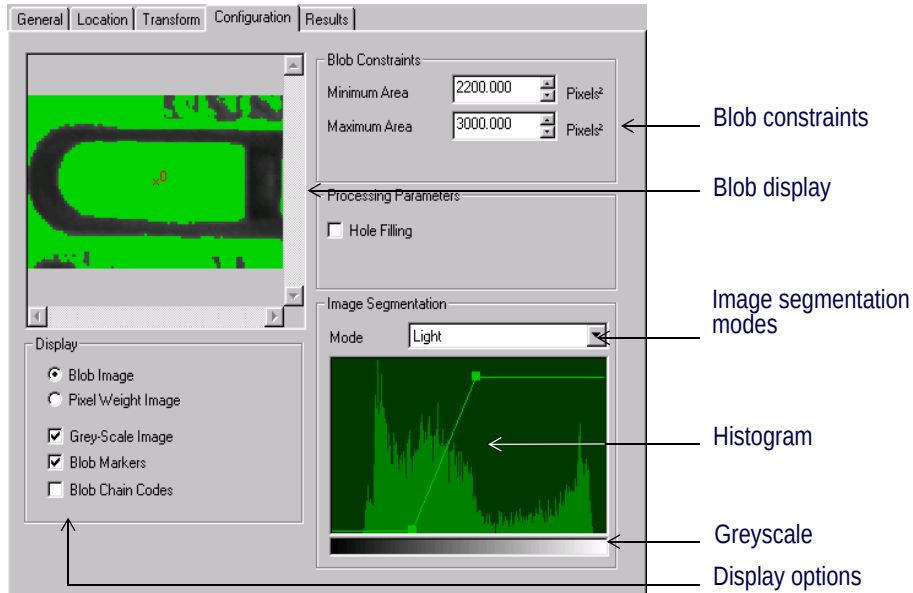


Figure 322 Blob Analyzer — Configuration Pane

As you make changes to the image segmentation function, click the **Execute** command button to view the effects within the Blob Display.

Histogram

The histogram shows the distribution of pixel values within the current Image. Blob analysis requires images with strong contrasts between blob and non-blob areas.

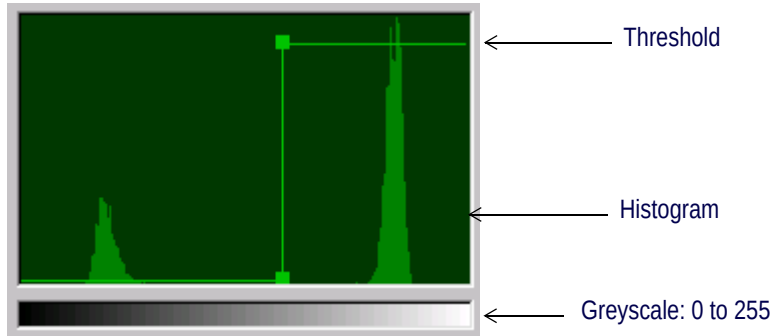


Figure 323 Bimodal Histogram

- Histograms used in blob detection should be bimodal, with sharp and well-distanced cusps.
- Bimodal distribution indicates a sharp difference between blob and non-blob areas within the Image.
- Distributions that do not indicate such strong contrasts may still be used for blob analysis, using appropriately configured soft or dynamic thresholds.
- The greyscale chart below the histogram visually indicates increasing greylevel values, from black (0) to white(255).

Blob Display

You can use the Blob **Display** to visualize the configuration and the results of the blob detection process. By default, this display shows the Image, sampled from the region of interest, upon which is overlaid the **Blob Image**.

- A crosshair markers identifies each blob's center of mass: two or more very different blobs may have centers of mass that are very close to each other.

Blob Image

The **Blob Image** mode displays two distinct zones:

- All background (non-blob) pixels are displayed in black.

- All blob pixels, including weighted pixels are displayed in green.

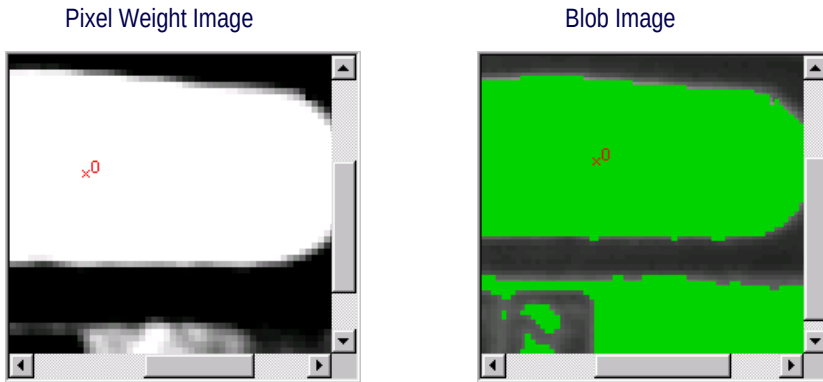


Figure 324 Pixel Weight Image vs. Blob Image

Pixel Weight Image

The **Pixel Weight Image** mode displays blob pixels in varying shades of grey.

- If you have applied a hard threshold, all blob pixels appear as white and all background as black.
- For soft thresholds, the range of grey shades depends on the range of greylevel values covered by the soft threshold.
- The lighter the shade of grey, the more a pixel contributes in value to the calculation of blob results: center of mass, area and perimeter.

Blob Chain Code

Enable the **Blob Chain Code** check box to display a representation of the chain code. The chain code for each detected blob is shown as a red contour line around the blob as shown in [Figure 325](#).

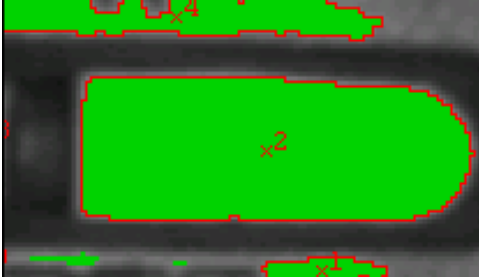


Figure 325 Blob Chain Code

Blob Constraints

Area constraints are useful for separating potential blobs from background regions, or from other blobs having similar pixel values. Enter or select the required values in the **Minimum Area** and **Maximum Area** spin-edit boxes.

- Area results are always displayed in calibrated units for Model-based applications.
- Image-based applications display the area in pixels unless you have enabled the **Calibrated Units** check box in the **Location Panel**.
- Leave a margin for variations in blob area depending on the PRECISION of the blob application you are building.

Processing Parameters

Checking the **Hole Filling** check box enables the Blob Analyzer to carry out a hole filling process on detected blobs.

All small blobs, and background will be included in the larger blob that surrounds them as illustrated in [Figure 326](#).

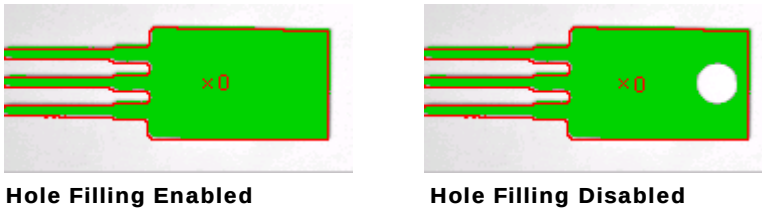


Figure 326 Hole Filling

Image Segmentation

The Blob Analyzer applies Image segmentation to separate pixels within the Rectangle into two categories: blob and non-blob areas. The results of this segmentation are displayed as either a Blob Image or a Pixel Weight Image. The segmentation mode you choose depends on the nature of the Images, typically provided by the Acquisition Device, and the relationship between blob data and background data.

Thresholds

Threshold functions set values at which image segmentation takes place. Depending on the segmentation mode selected, there may be either a single threshold or a double threshold. There are three types of threshold functions you can use for any of the four segmentation modes available: hard, soft and dynamic thresholds.

Hard Thresholds

Hard thresholds produce a blob image in which pixels have only two possible states: blob or non-blob.

- Background or non-blob pixels are each attributed a value of 0 in the Blob Image.
- Blob pixels are each attributed a value of 1 in the Blob Image.

Hard thresholding assumes that changes in data values occur at the boundary between pixels, without allowing a variation in pixels values across blob boundaries. Since this is rarely the case, soft thresholding is more often used for applications.

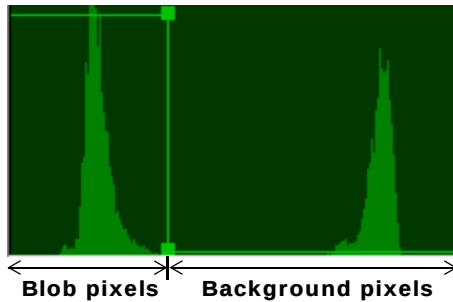


Figure 327 Hard Thresholding

Note: Hard thresholding is sometimes referred to as binary thresholding since all pixels can be considered as having one of two states: blob or background.

Soft Thresholds

Soft thresholds let you use blob detection in cases where boundaries of a blob region span a few pixels in width, with greylevels between blob and background.

A soft threshold is sloped and covers a range of pixel values that become weighted pixels once they are processed.

- Weighted pixels are used in the calculation of the blob's center of mass in proportion to their weighted value within the soft threshold range.
- Weighted pixels are shown as completely included within the Blob Image.
- Weighted pixels are displayed in various shades of grey, according to their threshold-determined value.

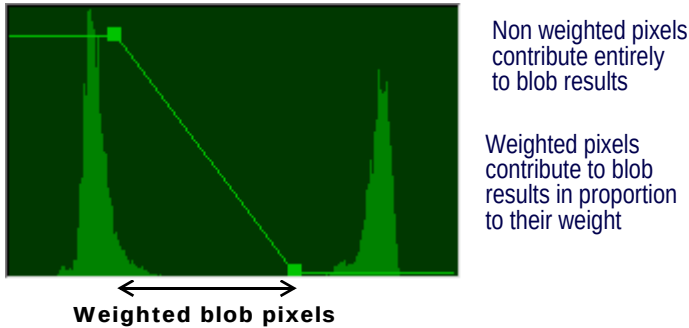


Figure 328 Soft Threshold

Dynamic Thresholds

Dynamic thresholds are particularly useful when there are lighting variations from one input Image to another.

- Set dynamic threshold as percentage of the total pixels in the Image.
- Use an “ideal blob”, as you might find within the Model Image, for example, to determine what percentage of the Image contains blob pixel values.
- Dynamic thresholding can be applied to any of the following options: light, dark, inside, outside.

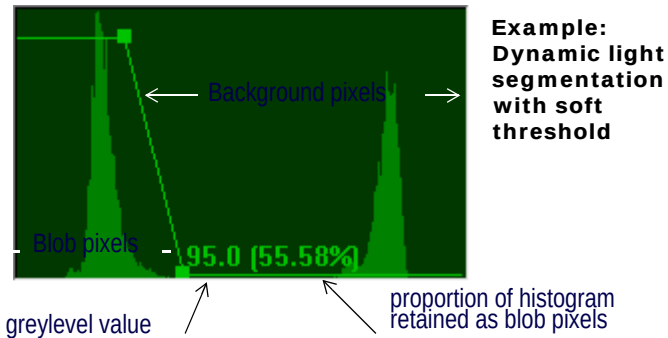


Figure 329 Dynamic Thresholding

Light Segmentation

The **Light** segmentation mode is used for light blobs on a dark object.

- All pixels whose values lie to the right of the threshold are potential blob regions.
- All pixels whose values lie to the left of the threshold are non-blob regions.
- Blobs include all pixels whose value is equal to the threshold value.

Dark Segmentation

The **Dark** segmentation mode is used for dark blobs on a light object. This is the inverse function of the **Light** segmentation mode.

- All pixels whose values lie to the left of the threshold are potential blob regions.
- All pixels whose values lie to the right of the threshold are non-blob regions.
- Blobs include all pixels whose value is equal to the threshold value.

Inside Segmentation

The **Inside** segmentation mode applies a dual threshold function. This mode is used for finding grey blobs within a background containing dark and light areas.

Examples:

- a) The Rectangle contains a grey blob on a light object or part, with a dark image background.
- b) The Rectangle contains a grey blob on a dark object or part, with a light image background.

Applications based on inside segmentation should always be configured using soft or dynamic thresholding functions.

Outside Segmentation

The **Outside** segmentation mode applies a dual threshold function and is the inverse of the **Inside** mode. This mode is used for finding dark and light blobs within a grey background.

Cases for using outside segmentation are not frequent because it is best to analyze dark and light blobs within an object by setting separate Rectangles for each and applying either a light or dark segmentation method.

Results Panel

The Results panel provides a grid of results and as a visual display of the results. The **Coordinate System** selector lets you choose the reference frame for Blob Analyzer results.

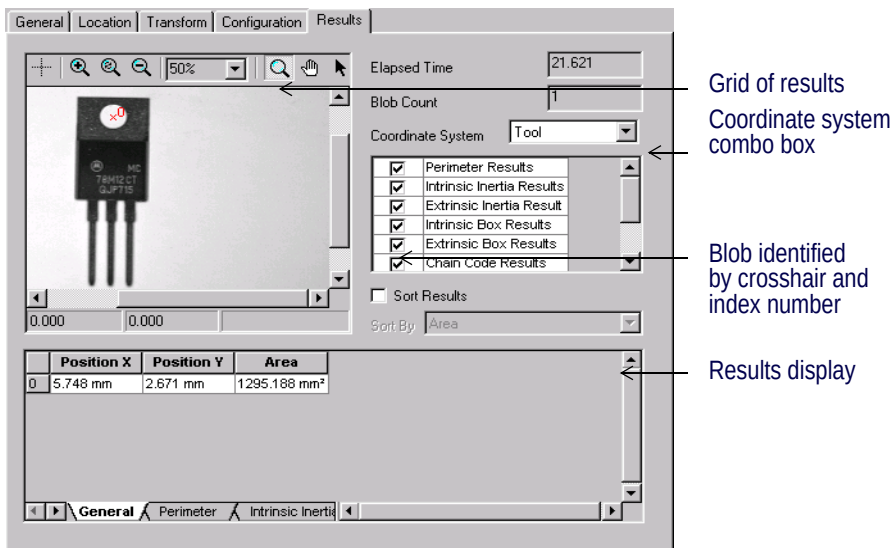


Figure 330 Blob Analyzer — Results Panel

Results Display

To help visualize the located blobs, the Results panel includes a visual display of the Rectangle. The Results display provides the Image of the current instance with each located blob identified either:

- a) by its index number and a crosshair at its center of mass, OR
- b) by an overlay, in green, of the blob area, if you enabled the **Blob Image Output** in the **General** panel.

Blob Count

Blob Count indicates the number of valid blobs that have been located by the tool.

Elapsed Time

Elapsed Time shows the total time needed for the Blob Analyzer to process and output results.

Enabling Blob Results

Given the number of possible blob results, the results are classified by categories. To optimize the tool execution time, you should enable only the result that you need for your application.

The tab-selected grid of results at the bottom of the Results panel, provides the results for each enabled category.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: Image, World, Object, Tool. Note also that the output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated. See the ***Understanding Blob Analyzer Concepts*** for more details on blob results.

Sort Results

By default, the table of results presents blobs indexed in the order in which they were located by the Blob Analyzer. To sort blobs:

- Enable the **Sort Results** check box.
- From the **Sort By** combo box select the blob property with which the Blob Analyzer will sort results.

Note: Blobs are sorted in descending order.

Description of Blob Results

The results returned by the Blob Analyzer process are listed below. Consult the ***Understanding the Blob Analyzer*** section of this chapter for more details on blob results.

General Results

- **Position X** and **Position Y** are the X-Y coordinates of the blob's center of mass with respect to the selected coordinate system.
- **Area** results are returned in calibrated units if Blob Analyzer is Model-based; if the Blob Analyzer is Image-based and the calibrated units check box is not enabled, the results returned are in pixels.

Perimeter Results

- **Roundness** is the degree of similarity between the blob and a circle. The roundness is for a perfectly circular blob is 1.
- **Convex Perimeter** is calculated from the average projected diameter of the blob and is more stable and accurate than the raw perimeter for convex shapes, including rectangular forms.
- **Raw Perimeter** results vary greatly in accuracy depending on the orientation of the blob with respect to the grid of pixels. Unless blobs are not convex, convex perimeter results provide greater accuracy.

Intrinsic Inertia Results

- **Inertia Minimum** is the moment of inertia about the major axis.
- **Inertia Maximum** is the moment of inertia about the minor axis.
- **Elongation** is the ratio of the moment of inertia about the blob's minor axis (Inertia Maximum) to the moment of inertia about the blob's major axis (Inertia Minimum).

Extrinsic Inertia Results

- **Inertia X-Axis** is the moment of inertia about the X-axis of the Tool coordinate system.
- **Inertia Y-Axis** is the moment of inertia about the Y-axis of the Tool coordinate system.
- **Principal Axes Rotation** is the angle of axis of the smallest moment of inertia with respect to the X-axis of the selected coordinate system.

Intrinsic Box Results

The intrinsic bounding box, which is aligned with the principal axes, defines the smallest rectangle enclosing the blob. The principal axes are defined with the minor axis and the major axis. Extents measure the distance between a blob's center of mass and the four sides of the bounding box.

- **Left** is the leftmost coordinate of the bounding box aligned with respect to the X-axis (major axis) of the principal axes.
- **Bottom** is the bottommost coordinate of the bounding box with respect to the X-axis (major axis) of the principal axes.
- **Right** is the rightmost coordinate of the bounding box aligned with the X-axis (major axis) of the principal axes.
- **Top** is the topmost coordinate of the bounding box aligned with the Y-axis (minor axis) of the principal axes.
- **Center X** and **Center Y** are the X and Y coordinates of bounding box center with respect to the principal axes. This does NOT correspond to the center of mass. See [Figure 331](#).
- **Height** is the height of the bounding box with respect to the Y-axis (minor axis) of the principal axes.
- **Width** is the width of the bounding box with respect to the X-axis (major axis) of the principal axes.
- **Extent Left** is the distance along the major axis between the blob's center of mass and the left side of the intrinsic bounding box

- **Extent Right** is the distance along the major axis between the blob's center of mass and the right side of the intrinsic bounding box.
- **Extent Bottom** is the distance along the minor axis between the blob's center of mass and the bottom of the intrinsic bounding box.
- **Extent Top** is the distance along the major axis between the blob's center of mass and the top of the intrinsic bounding box.
- **Rotation** is the orientation of the intrinsic bounding box with respect to the X-axis of the selected coordinates system.

Extrinsic Box Results

The bounding box defines the smallest rectangle, aligned with the Tool coordinate system, that can enclose the blob, as shown in [Figure 331](#).

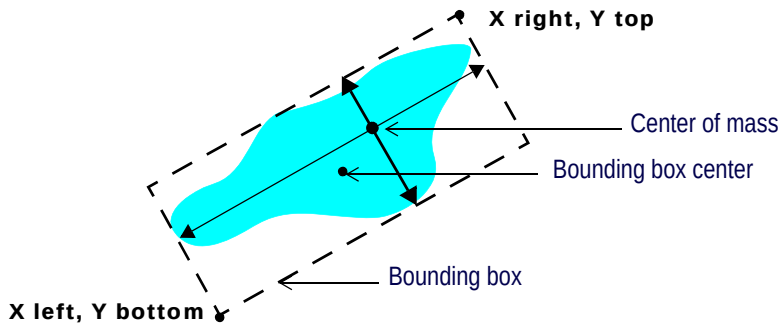


Figure 331 Bounding Box

- **Left** is the leftmost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system.
- **Bottom** is the bottommost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system.
- **Right** is the rightmost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system.

- **Top** is the topmost coordinate of the bounding box aligned with respect to the X-axis of the Tool coordinate system.
- **Center X** and **Center Y** are the X-Y coordinates of the center of the bounding box.
- **Height** is the of the bounding box with respect to the Y-axis of the Tool coordinate system.

Pattern Locator

The Pattern Locator finds and locates instances of a pattern occurring within an Image.

Pattern Locator Chapter Outline

[Understanding Pattern Locator Concepts](#)

[What is a Pattern?](#)

[Pattern Detection](#)

[Using the Pattern Locator](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Pattern Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Pattern Locator Concepts

The Pattern Locator is best suited for applications that require the detection of low contrast or rather small features such as letters, numbers, symbols and logos on a part. Note that patterns that can provide well-defined contours can most often be modeled and found by the Locator tool.

Typical cases for using the Pattern Locator include:

- Detecting the presence/absence of a pattern on a modeled object (Locator).
- Finding the rotation of the pose of a contour-symmetric object using grey-scale features on the object.
- Disambiguating objects having the same contours by their grey-scale features. (See the Regulator Inspection example)

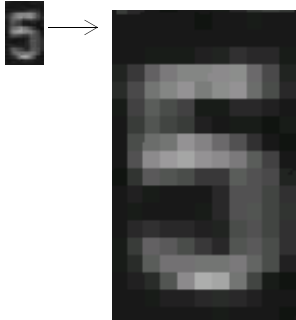
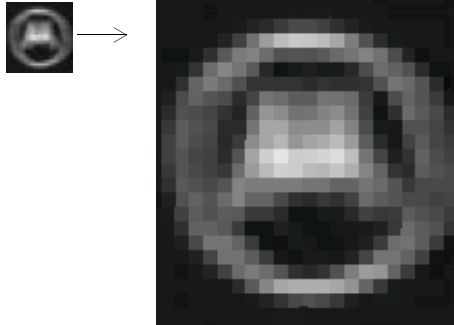
As is the case for other HexSight inspection tools the Pattern Locator is best used with model-based positioning.

The search algorithm of the Pattern Locator has been significantly improved. It still does not support rotated patterns and should generally be used as a model-based inspection tool for detecting the presence of patterns on object instances that have been located using the contour-based HexSight Locator. The main improvements are:

- The Pattern Locator now uses a much faster multi-resolution grey-scale correlation pyramidal search algorithm.
- For simple non-model-based applications that use grey-scale correlation, model patterns can be loaded and saved directly into files without having to use Locator models.

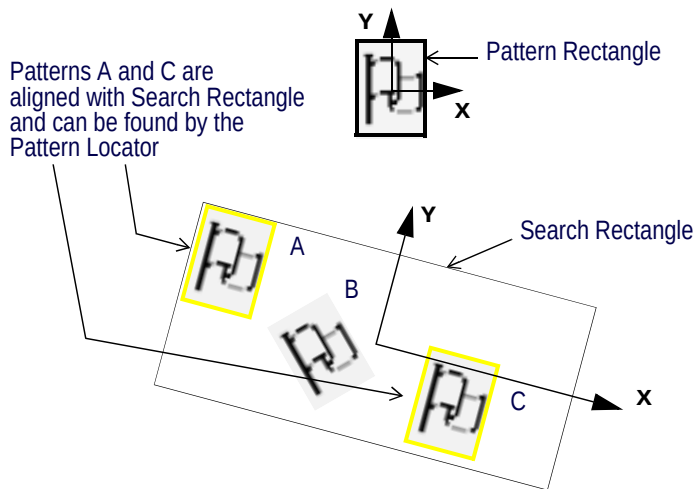
What is a Pattern?

A pattern is defined as an $n*m$ grid of pixels having a specific arrangement of greylevel values. In the Pattern Locator interface, a reference, or “sample”, pattern is defined within the Pattern Rectangle.

Example A:
number '5' pattern**Example B:**
'Logo' pattern**Figure 332** Pattern Examples

Pattern Rectangle

The **Pattern Rectangle** bounds and defines the actual pattern being sought by the Pattern Locator tool. Once the pattern rectangle is defined and the pattern image is present, the pattern is learned on the next execute of the process.

**Figure 333** Pattern Orientation

The axes marker of the Pattern Rectangle sets the orientation of the pattern. When the tool searches for pattern instances, it considers the X-Y axes of the pattern as aligned with the X-Y axes on the Search Rectangle that defines the area within which the pattern is being sought as illustrated in [Figure 333](#).

The sampling parameters, bilinear interpolation and sampling-step, that are configured for the **Search Rectangle** are also applied to the **Pattern Rectangle**.

Note: The Axes marker defines the orientation of the pattern; it does not define a new coordinate system nor a frame of reference.

Pattern Image

The source image, upon which the pattern is defined, is called the Pattern Image. Each instance of a Pattern Locator tool can use one, and only one, Pattern Image.

If the tool is model-based, the default pattern image is provided by the Model image. This default input can be overridden to select another image upon which you define the pattern.

Besides the Model Image, the Pattern can be defined on one of the following:

- **The current input image:** This image is then stored to the hard disk. Once the application is terminated this image remains on the hard disk for further use. However the position of the Pattern Rectangle in this image must be redefined when you restart the application.
- **Image from a Pattern Database:** A Pattern Database is in fact the Pattern Image. The position of the Pattern Rectangle is not stored to the database and must be redefined when the Pattern Database is loaded to the application.
- **An image from any other HexSight database:** You have the option of selecting any single image from a chosen HexSight database file (*.hdb) and then defining a Pattern Rectangle within this image. This image can be saved as a new Pattern Database for further use.

Selection of Efficient Patterns

Pattern Area Size

The Pattern Rectangle should include the minimum information required to define and characterize the pattern.

- Not only will a large pattern area be longer to process, it may often result in false detections.
- The minimum size of a pattern is fixed as 3X3 pixels.

Symmetry

If a pattern is symmetrical, it can be found but its orientation will be ambiguous.

- For example, in [Figure 334](#), a large part of the pattern consists of a circular, and thus symmetrical contour. Also, a large part of the inner “a” pattern can also be identified at any rotation. Because of this rotational symmetry, the Pattern Locator cannot disambiguate the various poses of the pattern.

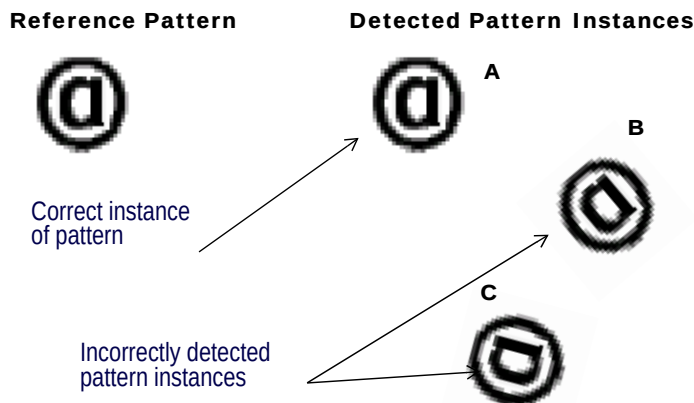


Figure 334 Detecting Symmetric Patterns

Rotation

Pattern detection does not compensate for rotation. The Pattern Locator searches for pattern instances as having an orientation that matches the orientation of the reference pattern defined in the Pattern Rectangle.

Pattern Detection

The basic pattern detection strategy is to search for pattern instances in the area defined in the **Search Rectangle** step-by-step. In each step, an area of the same size as the reference pattern is compared to the superimposed **Pattern Rectangle**. This procedure is carried out by sliding the pattern sample in both X and Y directions until the entire search area has been investigated.

- To reduce the time required by a pixel-by-pixel search of the entire image, the Pattern Locator applies a multi-resolution search strategy.

Search Rectangle

The Search Rectangle bounds the region of interest within which the process searches for instances of the reference pattern. Since pattern location can be a lengthy process, restricting the search for pattern instances to a specific area of the input Image greatly reduces the tool's execution time.

Multi-Resolution Search

In its multi-resolution search strategy, the Pattern Locator carries out a search at each defined resolution level. Basically, the coarser resolution levels (**Search Coarseness**) are used to generate hypotheses; the higher resolution levels (**Positioning Coarseness**) are used to refine the position of validated pattern instances. Only those patterns that meet the **Match Threshold** are retained.

Subsampling

Subsampling is applied to both the sample pattern and the search area.

- Subsampling can advantageously reduce the execution time of the application, particularly for large patterns. This is especially true when high accuracy is not of primary importance.
- Moreover, a sampling step that is too high can result in false pattern instances.

Search Coarseness

The Pattern Locator first conducts a search at the coarsest Search Coarseness (**Quick**) resolution of and generates a set of possible matching patterns hypotheses. It then repeats the search process at the next lowest sampling level, conducting the search on only those areas of the image close to instances from the previous set of pattern matching hypotheses. This process is repeated until the lowest search coarseness (**Exhaustive**) is reached. Note that lowest **Search Coarseness** level corresponds to the highest **Positioning Coarseness** level.

Positioning Coarseness

Using the same strategy as for the search, the Pattern Location refines the accuracy of the pose of the pattern, starting from the coarsest (**Fast**) and finishing at the most refined (**Accurate**) **Positioning Coarseness** level.

Pattern Detection Constraints

Pattern detection parameters limit the number and the quality of pattern hypotheses found during the search.

Instances To Find

Instances to Find determines the maximum number of patterns that can be searched for.

- To optimize search time you should set this value to no more than the expected number of instances.
- When the actual number of instances exceeds the **Instances to Find** value, the Pattern Locator stops once it attains the set value.

- If the number of patterns present in the input image exceeds the constraint, the Pattern Locator retains only the best pattern instances.

Match Threshold

The **Match Threshold** sets the minimum required **Match** value required to consider a detected pattern as valid.

Automatic Coarseness Selection

When **Automatic Coarseness Selection** check box is enabled, the Pattern Locator automatically optimizes the **Search Coarseness** and **Positioning Coarseness** values.

- **Automatic Coarseness Selection** is the recommended mode.
- To manually set the **Search** and **Positioning Coarseness** parameters you must disable the Automatic Level check box.

Using the Pattern Locator

The Pattern Locator setup is carried out on six tab-selected panels titled: General, Location, Transform, Pattern, Configuration and Results. Steps for using and setting up the Pattern Locator follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance for the Pattern Locator and enter the tool's input Images and Views.

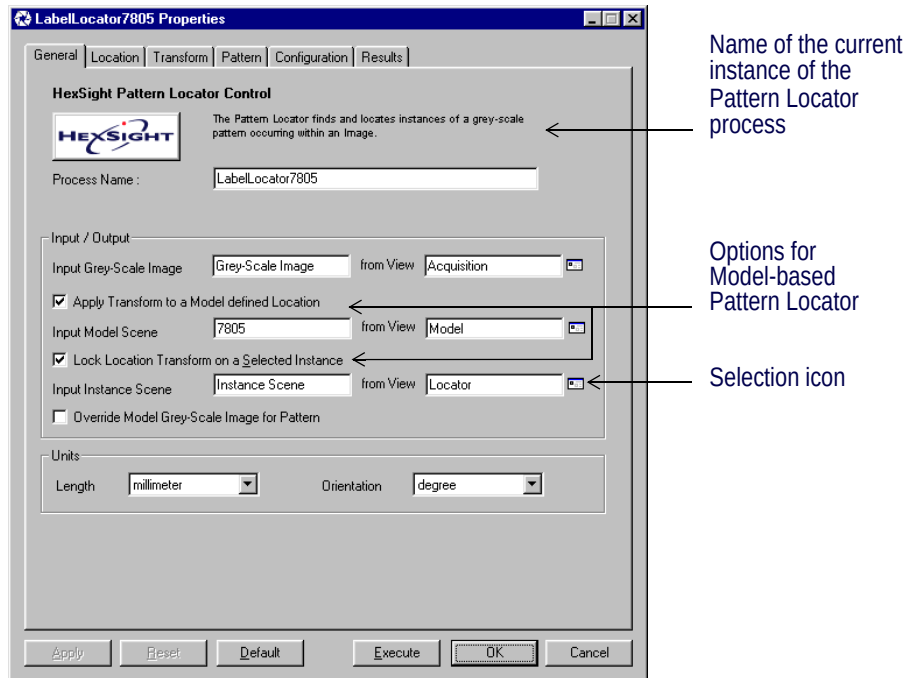


Figure 335 Pattern Locator — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 28 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 28 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSPatternLocator#, where # is replaced by its rank number in the list of instantiated processes.

Inputs

Image

The **Image** is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Override Model Image for Pattern

By default, if the tool is model-based, the pattern is defined on the model image. Enabling the **Override Model Image for Pattern** allows the pattern to be defined on another image.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

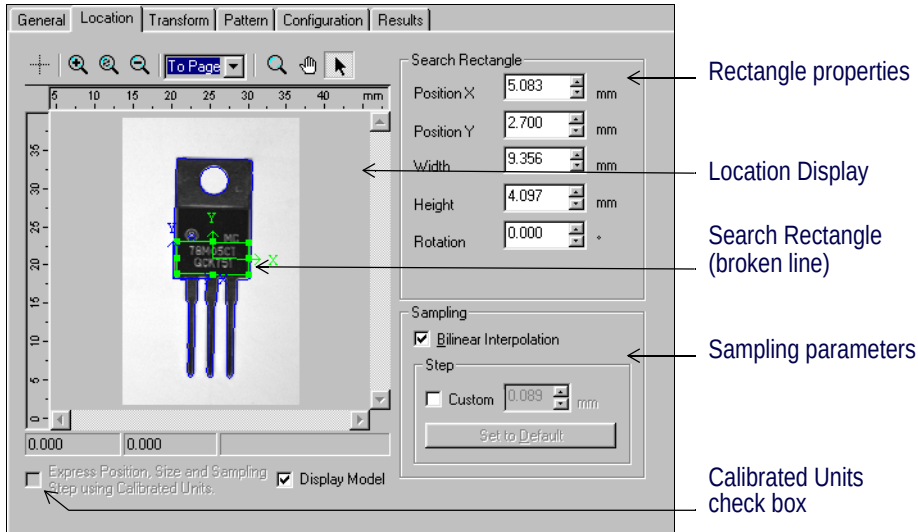


Figure 336 Pattern Locator — Location Panel

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

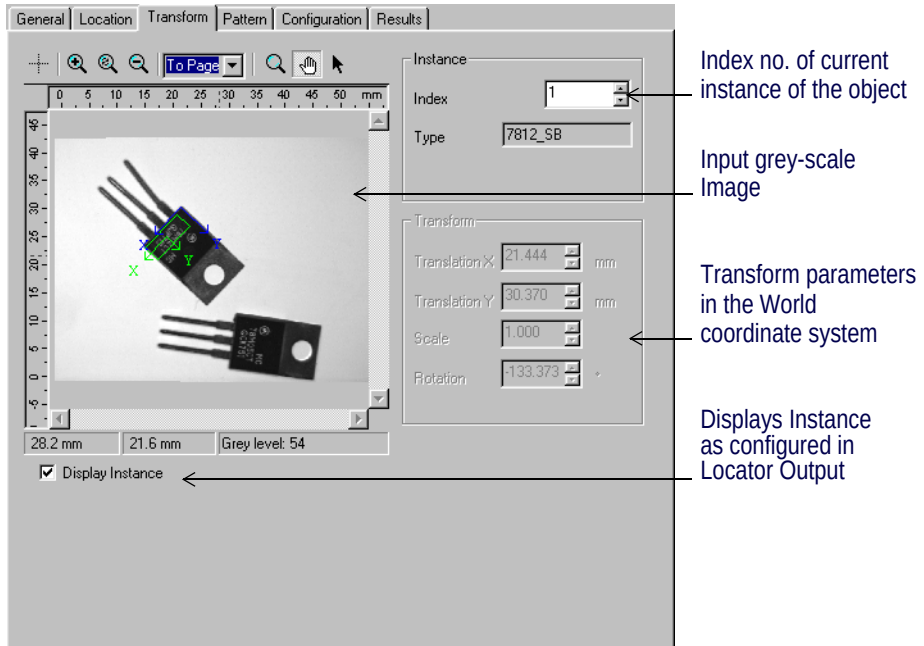


Figure 337 Pattern Locator —Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Pattern Rectangle region of interest, transposed according to the transform parameters. Each located pattern instance is indicated by a crosshair marker.

Display Instance

Enable the **Display Instance** check box to activate a visual representation of the instance selected in the **Index** text box. This representation of the object is provided by the Instance Scene generated by the Locator. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an instance of object in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Pattern Panel

Use this panel to position the **Pattern Rectangle** that defines the sample pattern. This Pattern Rectangle appears in the display as a green rectangle.

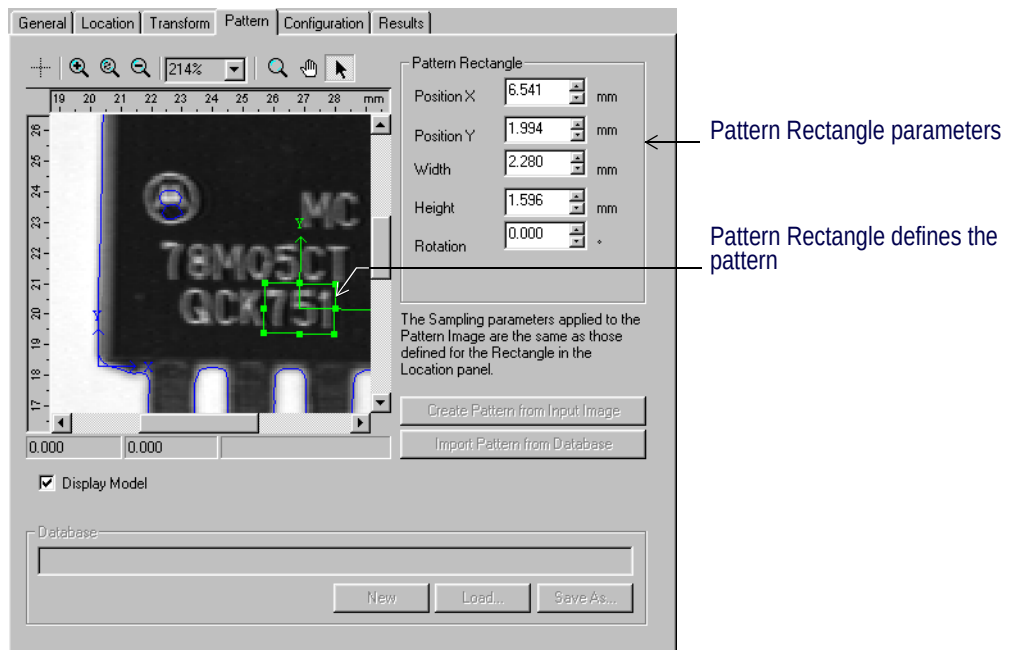


Figure 338 Pattern Locator — Pattern Panel

Pattern Rectangle

The Pattern Rectangle bounds the pattern you want the Pattern Locator to find within the Image. This Pattern Rectangle appears as a green box containing a coordinate system marker. The Pattern Rectangle should be just large enough to encompass the pattern.

To place and size the Pattern rectangle:

- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.
- You can also change height, width and rotation by entering values in the appropriate boxes.

Pattern Display

By default, if the pattern is Model-based, the pattern is defined on the Model Image and this image that appears in the display. If you enabled **Override Model Image for Pattern** check box in the General Panel or if the Pattern Locator is NOT Model-based, you must specify the source image on which you will define the Pattern.

Create Pattern from Input Image

Create Pattern from Input Image opens and displays the current input image for defining the Pattern Rectangle.

- The position of the Pattern Rectangle in the input image is not saved in the database but its position is saved in the application's configuration file (*.cfg).

Import Pattern from Database

Import Pattern from Database allows you to open a HexSight database and retrieve the image on which you will define the pattern

- If you open a database containing more than one image you must specify which image you will use.
- The position of the Pattern Rectangle in the image is not saved in the database but its position is saved in the application's configuration file (*.cfg).

Display Model

When **Display Model** is enabled, located object instances are represented by their associated Model, including the coordinate system marker.

Database

The **Database** box provides the name and path of the database that provides the imported image.

Configuration Panel

This section will guide you through the configuration of parameters applied by the Pattern Locator to detect and locate pattern instances within an image.

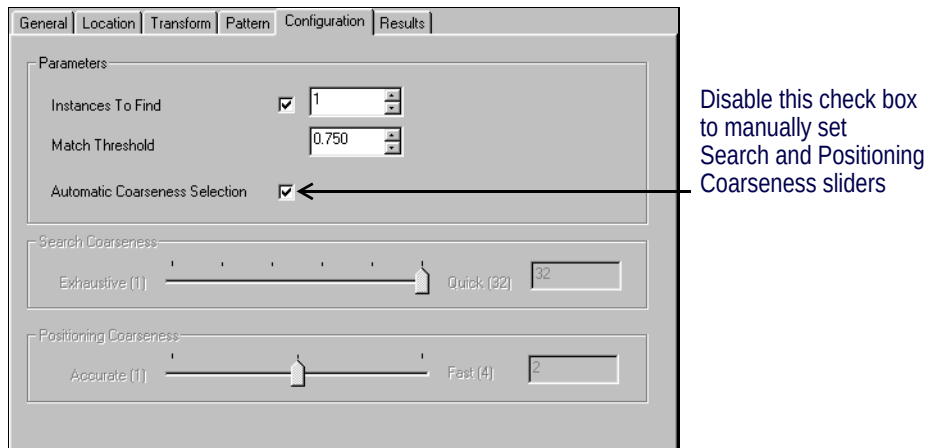


Figure 339 Pattern Locator — Configuration Pane

Instances to Find

Instances to Find determines the maximum number of instances that the Pattern Locator will search for.

- You should set this value to no more than the expected number of instances.

Match Threshold

Match Threshold sets the minimum Match strength required for a Pattern to be recognized as valid. A perfect match value is **1**.

- If the match threshold is too high, many pattern instances may be rejected.
- If the match threshold is too low, too many false pattern instances may be detected.

Automatic Coarseness Selection

By default, Automatic Coarseness Selection is enabled, in which case the Pattern Locator automatically determines and sets the values for **Search Coarseness** and **Positioning Coarseness**.

- To manually set the Search and Positioning coarseness values you must disable the Automatic Coarseness Selection check box.

Search Coarseness

Search Coarseness levels are used to generate pattern hypotheses in the input image. The **Search Coarseness** value ranges from 1 (**Exhaustive**) to 32 (**Fast**). This value is higher than or equal to the **Positioning Coarseness** value.

Positioning Coarseness

Positioning Coarseness levels are used to confirm pattern hypotheses and refine their pose. The **Positioning Coarseness** value ranges from 1 (**Accurate**) to 4 (**Fast**). This value is lower than or equal to the **Search Coarseness** value.

Results Panel

The Results panel provides a grid of results and a visual display of the results. The **Coordinate System** selector lets you choose the reference frame for Pattern Locator results.

Grid of Results

The results returned by the Pattern Locator process are listed below.

- The leftmost column shows the **Index** number of located pattern instances, starting at 0.
- The **Match** value ranges from **0** to **1**, with **1** being the best quality. A value of 1 means that 100% of the reference pattern was successfully matched to the found pattern instance.
- **Position X** shows the X coordinate of the center of the Pattern Rectangle with respect to the selected coordinate system and units.
- **Position Y** shows the Y coordinate of the center of the Pattern Rectangle with respect to the selected coordinate system and units.
- The Pattern Locator does not provide the actual rotation of the pattern. A **Rotation** result is output only for a model-based Pattern Locator, in which case the rotation is that of the Search Rectangle with respect to the selected **Coordinate System**.

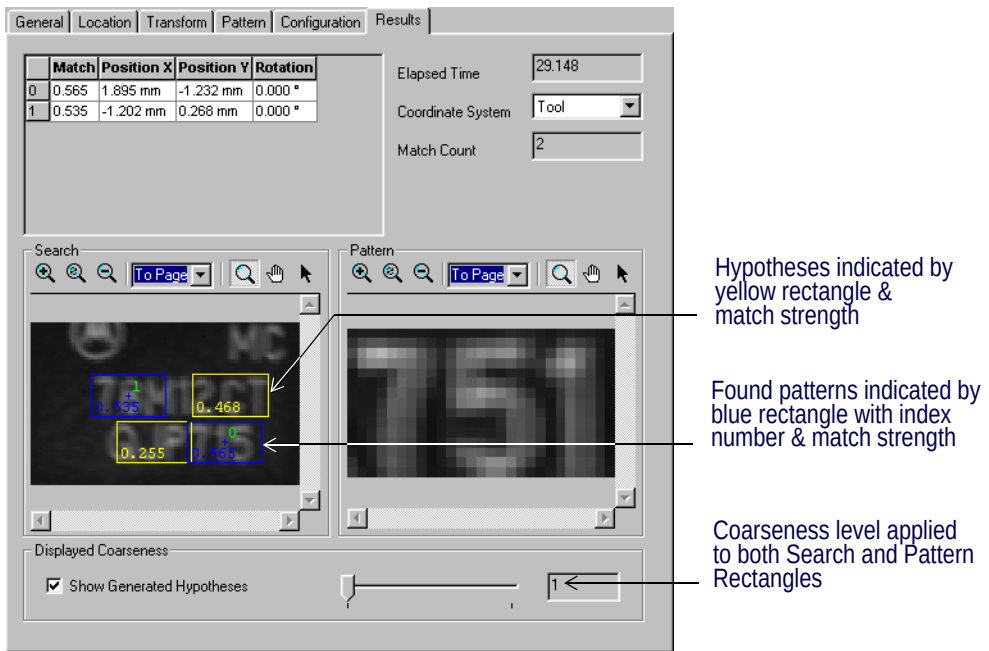


Figure 340 Pattern Locator - Results Panel

Elapsed Time

Elapsed Time shows the total time needed for the Pattern Locator to process and output results.

Coordinate System

Use the **Coordinate System** combo box to select the coordinate system in which the results are output. The available coordinate systems are: Image, World, Object, Tool.

Match Count

Number of instances of the pattern found in the last execution.

Results Displays

The Results displays provide a visual representation of the Pattern Locator results. The **Displayed Coarseness** slider lets you visualize the contents of both the **Search** and **Pattern** rectangles at the selected coarseness value.

Search Display

The Search display show the area of the input image bounded by the **Search** rectangle.

- When **Show Generated Hypotheses** is enabled, rejected pattern hypotheses are displayed in as yellow rectangles.
- Located pattern instances are shown as blue rectangles, along with their **Match** strength value.

Pattern Display

The **Pattern** display provides a visual display of the pattern defined by the **Pattern Rectangle**.

Color Finder

The Color Finder Tool searches analyzes images to find areas of color that match user-defined filters.

Typically, this tool is used to verify and analyze an area on an object for the purpose of verifying if the object meets defined color criteria.

Color Finder Chapter Outline

[Understanding Color Finder Concepts](#)

[Image Sampling](#)

[Color Finder Results](#)

[Using the Color Finder](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Color Finder Concepts

The Color Finder includes the following configuration controls, which can be found on the Configuration panel.

Image Sampling

By default, Image Sampling is set to 1, which means that the image is not subsampled, and each pixel in the image is processed by the Color Finder Tool, for every filter.

Color Finder Results

Color Filters

The Color Finder Tool analyzes the region of interest by applying all the defined filters to the image within the region of interest. Any number of filters can be added to the Color Finder.

Color Values

The value of a filter can be configured either by its HSL values or its RGB values.

Defining a Color by its RGB values

RGB refers to a way mode of describing, or quantifying a colors by its red (R), green (G), and blue (B) values.

Defining a Color by its HSL Values

HSL refers to a way mode of describing, or quantifying colors by their hue (H), saturation (S), and luminance (L) values.

Color Tolerances

The color filter accepts any color values that are within defined tolerances. Tolerance values can only be expressed in HSL values. The tolerance range for each H, S, and L tolerance is applied to the defined color.

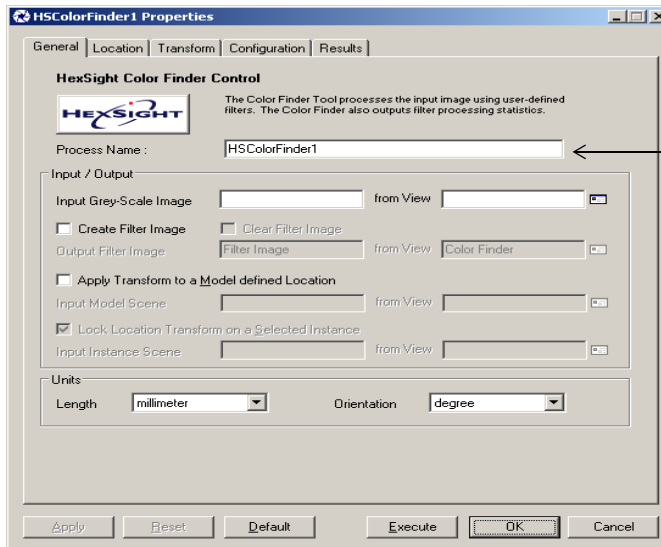
Using the Color Finder

Setup of the Color Finder is carried out on five tab-selected panels: General, Location, Transform, Configuration and Results. Steps for using and setting up the Color Finder will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Color Finder and enter the tool's input and output Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.



Name of the
current instance
of the Color Finder
tool

Figure 341 Color Finder — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the ***Inspection Tools*** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes

Table 29 Positioning Modes: Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 29 Positioning Modes: Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Output Filter Image

Output Filter Image is the Image that will be created in the runtime database by the tool. Since this is an output Image, it will overwrite the selected Image in the runtime database when the tool is executed.

- Type the name of the Image in the **Output Filter Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This

parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Locator. This Scene is provided by a HexSight Locator configured to output its Instance Scene in the runtime database. See the **Locator** chapter of this guide for more details.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

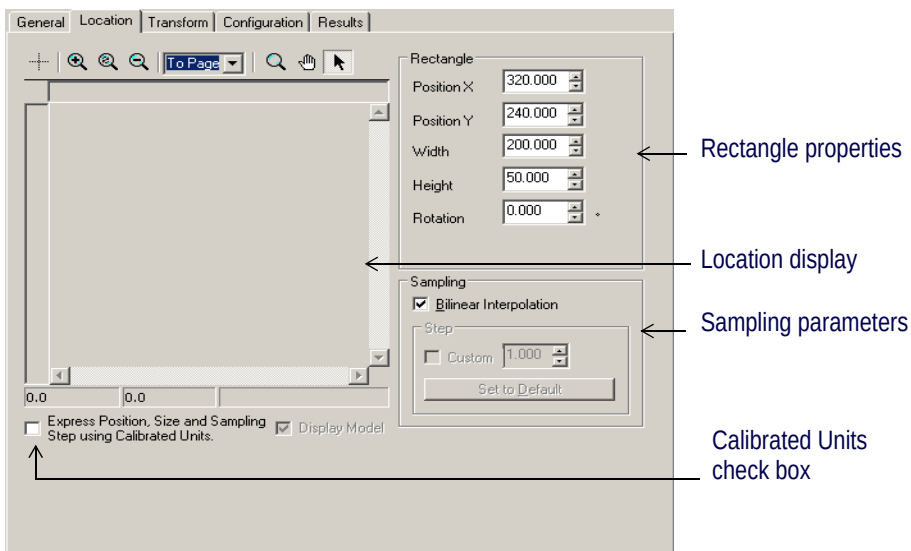


Figure 342 Color Finder — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

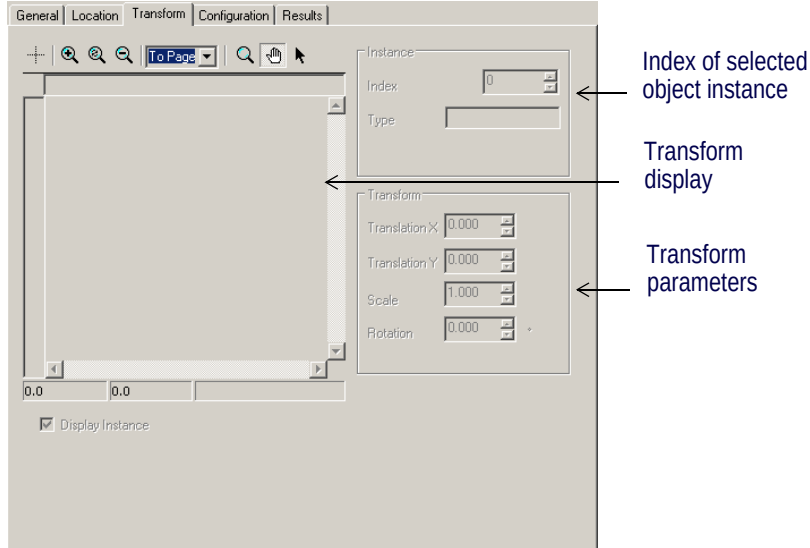


Figure 343 Color Finder — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest. When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

This section will guide you through the configuration of parameters applied by the Color Finder to detect and locate color matches within an image.

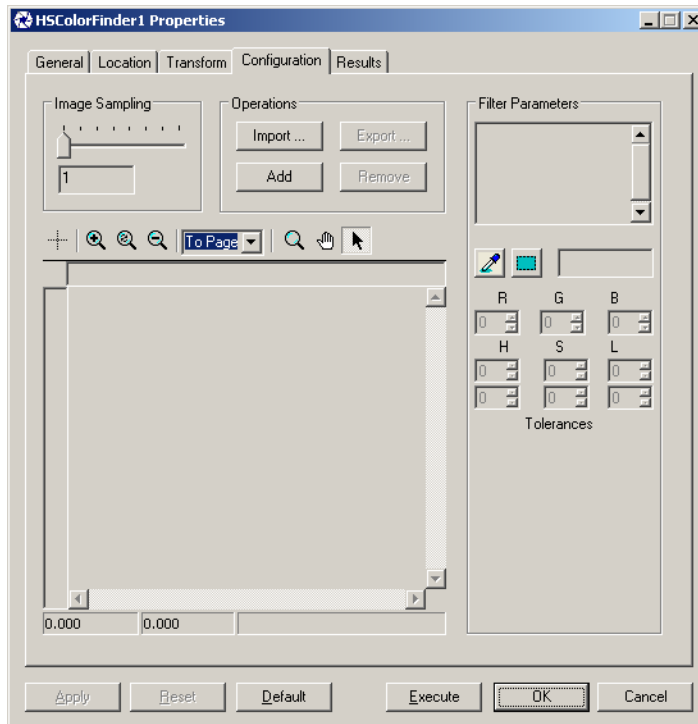


Figure 344 Color Finder — Configuration Pane

Image Sampling

By default, **Image Sampling** is set to 1, which means that the image is not subsampled, and each pixel in the image is processed by the Color Finder Tool, for every filter.

- Increasing the Image Sampling factor reduces the number of pixels and the quantity information analyzed by the tool. With a subsampling factor of 2, the image in the region of interest is subsampled in tiles of 2x2 pixels. With a subsampling factor of 3 the image is subsampled in tiles of 3x3 pixels, and so forth.

- Increasing the Image Sampling value may reduce the execution time but affects the accuracy of color finder results. This may be advantageous in some applications, particularly when the region of interest is quite large, when many filters are configured for the tool, and when color results are acceptable at resampled levels.

[Figure 345](#) illustrates the results of a color filter tool at different sampling levels.

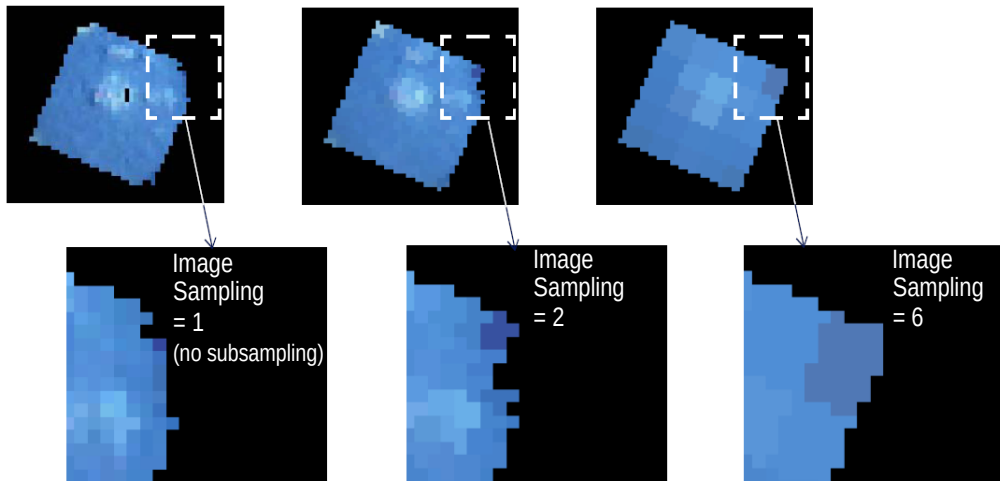


Figure 345 Effect of Image Sampling

Color Filters

The Color Finder Tool analyzes the region of interest by applying all the defined filters to the image within the region of interest. Any number of filters can be added to the Color Finder.

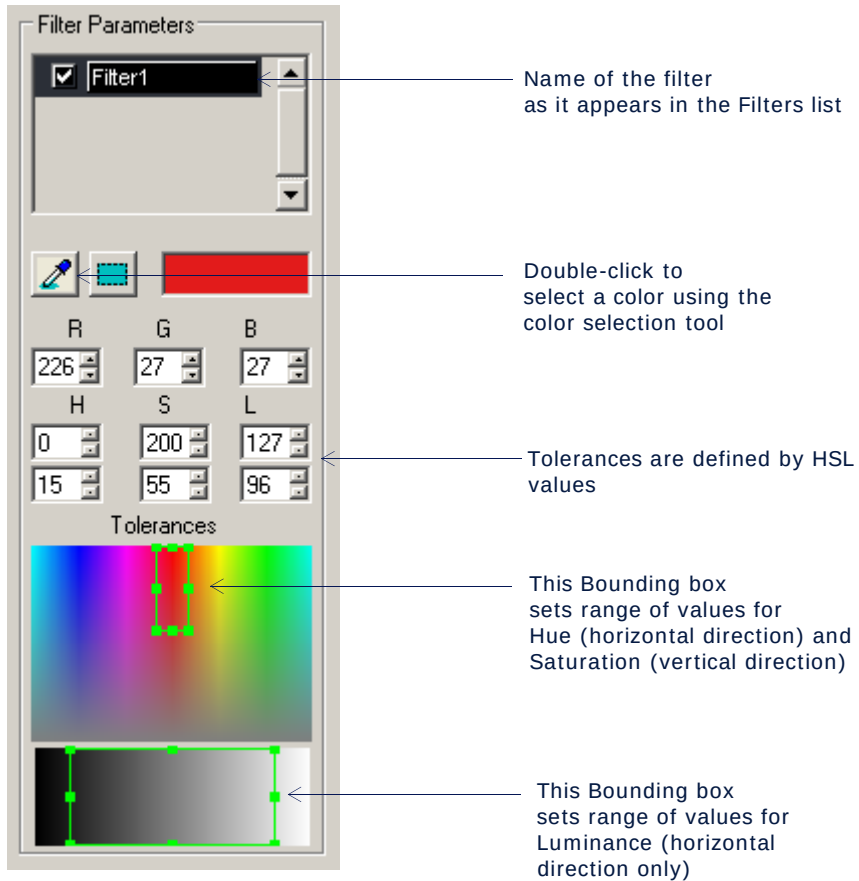


Figure 346 Configuring a Color Filter in the Filter Editor Window

Color Values

The value of a filter can be configured either by its HSL values or its RGB values.

[Table 30](#) lists a few colors with their corresponding RGB and HSL values

Table 30 RGB and HSL Values for some Common Colors

Color Name	RGB Values	HSL Values
White	255,255,255	0,0,255
Black	0,0,0	0,0,0
Middle Grey	127,127,127	0,0,128
Red	255,0,0	85,255,128
Green	0,255,0	0,255,128
Blue	0,0,255	170,255,128

Defining a Color by its RGB values

RGB refers to a way mode of describing, or quantifying a colors by its red (R), green (G), and blue (B) values.

- Value for R,G, and B range from 0 (no color) to 255 (maximum color).
- Pure white is defined by (R,G,B) = (255,255,255)
- Pure black is defined by (R,G,B) = (0,0,0)
- Table shows the some common colors expressed in RGB and HSL

Defining a Color by its HSL Values

HSL refers to a way mode of describing, or quantifying colors by their hue (H), saturation (S), and luminance (L) values.

Hue - H

Hue is the quality of color that we perceive as the color itself, for example: red, green, yellow. The hue is determined by the perceived dominant wavelength, or the central tendency combined wavelengths, within the visible spectrum.

- Hue values range from 0 to 255. These values correspond to a displacement along the color spectrum starting from red =0.

- At H=0, the color is a shade of red, at H=117, the color is a shade of green, at H=170, the color is a shade of blue.

Saturation - S

Saturation is what we perceive as the amount of purity of the color, or of the grey in a color. For example a high saturation value produces a very pure, intense color. Reducing the saturation value adds grey to the color.

Luminance - L

Luminance is perceived as the brightness of the color, or the amount of white contained in the color. As the value increases the color becomes lighter and tends towards white. As the luminance value decreases the color is darker and tends towards black.

Color Tolerances

The color filter accepts any color values that are within defined tolerances. Tolerance values can only be expressed in HSL values. The tolerance range for each H, S, and L tolerance is applied to the defined color.

A defined tolerance value is distributed equally above and below the color value to which it applies.

For example, if the **Color** luminance (L) value is 200 and the Tolerance luminance (L) value is 20, the filter will accept pixels within a range of luminance values equal to [190,200].

Results Panel

The Results panel shows the results of the Color Finder process, including a visual display of the sampled Image. Read only text boxes provide dimensional and positional informations on this Image.

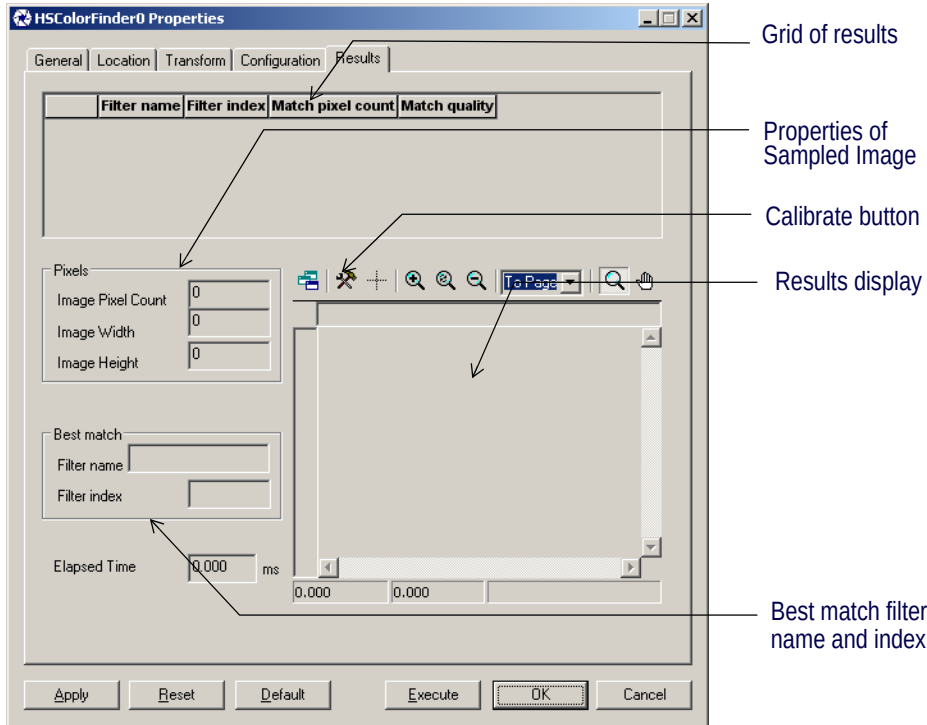


Figure 347 Color Finder — Results Panel

Results Display

The **Results Display** shows the sampled Image. Use the Calibrate button of the **Results Display** to toggle between calibrated and uncalibrated display modes.

Grid of Results

The **Grid of Results** presents the statistical results for the region of interest analyzed by the Color Finder Tool.

Filter (n) Name

Name of the filter. This result is output for each filter, starting at Filter 0.

Filter (n) Index

Index of the filter. This result is output for each filter, starting at Filter 0.

Filter (n) Match Pixel Count

Number of pixels that match the conditions set by the filter. This result is output for each filter, starting at Filter 0.

Filter (n) Match Quality

Filter Match Quality is a percentage value of pixels matched to the specified filter. This value is equal to the number of matched pixels (Filter (n) Match Pixel Count), divided by the total number of pixels in the region of interest (Image Pixel Count). This result is output for each filter, starting at Filter 0.

Description of Color Finder Tool Results

The Color Finder Tool outputs the following results:

Image Pixel Count

Image Pixel Count is the number of pixels in the tool region of interest. This is equal to Image Height x Image Width.

Image Width

X-axis length, in pixels, of the tool region of interest.

Image Height

Y-axis length, in pixels, of the tool region of interest.

Best Filter Name

Best Filter Name is the name of the filter for which the greatest number of pixels were found. This is the name of the filter as it appears in the filters list of the tool interface.

Best Filter Index

Best Filter Index is the index number of the filter for which the greatest number of pixels were found. This is the index number of the filter as it appears in the filters list of the tool interface.

Elapsed Time

The Elapsed Time is the total execution time of the Color Finder Tool.

Point Finder

The Point Finder finds and locates point features on objects and returns the coordinates of the found point. The tool can output the found point entity to the runtime database for use by other HexSight tools such as the Builder tool.

Using the Point Finder

Setup of the Point Finder is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Point Finder will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

- [General Panel](#)
- [Location Panel](#)
- [Transform Panel](#)
- [Configuration Panel](#)
- [Results Panel](#)

General Panel

Use this panel to name the instance of the Point Finder and enter the tool's inputs and output.

The inputs required by the tool depend on the selected [Positioning Mode](#).

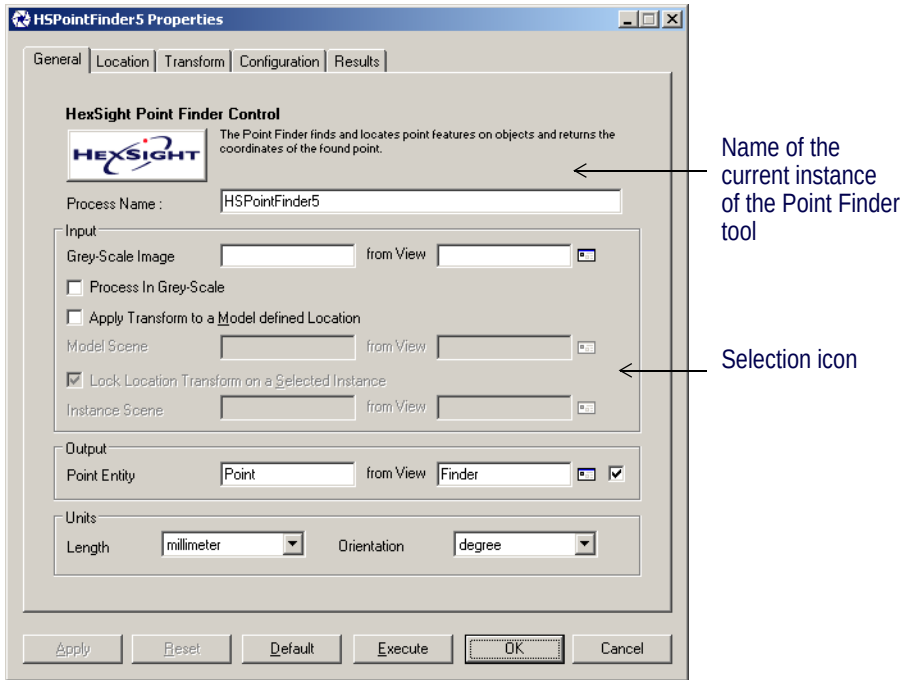


Figure 348 Point Finder — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based automatic. See the **Inspection Tools** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 31 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSPointFinder#, where # is replaced by its rank number in the list of instantiated processes.

Input

Image

The Image is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the Input Image text box and the name of the View containing this Image in the associated from View text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool.

Model Scene

The Model Scene input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the Model Scene text box and the name of the View containing this Scene in the associated from View text box

- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the Lock Location Transform on a Selected Instance check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The Instance Scene input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the Instance Scene text box and the name of the View containing this Scene in the associated from View text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Output

By default, the Point Finder tool generates a Point Entity that is output to the application's runtime database. This output can be used by other processes such as the Builder. If only the results of the point found by the tool are needed, the entity output can be disabled to improve the execution time.

Point Entity

The Point Entity generated by the tool is stored to the runtime database. It can be used by the Builder tool to find and generate other geometric entities.

- Type the name of the Entity in the **Point Entity** text box and the name of the View containing this Entity in the associated **from View** text box.

- Alternately, you can click the selection Icon to obtain a menu of compatible Views currently available in the runtime database.
- You can deactivate the output of the entity using the check box. The results of the found point will be available to other processes but no Point entity will be created in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

- When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.
- When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

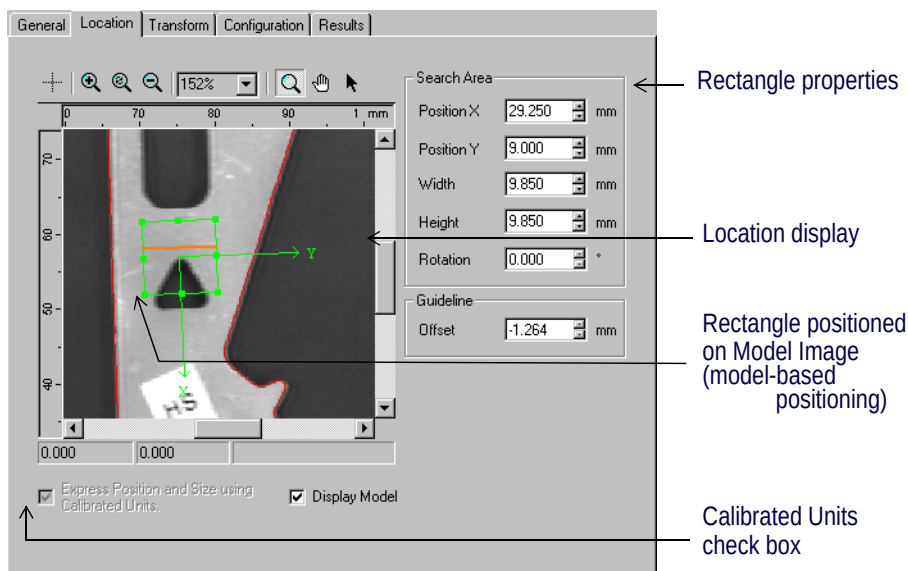


Figure 349 Point Finder — Location Pane

Express Position and Size using Calibrated Units

Use the check box labeled **Express Position and Size using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning.

- This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.
- With Model-based positioning, positions and dimensions are always expressed in calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Properties

The tool's region of interest is the **Rectangle** that is defined by: **Position X**, **Position Y**, **Width**, **Height**, and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position and Size using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Guideline

The **Guideline** marker can be displaced along the X-axis of the tool Rectangle. The Guideline acts as both a visual guide for positioning the tool and as a constraint for the tool's [Search Mode](#).

- **Offset** is the offset from the tool's X-axis.
- See [Figure 350](#) for an illustration of positive and negative **Offset** values.

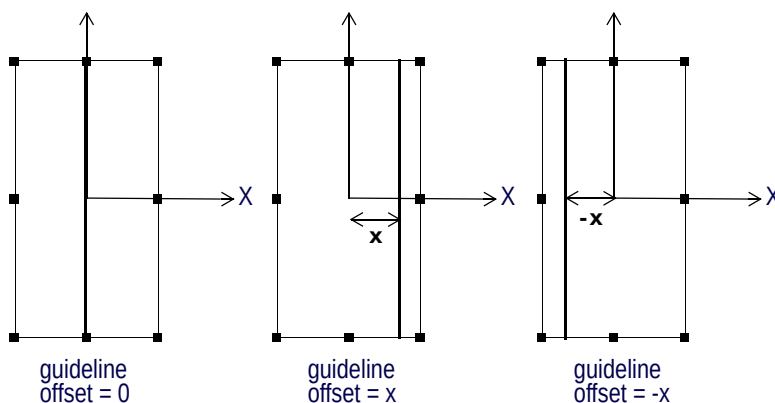


Figure 350 Point Finder — Guideline Offset

Positioning the Rectangle and Guideline

The Tool will search for the point that is closest to the Guideline as defined by the [Search Mode](#) constraint (in the Configuration panel). Typically the Line Finder is used to find a corner or peak on an object's contour.

- Type the Rectangle and Guideline parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Move the guideline by dragging - it will only move along the X axis, within the bounds of the Rectangle
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

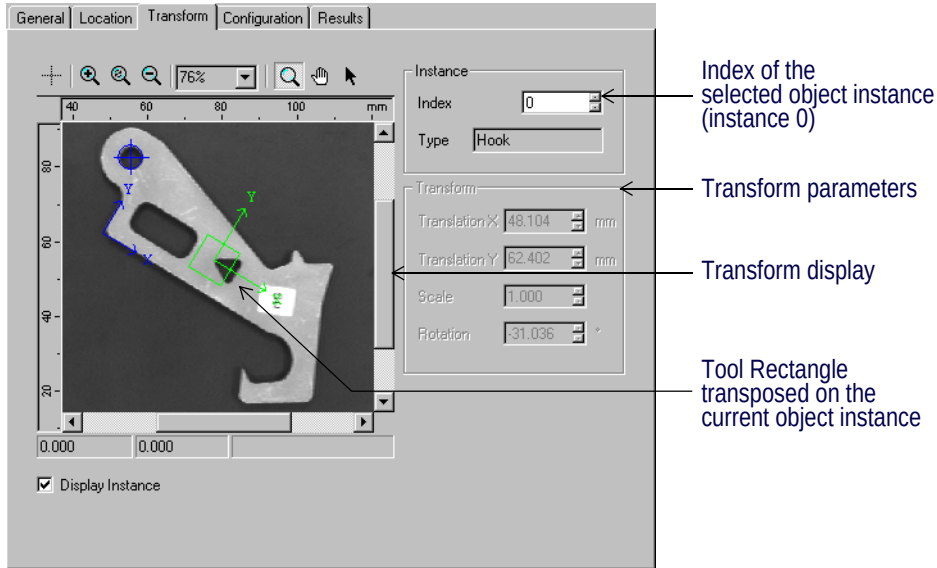


Figure 351 Point Finder — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation.

- When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.
- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

Configuration Panel

Use this panel to configure the search parameters used by the Point Finder process.

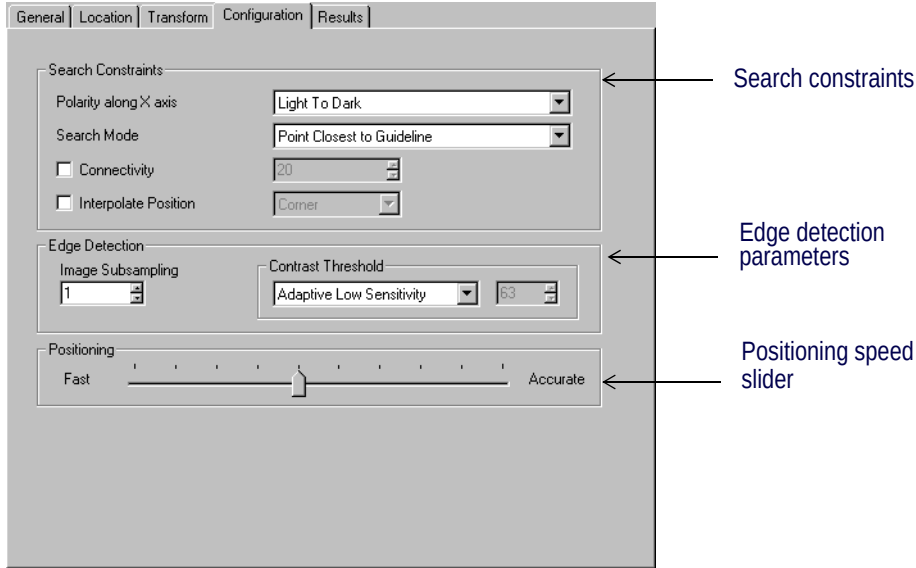


Figure 352 Point Finder — Configuration Panel

Search Constraints

Use Search Constraints to restrict the tool's search process. For example, you can set the tool to search for the point instance that is closest to the guideline.

Polarity along the X axis

Polarity identifies the change in greylevel values along the tool's X axis, in the positive direction.

- **Dark To Light:** The Point Finder searches only for point instances occurring at a dark to light transition in greylevel values.
- **Light To Dark:** The Point Finder searches only for point instances occurring at a light to dark transition in greylevel values.
- **Either:** The Point Finder searches only for point instances occurring either at a light to dark or dark to light transition in greylevel values. This mode will increase processing time.

- **Don't Care:** The Point Finder searches only for point instances occurring at any transition in greylevel values including reversals in contrast , for example on an unevenly colored background.

Search Mode

The **Search Mode** specifies the method used by the tool to generate and select a hypothesis.

- **Point Closest To Guideline:** Selects the point hypothesis closest to the Guideline.
- **Point With Maximum Negative X Offset:** Selects the point hypothesis closest to the Rectangle bound that is at maximum negative X offset.
- **Point With Maximum Positive X Offset:** Selects the point hypothesis closest to the Rectangle bound that is at maximum positive X offset.

Connectivity

By default, **Connectivity** is disabled. When enabled, you can set the minimum number of connected edges that are required to generate a point hypothesis.

Interpolate Position

By default, **Interpolate Position** is disabled. When Enabled, you can select the mode used by the tool to compute a point hypothesis.

- **Corner:** The tool will compute a hypothesis that fits a corner point to interpolated lines from connected edges.
- **Intersection:** The tool will compute a hypothesis that is an intersection between the search axis and connected edges of an interpolated line.

Edge Detection

Edge detection parameters determine how the tool detects edges in the input image. These edges are used to generate hypotheses used by the tool to find and generate a valid entity.

Image Subsampling

The tool applies the **Image Subsampling** value when detecting edges that are used to generate hypotheses.

- High values provide a coarser search with a shorter execution time.
- Conversely lower values can provide a more refined search with slower execution time.
- Using a higher subsampling value may help improve accuracy in blurry images.

Contrast Threshold

Contrast Threshold sets the sensitivity to the contrast needed for an edge to be detected in the input image and used to compute a point entity.

- The default **Normal Adaptive Sensitivity** is recommended for most applications.
- **Adaptive** thresholds provide flexibility to variations in image lighting conditions and contrast.
- **Adaptive High Sensitivity** accepts lower contrasts and therefore results in a greater number of edges. **Adaptive Low Sensitivity** retains high-contrast edges and removes lower contrast edges, such as those caused by noise and shadows.
- **Fixed Value** requires that you manually set the **Contrast Threshold** value, expressed in terms of a step in greylevel values that ranges from 0 to 255. Higher values reduce sensitivity to contrast, resulting in fewer, high-contrast edges. Conversely lower values increase sensitivity and add a greater amount of edges.

Positioning

The slider provided for adjusting the positioning accuracy ranges from **Fast** to **Accurate**.

- The default center position of the slider is the optimized and recommended setting for typical applications.
- In applications where accuracy is not critical, setting the slider towards **Fast** can provide an improvement in speed.
- Moving the slider towards **Accurate** increases the positional accuracy of found entity.

Results Panel

The Results panel shows the results of the Point Finder process, including a visual display of the found entity.

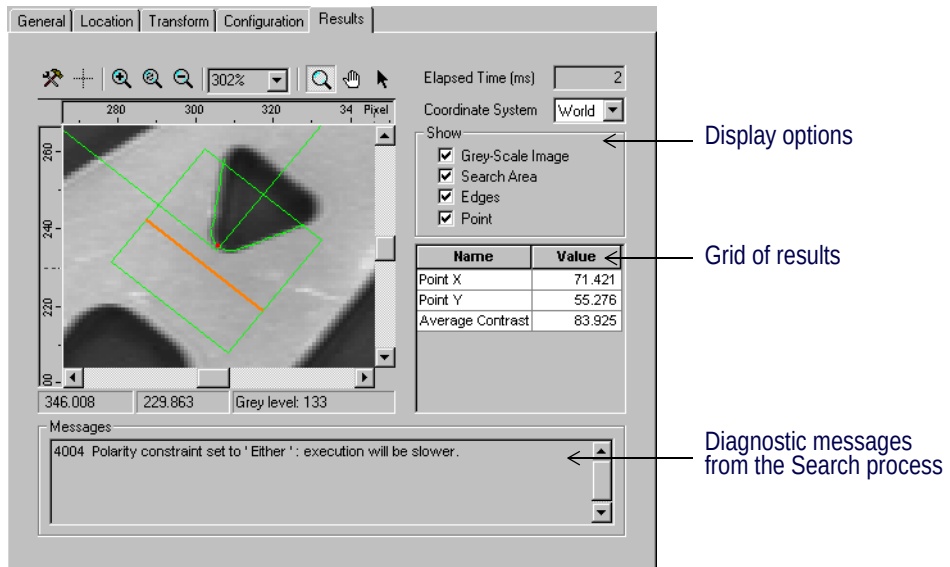


Figure 353 Point Finder — Results Panel

Elapsed Time

Elapsed Time shows the total execution time of the Point Finder.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image**, **World**, **Object**, **Tool**.

The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Results Display

The **Results Display** shows the found entity. Use the Calibrate button of the **Results Display** to toggle between calibrated and uncalibrated display modes.

Show check boxes enable various display options:

- Enable **Image** to display the current input Image.
- Enable **Search Area** to display the tool's Rectangle and guideline (orange).
- Enable **Edges** to display edges. Candidate edges used for hypothesis generation are displayed in a different color (green) than other edges (blue).
- Enable **Point** to display the point entity (red) found by the tool.

Grid of Results

Results for the found point instance are displayed in the grid. They can be expressed in either the **Image**, **World**, **Object** or **Tool** coordinate system, according to the setting of the **Coordinate System** parameter.

- **Point X, Point Y**: The coordinates of the found point entity.
- **Average Contrast**: Average greylevel contrast between light and dark pixels on either side of the found point.

Messages

The tool outputs diagnostic messages for the search process. [Table 32](#) presents the possible diagnostic messages, with their code numbers.

Table 32 Diagnostic Codes and Messages

Codes	Message
4001	Execution interrupted : no Input Image was provided
4002	Execution interrupted : Selected Instance not found
4003	Execution interrupted : Search Area not included in Input Image
4004	Polarity constraint set to ' Either ' : execution will be slower
4005	Search Mode set to ' Best ... ' : hypothesis generation will be slower
4006	Hypothesis generation interrupted : Search Area Height is zero
4007	Hypothesis generation interrupted : not enough edges matched
4008	Hypothesis generation interrupted : no edges were detected
4009	Hypothesis generation interrupted : no edges fit constraints
4010	Refinement interrupted : Search Area Height is zero
4011	Hypothesis : Deviation Angle (%1) failed
4012	Refinement interrupted : not enough edges matched to perform fit
4061	Execution interrupted : no edge respecting Connectivity constraint
4063	Execution interrupted : no edge matching constraints

Line Finder

The Line Finder finds and locates linear features on objects and returns the line angle and point coordinates. The tool can output the found line entity to the runtime database for use by other HexSight tools such as the Builder tool.

Line Finder Chapter Outline

[Using the Line Finder](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Using the Line Finder

Setup of the Line Finder is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Line Finder will follow the left to right order of these panels.

General Panel

Use this panel to name the instance of the Line Finder and enter the tool's inputs and output.

The inputs required by the tool depend on the selected [Positioning Mode](#).

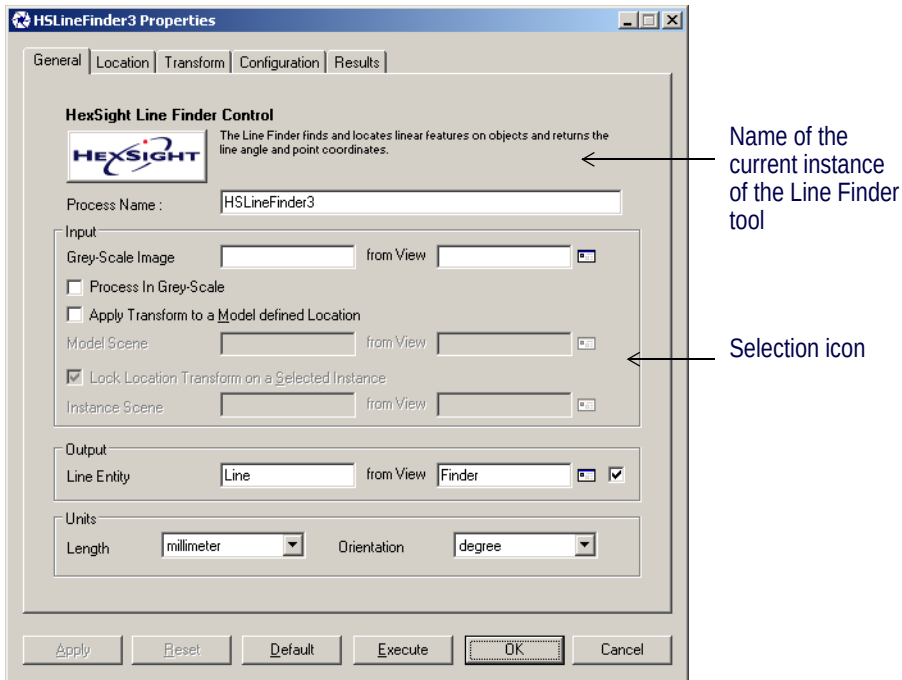


Figure 354 Line Finder — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based automatic. See the ***Inspection Tools*** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position and Size using Calibrated Units**

[Table 33](#) provides an overview of the required inputs and settings for the four positioning modes.

Table 33 Positioning Modes: Required Inputs and Settings

Table 34

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 34

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSLineFinder#, where # is replaced by its rank number in the list of instantiated processes.

Input

Image

The **Image** is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the Acquisition Device, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool.

Model Scene

The **Model Scene** input is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator tool that is configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The **Instance Scene** input is the Scene in the runtime database that provides the transform parameters for object instances found by the HexSight Locator. This Scene is provided by a Locator configured to output an Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database

Output

By default, the Line Finder tool generates a Line Entity that is output to the application's runtime database. This output can be used by other processes such as the Builder. If only the results of the line found by the tool are needed, the entity output can be disabled to improve the execution time.

Line Entity

The **Line Entity** generated by the tool is stored to the runtime database. It can be used by the Builder tool to find and generate other geometric entities.

- Type the name of the Entity in the **Line Entity** text box and the name of the View containing this Entity in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Views currently available in the runtime database.
- You can deactivate the output of the entity using the check box. The results of the found line will be available to other processes but no Line entity will be created in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

- When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.
- When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

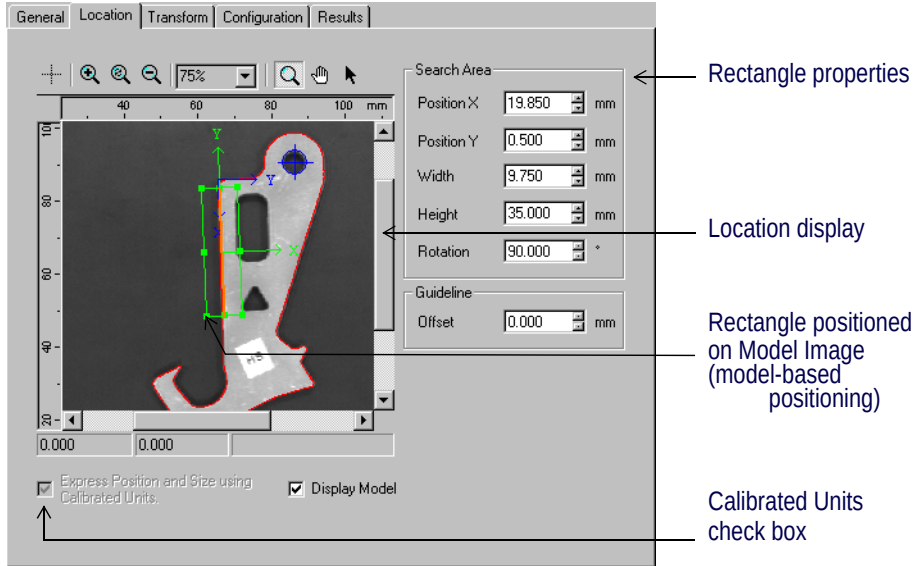


Figure 355 Line Finder — Location Panel

Express Position and Size using Calibrated Units

Use the check box labeled **Express Position and Size using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning.

- This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.
- With Model-based positioning, positions and dimensions are always expressed in calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Properties

The tool's region of interest is the **Rectangle** that is defined by: **Position X**, **Position Y**, **Width**, **Height**, and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position and Size using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Guideline

The **Guideline** marker can be displaced along the X-axis of the tool Rectangle. The Guideline acts as both a visual guide for positioning the tool and as a constraint for the tool's [Search Mode](#).

- **Offset** is the offset from the tool's X-axis.
- See [Figure 356](#) for an illustration of positive and negative **Offset** values.

Positioning the Rectangle and Guideline

- Type the Rectangle and Guideline parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Move the guideline by dragging - it will only move along the X axis, within the bounds of the Rectangle

- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

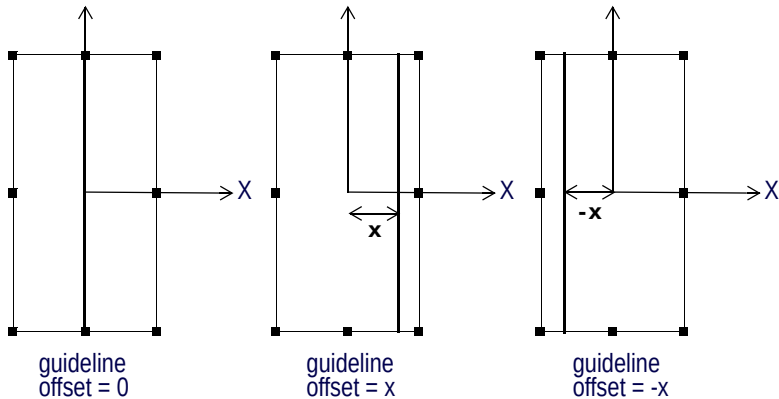


Figure 356 Line Finder — Guideline Offset

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

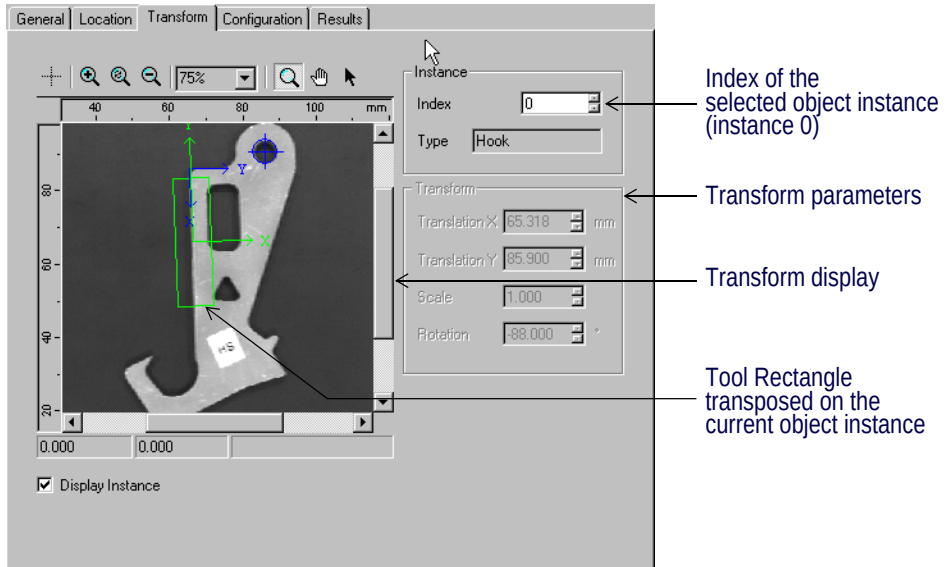


Figure 357 Line Finder — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation.

- When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.
- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

Configuration Panel

Use this panel to configure the search parameters used by the Line Finder process.

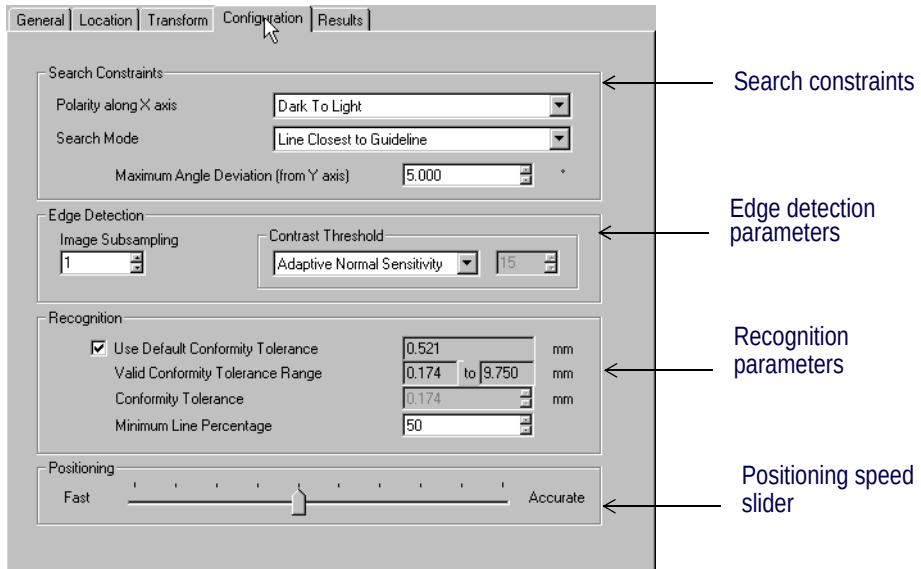


Figure 358 Line Finder — Configuration Panel

Search Constraints

Use Search Constraints to restrict the tool's search process. For example, you can set the tool to search for the line instance that is closest to the guideline.

Polarity along the X axis

Polarity identifies the change in greylevel values along the tool's X axis, in the positive direction.

- **Dark To Light:** The Line Finder searches only for line instances occurring at a dark to light transition in greylevel values.
- **Light To Dark:** The Line Finder searches only for line instances occurring at a light to dark transition in greylevel values.
- **Either:** The Line Finder searches only for line instances occurring either at a light to dark or dark to light transition in greylevel values. This mode will increase processing time.

- **Don't Care:** The Line Finder searches only for line instances occurring at any transition in greylevel values including reversals in contrast along the line, for example on an unevenly colored background.

Search Mode

The **Search Mode** specifies the method used by the tool to generate and select a hypothesis.

- **Best Line:** Selects the best line according to hypotheses strengths. This mode will increase processing time.
- **Line Closest To Guideline:** Selects the line hypothesis closest to the Guideline.
- **Line With Maximum Negative X Offset:** Selects the line hypothesis closest to the Rectangle bound that is at maximum negative X offset.
- **Line With Maximum Positive X Offset:** Selects the line hypothesis closest to the Rectangle bound that is at maximum positive X offset.

Maximum Angle Deviation

Maximum Angle Deviation is the maximum allowable deviation in angle between the line hypothesis and the line found by the tool.

- The deviation is calculated along the Y axis between the line hypothesis and the line found by the tool.
- By default the Line Finder accepts a 20 degree deviation then uses the defined **Maximum Angle Deviation** value to test the hypothesis and refine the pose of the found line.
- The angle is expressed either in degrees or in radians, depending on the units selected with the **Orientation Units** combo box.

Edge Detection

Edge detection parameters determine how the tool detects edges in the input image. These edges are used to generate hypotheses used by the tool to find and generate a valid entity.

Image Subsampling

The tool applies the **Image Subsampling** value when detecting edges that are used to generate hypotheses.

- High values provide a coarser search with a shorter execution time.
- Conversely lower values can provide a more refined search with slower execution time.
- Using a higher subsampling value may help improve accuracy in blurry images.

Contrast Threshold

Contrast Threshold sets the sensitivity to the contrast needed for an edge to be detected in the input image and used to compute a line entity.

- The default **Normal Adaptive Sensitivity** is recommended for most applications.
- **Adaptive** thresholds provide flexibility to variations in image lighting conditions and contrast.
- **Adaptive High Sensitivity** accepts lower contrasts and therefore results in a greater number of edges. **Adaptive Low Sensitivity** retains high-contrast edges and removes lower contrast edges, such as those caused by noise and shadows.
- **Fixed Value** requires that you manually set the **Contrast Threshold** value, expressed in terms of a step in greylevel values that ranges from 0 to 255. Higher values reduce sensitivity to contrast, resulting in fewer, high-contrast edges. Conversely lower values increase sensitivity and add a greater amount of edges.

Recognition

Use recognition parameters to configure the specific search process requirements of the application.

Use Default Conformity Tolerance

Disabling **Use Default Conformity Tolerance** allows you to manually modify the **Conformity Tolerance** value.

Valid Conformity Tolerance Range

Valid Conformity Tolerance Range defines the upper and lower limits for the **Conformity Tolerance**.

- This range is based on the calibration and resolution of the input Image and cannot be edited.
- You can only set Conformity Tolerance to a value within this range.

Conformity Tolerance

Conformity Tolerance corresponds to the maximum distance in calibrated units by which a matched edge can deviate from either side of its expected position on the line.

- To manually set **Conformity Tolerance** you must disable the **Use Default Conformity Tolerance** check box.
- If you set a value lower than the minimum **Valid Conformity Tolerance Range**, this value will be automatically reset to the minimum valid value.
- If you set a value higher than the maximum **Valid Conformity Tolerance Range**, this value will be automatically reset to the maximum valid value.

Minimum Line Percentage

Minimum Line Percentage sets the percentage of line length that needs to be matched for a line hypothesis to be considered as valid.

Positioning

The slider provided for adjusting the positioning accuracy ranges from **Fast** to **Accurate**.

- The default center position of the slider is the optimized and recommended setting for typical applications.
- In applications where accuracy is not critical, setting the slider towards **Fast** can provide an improvement in speed.
- Moving the slider towards **Accurate** increases the positional accuracy of found entity.

Results Panel

The Results panel shows the results of the Line Finder process, including a visual display of the found entity.

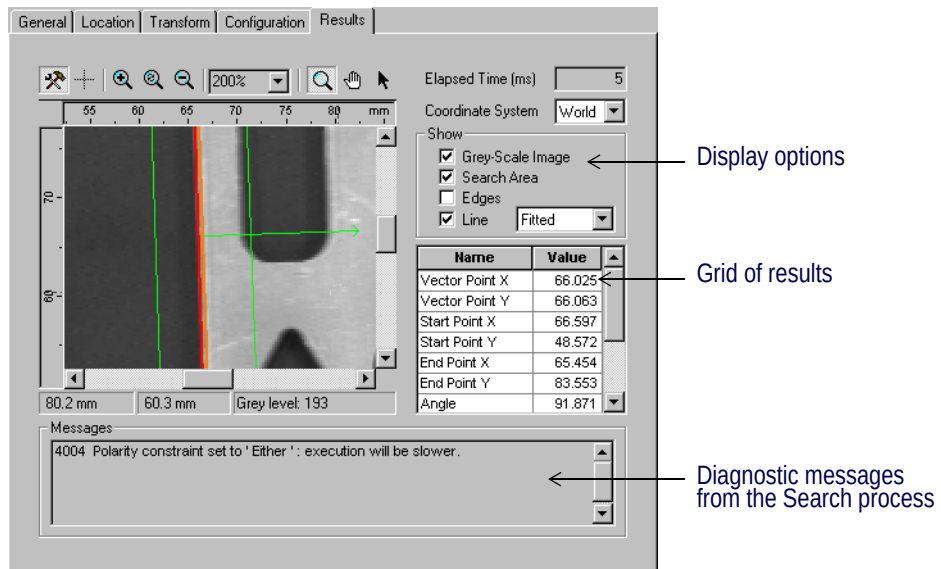


Figure 359 Line Finder — Results Pane

Elapsed Time

Elapsed Time shows the total execution time of the Line Finder.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image**, **World**, **Object**, **Tool**.

The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Results Display

The **Results Display** shows the found entity. Use the Calibrate button of the **Results Display** to toggle between calibrated and uncalibrated display modes.

Show check boxes enable various display options:

- Enable **Image** to display the current input Image.
- Enable **Search Area** to display the tool's Rectangle and guideline (orange).
- Enable **Edges** to display edges. Candidate edges used for hypothesis generation are displayed in a different color (green) than other edges (blue).
- Enable **Line** to display the line entity (red) found by the tool. The line can be displayed in one of two modes:

Fitted: The entire fitted line is represented.

Matched: Only matched sections of the line are represented.

Grid of Results

Results for the fitted line instance are displayed in the grid. They can be expressed in either the **Image**, **World**, **Object** or **Tool** coordinate system, according to the setting of the **Coordinate System** parameter. Depending on the selected coordinate system, some values will not be output because they are non-valid or non-significant in the selected frame of reference. See [Figure 360](#) for an illustration of Line properties.

Vector Point X, Vector point Y

The coordinates of the point of intersection between the line and the X axis of the Rectangle. See [Figure 360](#). Exceptionally, when the line exits the Rectangle without covering the entire Rectangle height, the returned Vector Point may be located outside the Rectangle boundary.

Start Point X, Start Point Y

The coordinates of the point at the start of the line segment.

End Point X, End Point Y

The coordinates of the point at the end of the line segment.

Angle

Angle of the found line. A line is defined as the line passing through the point coordinates Vector Point X and Vector Point Y at the given Angle.

Average Contrast

Average greylevel contrast between light and dark pixels on either side of the found line.

Fit Quality

Normalized average error between the calculated line and the actual edges matched to the found line. Fit quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that the average error is 0. Conversely, a value of 0 means that the average matched error is equal to conformity tolerance.

Match Quality

Percentage of edges actually matched to the found line. Match Quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that edges were matched for every point along the found line. Similarly, a value of 0.2 means edges were matched to 20% of the points along the found line.

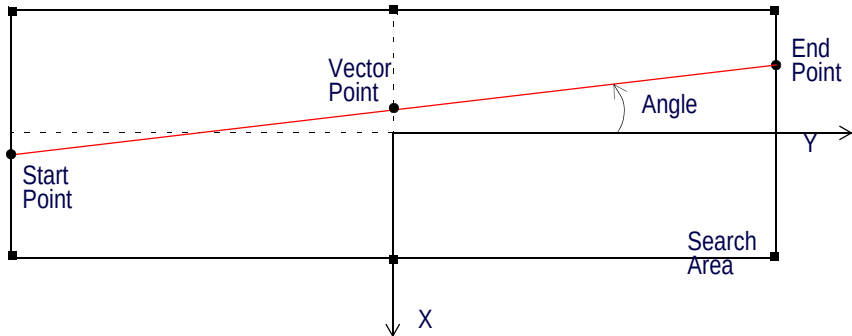


Figure 360 Line Entity Properties

Messages

The tool outputs diagnostic messages for the search process. [Table 35](#) presents the possible diagnostic messages, with their code numbers.

Table 35 Diagnostic Messages Output by the Line Finder

Codes	Message
4001	Execution interrupted : no Input Image was provided
4002	Execution interrupted : Selected Instance not found
4003	Execution interrupted : Search Area not included in Input Image
4004	Polarity constraint set to ' Either ' : execution will be slower
4005	Search Mode set to ' Best ... ' : hypothesis generation will be slower
4006	Hypothesis generation interrupted : Search Area Height is zero
4007	Hypothesis generation interrupted : not enough edges matched
4008	Hypothesis generation interrupted : no edges were detected
4009	Hypothesis generation interrupted : no edges fit constraints

Table 35 Diagnostic Messages Output by the Line Finder

Codes	Message
4010	Refinement interrupted : Search Area Height is zero
4011	Hypothesis : Deviation Angle failed
4012	Refinement interrupted : not enough edges matched to perform fit
4040	Hypothesis : Deviation Angle or Line Percentage failed
4044	Found Line : Deviation Angle or Line Percentage failed

Arc Finder

The Arc Finder finds and locates circular features on objects and returns the coordinates of the center of the arc, the start and end angles, and the radius. The tool can output the found arc entity to the runtime database for use by other HexSight tools such as the Builder tool.

Arc Finder Chapter Outline

[Using the Arc Finder](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Using the Arc Finder

Setup of the Arc Finder is carried out on five tab-selected panels: General, Location, Transform, Configuration, and Results. Steps for using and setting up the Arc Finder will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Arc Finder and enter the tool's inputs and output.

The inputs required by the tool depend on the selected [Positioning Mode](#).

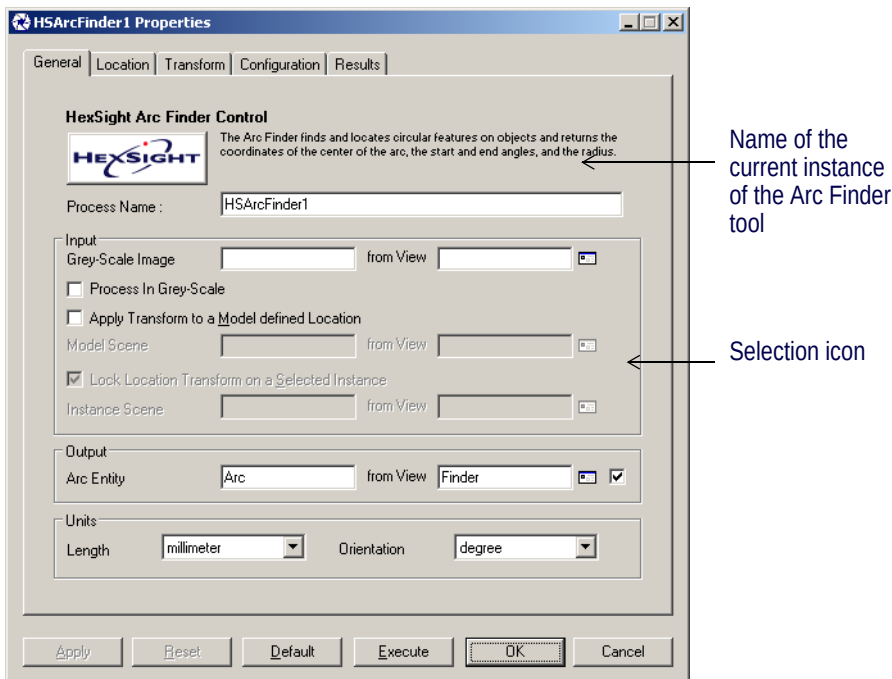


Figure 361 Arc Finder — General Panel

Positioning Mode

The positioning mode defines how the tool's region of interest is positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position and Size using Calibrated Units**

[Table 36](#) provides an overview of the required inputs and settings for the four positioning modes.

Table 36 Positioning Modes: Required Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 36 Positioning Modes: Required Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSArcFinder#, where # is replaced by its rank number in the list of instantiated processes.

Input

Image

The **Image** is the Image from the runtime database on which the tool is applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image output by a HexSight tool can be used.

- Type the name of the Image in the **Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Only grey scale information will be used for edge extraction. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

- If you enable Model-based positioning, you must specify the input Model Scene that will be used to position the tool on the Model.

Model Scene

The input **Model Scene** is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

- If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Instance Scene

The input **Instance Scene** is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Locator. This Scene is provided by a HexSight Locator configured to output its Instance Scene in the runtime database. See the **Locator** chapter of the User Guide for more details.

- Type the name of the Scene in the **Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Output

By default, the Arc Finder tool generates an Arc Entity that is output to the application's runtime database. This output can be used by other processes such as the Builder. If only the results of the arc found by the tool are needed, the entity output can be disabled to improve the execution time.

Arc Entity

The **Arc Entity** generated by the tool is stored to the runtime database. It can be used by the Builder tool to find and generate other geometric entities.

- Type the name of the Entity in the **Arc Entity** text box and the name of the View containing this Entity in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Views currently available in the runtime database.
- You can deactivate the output of the entity using the check box. The results of the found arc will be available to other processes but no Arc entity will be created in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Arc, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Arc that defines the tool region of interest.

- When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.
- When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

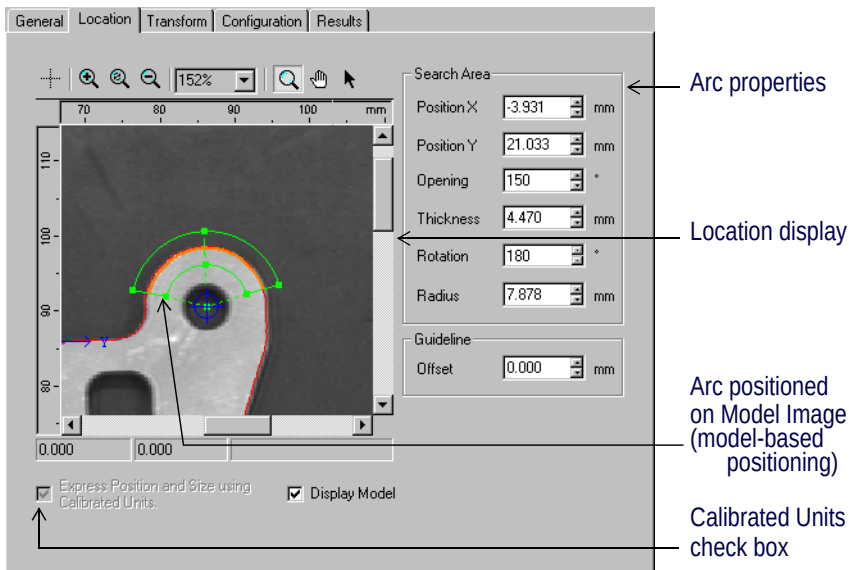


Figure 362 Arc Finder — Location Panel

Express Position and Size using Calibrated Units

Use the check box labeled **Express Position and Size using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning.

- This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.
- With Model-based positioning, positions and dimensions are always expressed in calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

- This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Arc Properties

The tool's region of interest is the **Arc** that is defined by: **Position X**, **Position Y**, **Width**, **Thickness**, **Radius**, and **Rotation**. See [Figure 363](#).

- **Position X**, **Position Y**, **Thickness**, and **Radius** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Opening** and **Rotation** are always expressed in the selected orientation unit.
- With Image-based positioning, the Arc is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position and Size using Calibrated Units** check box.
- With Model-based positioning, the Arc is positioned with respect to the Object coordinate system.

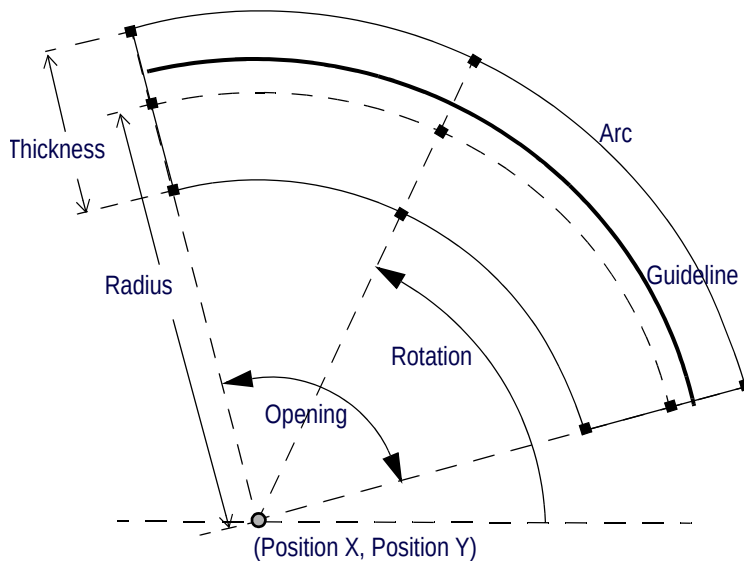


Figure 363 Arc Finder - Arc Properties

Guideline

The **Guideline** marker can be displaced radially within the Arc. The Guideline acts as both a visual guide for positioning the tool and as a constraint for the tool's [Search Mode](#).

- **Offset** is the radial offset of the Guideline marker from the center of the tool Arc.
- Values are positive moving outwards from the tool center and negative moving inwards.

Positioning the Arc and Guideline

Type Arc and Guideline parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.

To manually position and set the Arc search area and Guideline within the display, use the following tips and refer to [Figure 364](#).

- Move the Arc by dragging the tool's center: (PositionX, PositionY).
- Move the guideline by dragging - it will only move radially within the bounds of the Arc
- Change thickness by dragging center handles.
- Rotate the Arc by dragging outer "corners" of the Arc.
- Adjust radius by dragging the crosshair at "center" of the Arc.

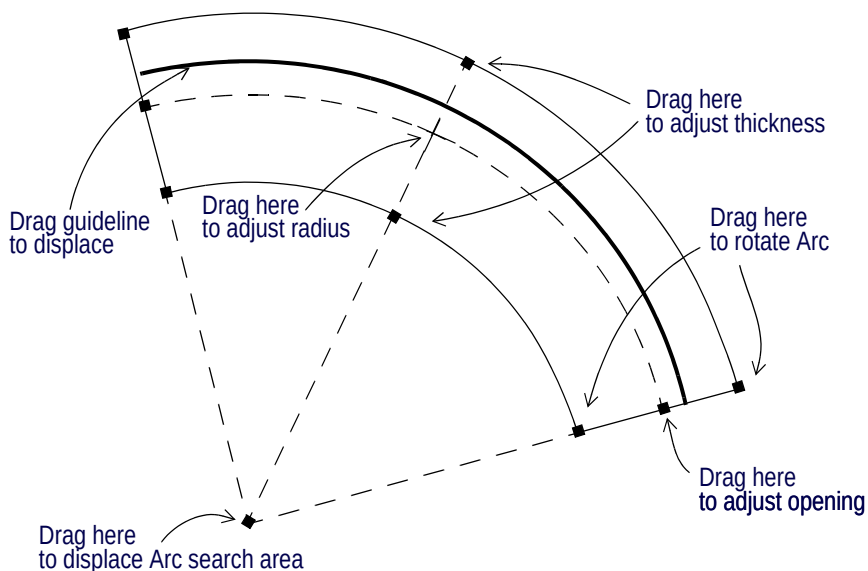


Figure 364 Arc Finder - Manual Positioning of Arc Search Area and Guideline

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

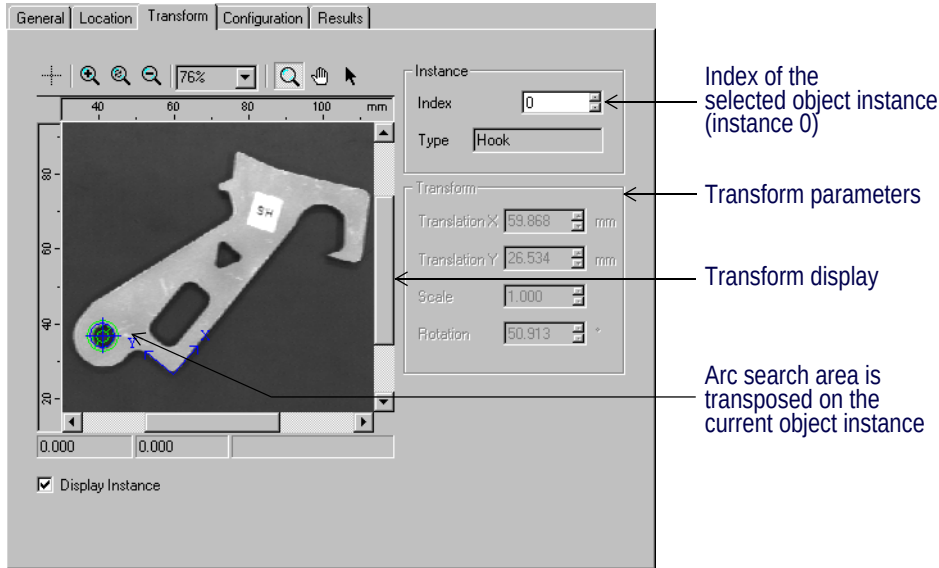


Figure 365 Arc Finder — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Arc region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the instances that are available in the current Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation.

- When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.
- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

Configuration Panel

Use this panel to configure the search parameters used by the Arc Finder process.

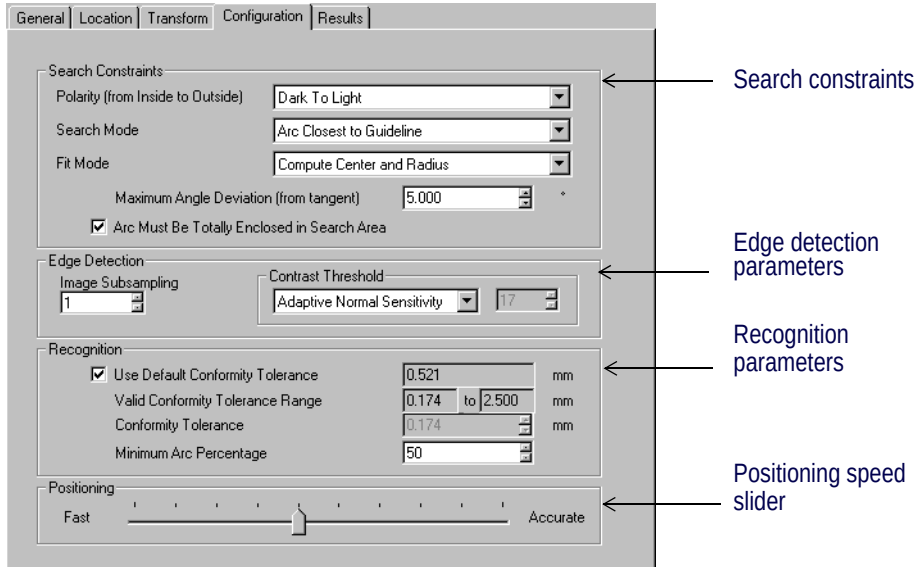


Figure 366 Arc Finder — Configuration Panel

Search Constraints

Use Search Constraints to restrict the tool's search process. For example, you can set the tool to search for the arc instance that is closest to the guideline.

Polarity from Inside to Outside

Polarity identifies the change in greylevel values outwards from the arc's center.

- **Dark To Light:** The Arc Finder searches only for arc instances occurring at a dark to light transition in greylevel values.
- **Light To Dark:** The Arc Finder searches only for arc instances occurring at a light to dark transition in greylevel values.
- **Either:** The Arc Finder searches only for arc instances occurring either at a light to dark or dark to light transition in greylevel values. This mode will increase processing time.

- **Don't Care:** The Arc Finder searches only for arc instances occurring at any transition in greylevel values including reversals in contrast along the arc, for example on an unevenly colored background.

Search Mode

The **Search Mode** specifies the method used by the tool to generate and select a hypothesis.

- **Best Arc:** Selects the best arc according to hypotheses strengths. This mode will increase processing time.
- **Arc Closest To Guideline:** Selects the arc hypothesis closest to the Guideline.
- **Arc Closest To Inside:** Selects the arc hypothesis closest to the inside of the tool Arc (closest to the tool center).
- **Arc Closest To Outside:** Selects the arc hypothesis closest to the outside of the tool Arc (farthest to the tool center).

Fit Mode

The **Fit Mode** provides a choice of three methods for fitting hypotheses to a valid arc entity.

- **Compute Center and Radius:** The Arc Finder calculates and returns both the arc center and the arc radius. This is the default mode, which will typically provide the most accurate results.
- **Compute Center:** The Arc Finder calculates the arc center. The returned **Radius** value is the tool's radius (i.e. Arc Radius).
- **Compute Radius:** The Arc Finder calculates the arc radius. The arc returned **Center Point** values are the tool's center (i.e. Arc PositionX and PositionY).

Maximum Angle Deviation

Maximum Angle Deviation is the maximum allowable deviation in angle between the arc hypothesis and the arc found by the tool.

- The deviation is calculated between the tangent angle of the arc at points where the edge is matched to the arc.
- The angle is expressed either in degrees or in radians, depending on the units selected with the **Orientation Units** combo box.

Arc Must Be Totally Enclosed in Search Area

By default **Arc Must Be Totally Enclosed in Arc** is enabled, which means that the tool will find an arc *only* if both its **start** and **end** points are located on the radial bounding sides of the Arc search area.

When disabled, the tool can find an arc that enters and/or exits the Arc at the inner or outer annular bounds of the Arc search area.

Edge Detection

Edge detection parameters determine how the tool detects edges in the input image. These edges are used to generate hypotheses used by the tool to find and generate a valid entity.

Image Subsampling

The tool applies the **Image Subsampling** value when detecting edges that are used to generate hypotheses.

- High values provide a coarser search with a shorter execution time.
- Conversely lower values can provide a more refined search with slower execution time.
- Using a higher subsampling value may help improve accuracy in blurry images.

Contrast Threshold

Contrast Threshold sets the sensitivity to the contrast needed for an edge to be detected in the input image and used to compute an arc entity.

- The default **Normal Adaptive Sensitivity** is recommended for most applications.

- **Adaptive** thresholds provide flexibility to variations in image lighting conditions and contrast.
- **Adaptive High Sensitivity** accepts lower contrasts and therefore results in a greater number of edges. **Adaptive Low Sensitivity** retains high-contrast edges and removes lower contrast edges, such as those caused by noise and shadows.
- **Fixed Value** requires that you manually set the **Contrast Threshold** value, expressed in terms of a step in greylevel values that ranges from 0 to 255. Higher values reduce sensitivity to contrast, resulting in fewer, high-contrast edges. Conversely lower values increase sensitivity and add a greater amount of edges.

Recognition

Use recognition parameters to configure the specific search process requirements of the application.

Use Default Conformity Tolerance

Disabling **Use Default Conformity Tolerance** allows you to manually modify the **Conformity Tolerance** value.

Valid Conformity Tolerance Range

Valid Conformity Tolerance Range defines the upper and lower limits for the **Conformity Tolerance**.

- This range is based on the calibration and resolution of the input Image and cannot be edited.
- You can only set Conformity Tolerance to a value within this range.

Conformity Tolerance

Conformity Tolerance corresponds to the maximum distance in calibrated units by which a matched edge can deviate from either side of its expected position on the arc.

- To manually set **Conformity Tolerance** you must disable the **Use Default Conformity Tolerance** check box.
- If you set a value lower than the minimum **Valid Conformity Tolerance Range**, this value will be automatically reset to the minimum valid value.
- If you set a value higher than the maximum **Valid Conformity Tolerance Range**, this value will be automatically reset to the maximum valid value.

Minimum Arc Percentage

Minimum Arc Percentage sets the percentage of perimeter length that needs to be matched for an hypothesis to be considered as valid.

Positioning

The slider provided for adjusting the positioning accuracy ranges from **Fast** to **Accurate**.

- The default center position of the slider is the optimized and recommended setting for typical applications.
- In applications where accuracy is not critical, setting the slider towards **Fast** can provide an improvement in speed.
- Moving the slider towards **Accurate** increases the positional accuracy of found entity.

Results Panel

The Results panel shows the results of the Arc Finder process, including a visual display of the found entity.

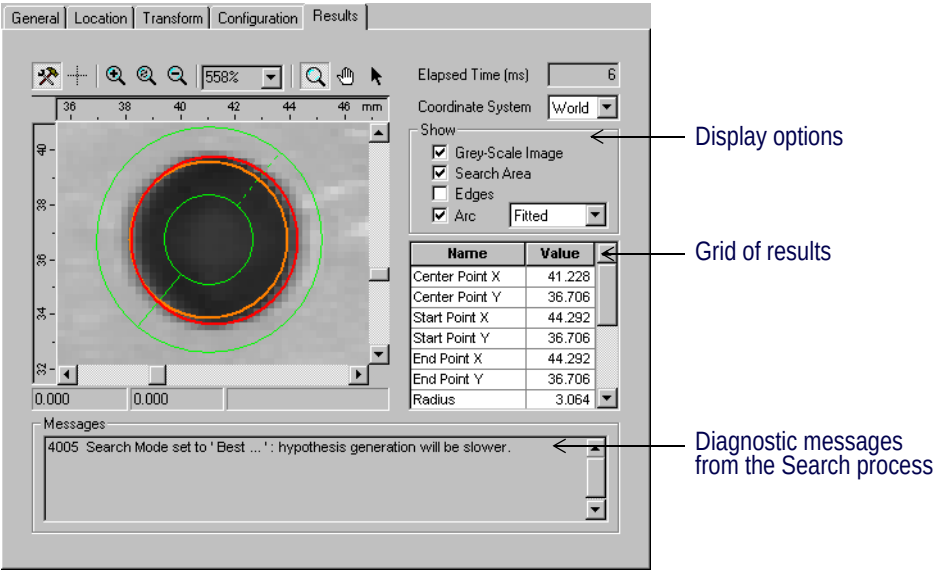


Figure 367 Arc Finder — Results Panel

Elapsed Time

Elapsed Time shows the total execution time of the Arc Finder.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image**, **World**, **Object**, **Tool**.

The output is influenced by the tool positioning mode, Model-based or Image-based and whether or not the system is calibrated.

Results Display

The **Results Display** shows the found entity. Use the Calibrate button of the **Results Display** to toggle between calibrated and uncalibrated display modes.

Show check boxes enable various display options:

- Enable **Image** to display the current input Image.
- Enable **Search Area** to display the tool's search area and guideline (orange).
- Enable **Edges** to display edges. Candidate edges used for hypothesis generation are displayed in a different color (green) than other edges (blue).
- Enable **Arc** to display the arc entity (red) found by the tool. The arc can be displayed in one of two modes:

Fitted: The entire fitted arc is represented.

Matched: Only matched sections of the arc are represented.

Grid of Results

Results for the fitted arc instance are displayed in the grid. They can be expressed in either the **Image**, **World**, **Object** or **Tool** coordinate system, according to the setting of the **Coordinate System** parameter. Depending on the selected coordinate system, some values will not be output because they are non-valid or non-significant in the selected frame of reference.

- **Center Point X, Center Point Y:** The coordinates of the arc center point.
- **Start Point X, Start Point Y:** The coordinates of the arc start point. On an x-to-y-axis trajectory, the arc start point, is the first point encountered.
- **End Point X, End Point Y:** The coordinates of the arc end point. On an x-to-y-axis trajectory, the arc end point is the last point encountered.
- **Radius:** Radius of the found arc.
- **Start Angle, End Angle:** Angle of the arc radius at the Start Point and End point of the found arc.
- **Average Contrast:** Average greylevel contrast between light and dark pixels on either side of the found arc.

- **Fit Quality:** Normalized average error between the calculated arc and the actual edges matched to the found arc. Fit quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that the average error is 0. Conversely, a value of 0 means that the average matched error is equal to conformity tolerance.
- **Match Quality:** Percentage of edges actually matched to the found arc. Match Quality ranges from 0 to 1, with 1 being the best quality. A value of 1 means that edges were matched for every point along the found arc. Similarly, a value of 0.2 means edges were matched to 20% of the points along the found arc.

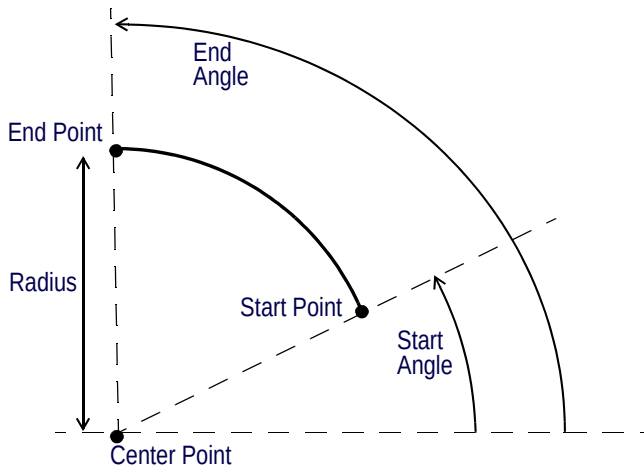


Figure 368 Arc Entity Properties

Messages

The tool outputs diagnostic messages for the search process. [Table 37](#) presents the possible diagnostic messages, with their code numbers.

Table 37 Arc Finder Search Result Messages

Code	Message
4001	Execution interrupted : no Input Image was provided
4002	Execution interrupted : Selected Instance not found
4003	Execution interrupted : Search Area not included in Input Image
4004	Polarity constraint set to ' Either ' : execution will be slower
4005	Search Mode set to ' Best ... ' : hypothesis generation will be slower
4006	Hypothesis generation interrupted : Search Area Height is zero
4007	Hypothesis generation interrupted : not enough edges matched
4008	Hypothesis generation interrupted : no edges were detected
4009	Hypothesis generation interrupted : no edges fit constraints
4010	Refinement interrupted : Search Area Height is zero
4011	Hypothesis : Deviation Angle failed
4012	Refinement interrupted : not enough edges matched to perform fit
4020	Hypothesis : Arc Percentage failed
4024	Found Arc : Arc Percentage failed
4025	Hypothesis generation interrupted : Arc not totally enclosed in Search Area
4026	Refinement interrupted : Arc not totally enclosed in Search Area

Builder Tool

The Builder creates geometric entities such as points, lines or arcs from user defined parameters or by combining other geometric entities. For example, lines or arcs can be created from multiple points while points can be created by intersecting lines and/or arcs.

Builder Tool Chapter Outline

[Using the Builder Tool](#)

[General Panel](#)

[Parameters Panel](#)

[Results Panel](#)

Using the Builder Tool

Setup of the Builder Tool is carried out on three tab-selected panels: General, Parameters, and Results. Steps for using and setting up the Builder will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Builder, enter the tool's input image and select units.

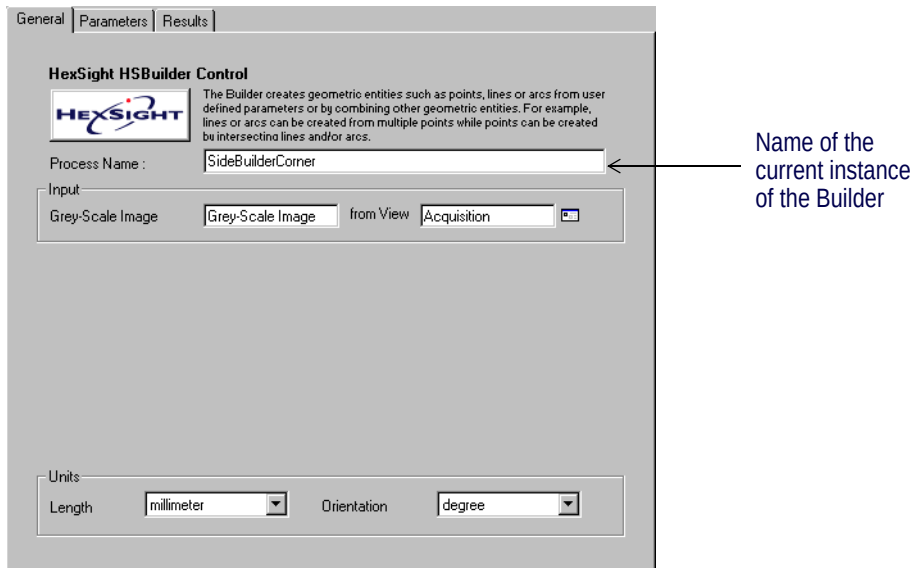


Figure 369 Builder— General Panel

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSBUILDER#, where # is replaced by its rank number in the list of instantiated processes.

Input

Image

The input **Image** is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image output by a HexSight tool can be used.

- Type the name of the Image in the **Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Units

Length

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions.

Orientation

Use the **Orientation** combo box to select the unit used to express angles.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Parameters Panel

Use this panel to configure the inputs and outputs of the Builder. Inputs required by the Builder depend on the type (Mode) of entity being built and can consist of parameters, entities, or both.

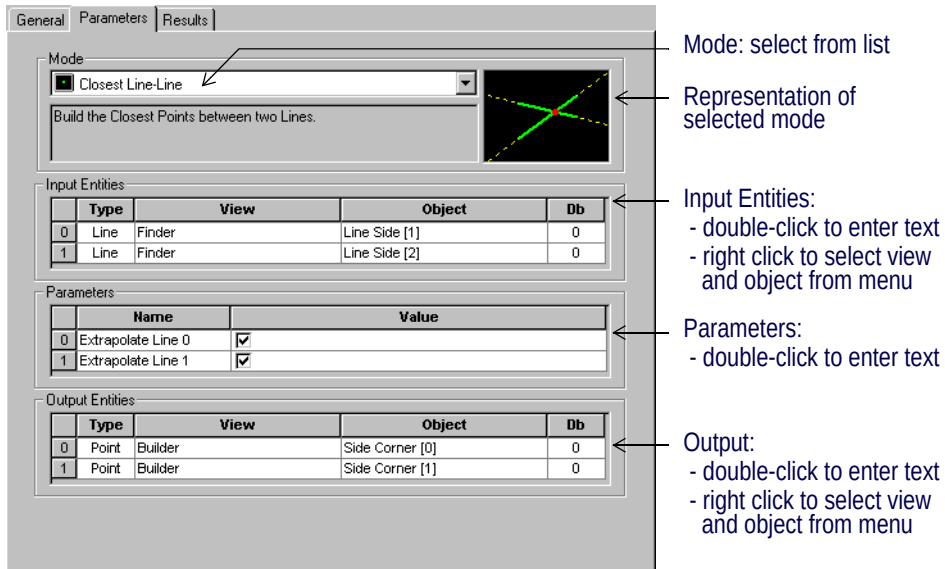


Figure 370 Builder Tool — Parameters Panel

Mode

The Mode specifies the type of object that the Builder will construct. The selected Mode determines the required input entities and parameters.

The interface provides both a description and a visual representation of the selected mode. provides a complete list of the different modes and their respective description and input requirements.

- Select a Mode from the drop-down list. The **Input Entities** and/or **Parameters** required for this mode will then be displayed in the interface.

Table 38 Builder Tool Modes for Building Entities

Mode Name	Short description	Type of Input Required
Fixed Point	Build a Point from coordinates.	Input Entities: None Parameters: Coordinates of the point, Calibrated Units

Table 38 Builder Tool Modes for Building Entities

Midway Point	Build a Point at the Midway between two Points.	Input Entities: Points Parameters: None
Vector Point of a Line	Build a Point at the Vector Point of a Line.	Input Entity: Line Parameters: None
Start Point of a Line	Build a Point at Start Point of a Line.	Input Entity: Line Parameters: None
End Point of a Line	Build a Point at End Point of a Line.	Input Entity: Line Parameters: None
Center Point of an Arc	Build a Point at Center Point of an Arc.	Input Entity: Arc Parameters: None
Start Point of an Arc	Build a Point at Start Point of an Arc.	Input Entity: Arc Parameters: None
End Point of an Arc	Build a Point at End Point of an Arc	Input Entity: Arc Parameters: None
Closest Point-Line	Build the Closest Point between a Point and a Line.	Input Entity: Point & Line Parameters: Extrapolation
Closest Point-Arc	Build the Closest Point between a Point and an Arc.	Input Entity: Point & Arc Parameters: Extrapolation
Closest Line-Line	Build the Closest Points between two Lines.	Input Entity: 2 lines Parameters: Extrapolation
Closest Line-Arc	Build the Closest Points between a Line and an Arc.	Input Entity: Line & Arc Parameters: Extrapolation
Closest Arc-Arc	Build the Closest Points between two Arcs.	Input Entity: 2 Arcs Parameters: Extrapolation
Fixed Line	Build a Line from parameters. (Center, Angle and Deltas)	Input Entity: None Parameters: Coordinates of vector point, deltas from vector point, line angle, Calibrated Units
Fixed Line (2 points)	Build a Line from coordinates. (Start and End Points)	Input Entity: None Parameters: Coordinates of start point, and end point, Calibrated Units
Line	Build a Line from two Points.	Input Entity: 2 Points Parameters: None

Table 38 Builder Tool Modes for Building Entities

Parallel Line	Build a Line parallel to a Line.	Input Entities: Line & Point Parameters: None
Perpendicular Line	Build a Line perpendicular to a Line.	Input Entities: Line & Point Parameters: None
Fixed Arc	Build an Arc from parameters. (Center, Angles and Radius)	Input Entities: None Parameters: Coordinates of Arc Center Point, Arc Start and End angles, Calibrated Units
Fixed Arc (3 points)	Build an Arc from coordinates. (Start, Mid and End Points)	Input Entity: None Parameters: Coordinates of start point, end point and point midway along the arc, Calibrated Units
Arc	Build an Arc from three Points.	Input Entity: 3 points Parameters: None
Fixed Circle	Build a circle from parameters. (Center and Radius)	Input Entities: None Parameters: Coordinates of the circle center, radius, Calibrated Units
Fixed Circle (2 points)	Build a Circle from coordinates. (Center and on Circle Point)	Input Entities: None Parameters: Coordinates of circle center and of a point on the circle, Calibrated Units
Fixed Circle (3 points)	Build a Circle from coordinates. (Three Points on Circle)	Input Entities: None Parameters: Coordinates of three points on the circle, Calibrated Units
Circle	Build a Circle from three Points.	Input Entity: 3 Points Parameters: None

Input Entities

Compatible Input Entities are geometric entities that have been created and output to the runtime database by HexSight Finder tools: Arc Finder, Line Finder, Point Finder, or another instance of the Builder tool.

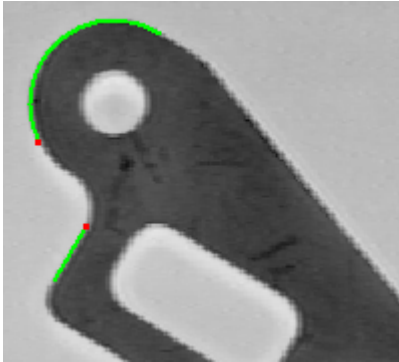
- Click in the grid and type the name of the **View** that contains the required entity; type the name of the Entity in the **Object** column.
- Alternately, you can right-click on the grid to obtain a menu of compatible Entities currently available in the runtime database.
- Typically the value in **Db** will be 0, which corresponds to the runtime database.

Parameters

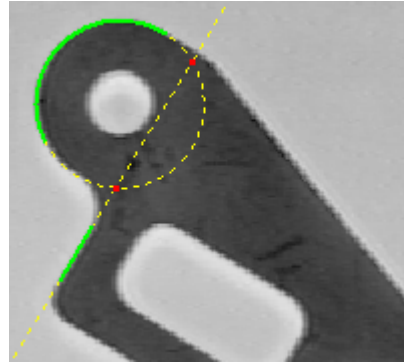
The Parameters displayed in the Interface depend on the selected Mode.

- Click in the **Value** column to type a value for the selected parameter.
- To specify a parameter value in pixels, you must disable the **Calibrated Units** check box. When the **Calibrated Units** is checked, the units applied are the units selected in the General panel.
- Enable **Extrapolate Line** or **Extrapolate Arc** to use the extrapolated entity to compute and generate the **Output Entity**. See Figure 371.

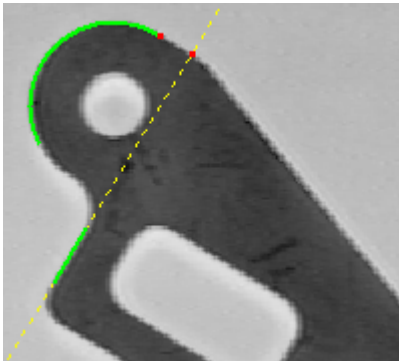
Note:Extrapolation can strongly affect the results of the Builder. The example in Figure 371 illustrates extrapolation combinations for the **Closest Line-Arc** Mode.



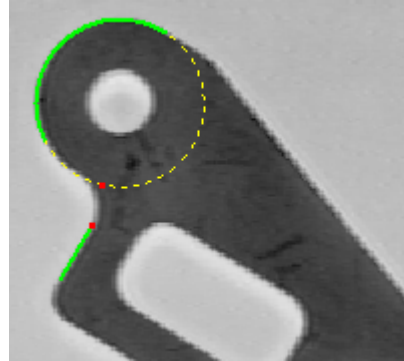
No extrapolation



Both Line and Arc are extrapolated



Only Line is extrapolated



Only Arc is extrapolated

Figure 371 Extrapolation of geometric entities - Closest Arc-Line example

Output Entities

Entities generated by the Builder are output to the application's runtime database. These entities can be used by other processes such as another instance of the Builder tool.

- Click in the grid and type the name of **View** that contains the required entity and type the name of the Entity under **Object**.

- Alternately, you can right-click on the grid to obtain a menu of *compatible* Views and Entities currently available in the runtime database.

Results Panel

The Results panel shows the results of the Builder process, including a visual display of the output entities.

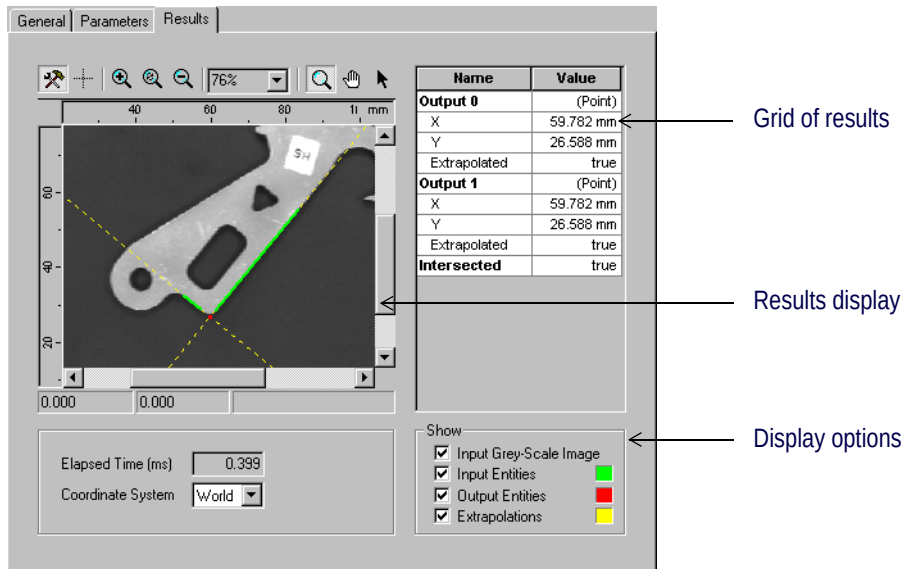


Figure 372 Builder — Results Panel

Elapsed Time

Elapsed Time shows the total execution time of the Builder.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image** and **World**.

Results Display

The **Results Display** shows the generated entity. Use the Calibrate button of the **Results Display** to toggle between calibrated and uncalibrated display modes.

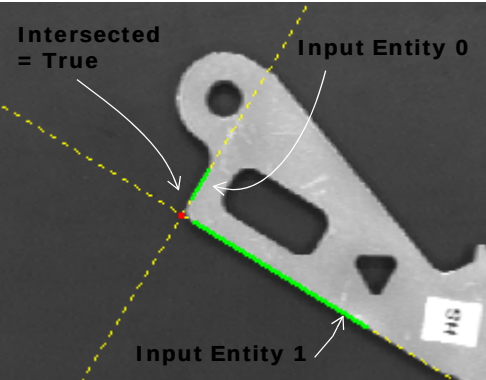
Show check boxes enable various display options:

- Enable **Input Image** to display the current input Image.
- Enable **Input Entities** to display current input entities (when applicable) used by the tool (green).
- Enable **Output Entities** to display current output entities (red).
- Enable **Extrapolations** to display extrapolations of entities (yellow).

Grid of Results

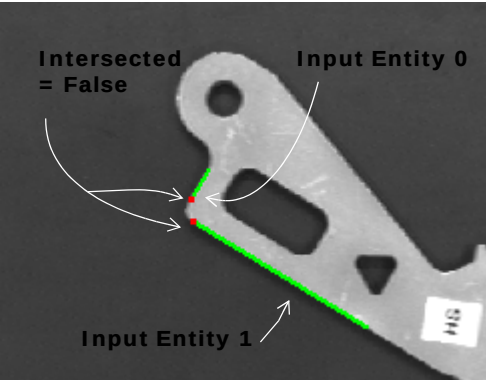
Results for the generated entity are displayed in the grid. The output results depend on the selected **Mode**. Numerical values are expressed in either the **Image** or **World** coordinate system, according to the setting of the **Coordinate System** parameter.

- The nature of the results depend on the Builder Mode.
- Builder modes of type **Closest Entity-Entity** generate two **Output Entities** and set of results for each Entity. These are labeled as **Output 0** and **Output 1**.
- For the same instance and Mode, results can vary significantly depending on the **Extrapolated** setting and the effect of extrapolation on intersection between entities.
- If Lines or Arcs are **Intersected**, outputs can be identical. **Figure 373** illustrates the outputs for a Closest Line-Line example with and without intersection.



Name	Value
Output 0	(Point)
X	48.169 mm
Y	62.502 mm
Extrapolated	true
Output 1	(Point)
X	48.169 mm
Y	62.502 mm
Extrapolated	true
Intersected	true

Output 0 = Output 1
because lines are intersected



Name	Value
Output 0	(Point)
X	50.176 mm
Y	61.287 mm
Extrapolated	false
Output 1	(Point)
X	49.789 mm
Y	65.181 mm
Extrapolated	false
Intersected	false

Output 0 ≠ Output 1

Figure 373 Effect of Intersection on results - Closest Line-Line example

Table 39 Builder Outputs and Results - Grouped by Similar Modes

Mode Name	Output Entity	Results
Fixed Point Midway Point Vector Point of a Line Start Point of a Line End Point of a Line Center Point of an Arc Start Point of an Arc End Point of an Arc	Point	X ,Y coordinates
Closest Point-Line Closest Point-Arc Closest Line-Line Closest Line-Arc Closest Arc-Arc	Point 0	X ,Y coordinates Extrapolated (True/False)
	Point 1	X ,Y coordinates Extrapolated (True/False)
		Intersected (True/False)
Line Fixed Line Fixed Line (2 points) Parallel Line Perpendicular Line	Line	Vector Point X ,Y coordinates Start Point X ,Y coordinates End Point X ,Y coordinates Angle
Arc Fixed Arc Fixed Arc (3 points)	Arc	Center Point X ,Y coordinates Start Point X ,Y coordinates End Point X ,Y coordinates Radius Start Angle, End Angle
Fixed Circle Fixed Circle (2 points) Fixed Circle (3 points) Circle	Arc	Center Point X ,Y coordinates Start Point X ,Y coordinates End Point X ,Y coordinates Radius Start Angle; 0 End Angle: 360

Image Processing Tool

The Image Processing Tool processes images by applying arithmetic, assignment, logical, filtering, morphological or histogram operations. Users can also define and apply custom filtering operations.

Image Processing Tool Chapter Outline

[Understanding the Image Processing Tool](#)

[Image Types](#)

[Elements of an Operation](#)

[Arithmetic Operations](#)

[Assignment Operations](#)

[Conversion Operations](#)

[Transform Operations](#)

[Logical Operations](#)

[Filtering Operations](#)

[Morphological Operations](#)

[Histogram Operations](#)

[Using the Image Processing Tool](#)

[General Panel](#)

[Processing Panel](#)

[Results Panel](#)

Understanding the Image Processing Tool

Each Image Processing Tool in application performs a selected operation on an image called the input image. An image processing operation can also involve another image or a constant, as well as set of processing parameters.

Image Types

The tool can accept unsigned 8-bit, signed and unsigned 16-bit and signed 32-bit images as input. The processing is usually performed in the more defined type, based on input or operand image, or in a promoted type (signed 16-bit) if needed.

The Image Processing Tool output is of same type as the input image unless:

- a) the user overrides the type by setting another value or
- b) an output image already exists: the output image type remains the same unless otherwise specified.

Elements of an Operation

An image processing operation requires at least one operand that is acted upon by an operation. For the Image Processing Tool, this first operand is always the Input Image. Some operations require a second operand, designated simply as the Operand. This Operand can be an image, the Operand Image, or a constant. The basic elements of an operation are illustrated in [Figure 374](#). Furthermore, some operations involve other parameters such as clipping, scaling and filters. Such parameters are discussed under the category of operation to which they apply.

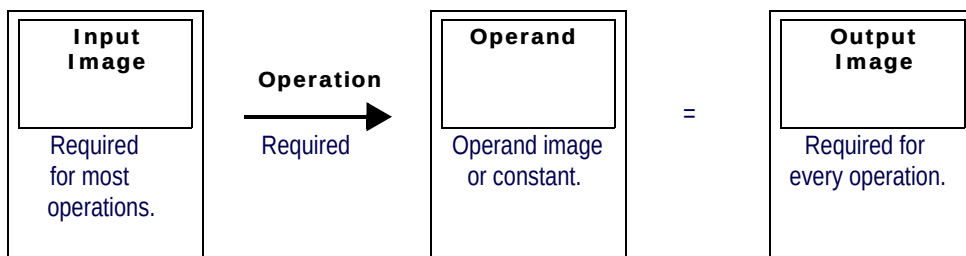


Figure 374 Basic Elements of an Image Processing Operation

Input Image

An Input image is typically required as the first operand. The only type of operation that does not require an Input Image is an Assignment operation.

Operation

The available operations are described in greater detail under the following sections: ***Arithmetic Operations***, ***Assignment Operations***, ***Logical Operations***, ***Filtering Operations***, ***Morphological Operations*** and ***Histogram Operations***.

Operand

Operations that require a second operand can use either an Operand image or a constant.

Operand Image

The Image Processing Tool can accept unsigned 8-bit, signed 16-bit and signed 32-bit images as an Operand Image.

Constant

Any constant value specified for an operation will be overridden by an Operand Image that has been defined for the operation.

Output Image

The output image is the image resulting from an image processing operation. The user can specify the type of output images as either unsigned 8-bit, signed 16-bit or signed 32-bit images. HexSight processes other than the Image Processing Tool can only take unsigned 8-bit images as input.

Arithmetic Operations

Arithmetic operations are performed by promoting the input values of the source pixels (from the Input image) and the Operand values to the more defined type, based on the Input Image, Operand image or desired output image type. The results of the operation are converted according to the following rules:

- a) Destination pixel value = ClipMode (Result * Scale)
- b) Destination pixel value is truncated as necessary

Clipping Modes

Two clipping modes are available: normal and absolute.

Normal Clipping Mode

The normal mode forces the destination pixel value to a value from 0 to 255 for unsigned 8-bit images, to a value from -32768 to 32767 for signed 16 bits images and so on. Values that are less than the specified *minimum* value are set to the *minimum* value. Values greater than the specified *maximum* value are set to the *maximum* value.

Absolute Clipping Mode

The absolute clipping mode takes the absolute value of the result and clips it using the same algorithm as for normal clipping mode.

Arithmetic Operation Modes

There are two Arithmetic operation modes. In the first, the operation is applied to every pixel of an input image and the corresponding pixel in the Operand image. The result is written in the output image.

In the second mode, the operand is a constant, and it is used on every pixel of the input image and the result is written in the output image.

The Image Processing Tool supports the following arithmetic operations: Addition, Subtraction, Multiplication, Division, Lightest and Darkest.

Addition

The Operand value (constant or Operand image pixel) is added to the corresponding pixel in the input image. The result is scaled and clipped, and finally written to the output image.

Subtraction

The Operand value (constant or Operand image pixel) is subtracted from the corresponding pixel in the input image. The result is scaled and clipped, and finally written to the output image.

Division

The input image pixel value is divided by the Operand value (constant or corresponding Operand image pixel). The result is scaled and clipped, and finally written to the output image.

Multiplication

The input image pixel value is multiplied by the Operand value (constant or corresponding Operand image pixel). The result is scaled and clipped, and finally written to the output image.

Lightest (Maximum)

The Operand value (constant or Operand image pixel) and corresponding pixel in the input image are compared to find the maximal value. The result is scaled and clipped, and finally written to the output image.

Darkest (Minimum)

The Operand value (constant or Operand image pixel) and corresponding pixel in the input image are compared to find the minimal value. The result is scaled and clipped, and finally written to the output image.

Assignment Operations

Assignment operations promote the input values of the source pixels and the Operand values to the more defined type, based on the input image, the Operand image or the desired output image type. This type of operation does not support scaling or clipping. The Image Processing Tool provides the following assignment operations: Initialization, Copy and Inversion.

Initialization

All the pixels of the output image are set to a specific constant value. The height and width of the output image must be specified.

Copy

Each input image pixel is copied to the corresponding output image pixel.

Inversion

The input image pixel value is inverted and the result is copied to the corresponding output image pixel.

Conversion Operations

Conversion operations are used to convert the input image to another format. This type of operation does not support scaling or clipping. The Image Processing Tool provides the following conversion operations: To intensity Dynamic Range and To intensity Static Range.

To Intensity Dynamic Range

Used to convert an image to a greyscale image, this mode will use the current range of the input image pixels to stretch or compress the spectrum of values along the whole 8 bit range. When used on a height map, the invalid value will remain invalid.

To Intensity Static Range

Converts an image to a greyscale image by changing all the pixels included in the specified range of values, to use the whole 8 bit range (0-255).

Transform Operations

Transform operations convert and output a frequency description of the input image. The available operations are a **Fast Fourier Transform (FFT)** and a **Discrete Cosine Transform (DCT)**. These transforms can be output as **1D Linear**, **2D Linear**, **2D Logarithmic** or **Histogram**.

Logical Operations

There are two logical operation modes. In the first, the operation is applied to every pixel of an input image and the corresponding pixel in the Operand image. The result is written in the output image. In the second mode, the operand is a constant, and it is used on every pixel of the input image and the result is written in the output image. No scaling or clipping is supported for logical operations. The Image Processing Tool provides the following logical operations: And, Nand, Nor, Or and Xor.

AND

The logical And operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image. The result is written to the output image.

NAND

The logical Nand operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image. The result is written to the output image.

NOR

The logical Nor operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image. The result is written to the output image.

OR

The logical Or operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image. The result is written to the output image.

XOR

The logical Xor operation is applied using the Operand value (constant or Operand image pixel) and the corresponding pixel in the input image. The result is written to the output image.

Filtering Operations

A filtering operation can be described as the convolution of an input image using a square, rectangular or linear kernel. The Image processing tools provides a set of

defined filters as well as a custom filtering operation that applies a user-defined kernel. The predefined filters are: Average, Gaussian, Horizontal Prewitt, Vertical Prewitt, Horizontal Sobel, Vertical Sobel, High Pass, Laplacian, Sharpen, SharpenLow and Median.

The Scale and Clipping parameters described previously under **Arithmetic Operations** also apply to filtering operations.

Average

The **Average** operation sets each pixel in the output image as the average of all the input image pixels in the neighborhood defined by the selected kernel size. This has the effect of blurring the image, especially edges.

The average filters are designed to remove noise. The kernel size can be 3, 5 or 7. The kernels used by the Image Processing Tool are shown [Figure 375](#).

1	1	1
1	1	1
1	1	1

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

Figure 375 Average Filtering Kernels

Gaussian

The Gaussian operation acts like a low pass filter. This has the effect of blurring the image. Gaussian filters are designed to remove noise. The kernel

size can be 3, 5 or 7. The kernels used by the Image Processing Tool are shown in [Figure 376](#).

1	2	1
2	4	2
1	2	1

2	7	12	7	2
7	31	52	31	7
12	52	127	52	12
7	31	52	31	7
2	7	12	7	2

1	1	2	2	2	1	1
1	3	4	5	4	3	1
2	4	7	8	7	4	2
2	5	8	10	8	5	2
2	4	7	8	7	4	2
1	3	4	5	4	3	1
1	1	2	2	2	1	1

Figure 376 Gaussian Filtering Kernels

Horizontal Prewitt

The Horizontal Prewitt operation acts like a gradient filter. This has the effect of highlighting horizontal edges (gradients) in the image. The kernel size used is 3 and it is shown in [Figure 377](#). The absolute clipping method is usually used with this filtering operation.

1	1	1
0	0	0
-1	-1	-1

Figure 377 Horizontal Prewitt Filtering Kernel

Vertical Prewitt

The Vertical Prewitt operation acts like a gradient filter. This has the effect of highlighting horizontal edges (gradients) in the image. The kernel size used is 3 and it is shown in [Figure 378](#). Absolute clipping method is usually used with this filtering operation.

Figure 378 Vertical Prewitt Filtering Kernel

-1	0	1
-1	0	1
-1	0	1

Horizontal Sobel

The Horizontal Sobel operation acts like a gradient filter. This has the effect of highlighting horizontal edges (gradients) in the image. The kernel size can be 3, 5 or 7. The absolute clipping method is usually used with this filtering operation. The kernels used by the Image Processing Tool are shown in [Figure 379](#).

1	2	1
0	0	0
-1	-2	-1

1	4	7	4	1
2	10	17	10	2
0	0	0	0	0
-2	-10	-17	-10	-2
-1	-4	-7	-4	-1

1	4	9	13	9	4	1
3	11	26	34	26	11	3
3	13	30	40	30	13	3
0	0	0	0	0	0	0
-3	-13	-30	-40	-30	-13	-3
-3	-11	-26	-34	-26	-11	-3
-1	-4	-9	-13	-9	-4	-1

Figure 379 Horizontal Sobel Filtering Kernels

Vertical Sobel

The Vertical Sobel operation acts like a gradient filter. This has the effect of highlighting vertical edges (gradients) in the image. The kernel size can be 3, 5 or 7. The absolute clipping method is usually used with this filtering operation. The kernels used by the Image Processing Tool are shown in [Figure 380](#).

-1	0	1	-1	-2	0	2	1	-1	-3	-3	0	3	3	1
-2	0	2	-4	-10	0	10	4	-4	-11	-13	0	13	11	4
-1	0	1	-7	-17	0	17	7	-9	-26	-30	0	30	26	9
			-4	-10	0	10	4	-13	-34	-40	0	40	34	13
			-1	-2	0	2	1	-9	-26	-30	0	30	26	9
								-4	-11	-13	0	13	11	4
								-1	-3	-3	0	3	3	1

Figure 380 Vertical Sobel Filtering Kernels

High Pass

The High Pass operation acts like a circular gradient filter. This has the effect of highlighting all edges (gradients) in the image. The kernel size can be 3, 5 or 7. The absolute clipping method is usually used with this filtering operation. The kernels used by the Image Processing Tool are shown in [Figure 381](#).

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	24	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	48	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Figure 381 High Pass Filtering Kernels

Laplacian

The Laplacian operation also acts like a circular gradient filter. This has the effect of highlighting all edges (gradients) in the image. The kernel size can be 3, 5 or 7. The absolute clipping method is usually used with this filtering

operation. The kernels used by the Image Processing Tool are shown in [Figure 382](#).

-1	-1	-1	-1	-3	-4	-3	-1	-2	-3	-4	-6	-4	-3	-2
-1	8	-1	-3	0	6	0	-3	-3	-5	-4	-3	-4	-5	-3
-1	-1	-1	-4	6	20	6	-4	-4	-4	9	20	9	-4	-4
			-3	0	6	0	-3	-6	-3	20	36	20	-3	-6
			-1	-3	-4	-3	-1	-4	-4	9	20	9	-4	-4
								-3	-5	-4	-3	-4	-5	-3
								-2	-3	-4	-6	-4	-3	-2

Figure 382 Laplacian Filtering Kernels

Sharpen

The Sharpen operation sets each pixel in the output image as the subtraction of the average of all the input image pixels in the neighborhood defined by the selected kernel size. This has the effect of sharpening the image, especially edges. The kernel size can be 3, 5 or 7. The kernels used by the Image Processing Tool are shown [Figure 383](#).

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	25	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	49	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Figure 383 Sharpen Filtering Kernels

SharpenLow

The SharpenLow operation has the effect of sharpening and smoothing the image at the same time. The kernel size can be 3, 5 or 7. The kernels used by the Image Processing Tool are shown in [Figure 384](#).

$(1/8) *$

-1	-1	-1
-1	16	-1
-1	-1	-1

-1	-3	-4	-3	-1
-3	0	6	0	-3
-4	6	40	6	-4
-3	0	6	0	-3
-1	-3	-4	-3	-1

-2	-3	-4	-6	-4	-3	-2
-3	-5	-4	-3	-4	-5	-3
-4	-4	9	20	9	-4	-4
-6	-3	20	72	20	-3	-6
-4	-4	9	20	9	-4	-4
-3	-5	-4	-3	-4	-5	-3
-2	-3	-4	-6	-4	-3	-2

Figure 384 SharpenLow Filtering Kernels

Median

The Median operation sets each pixel in the output image as the median luminance of all the input image pixels in the neighborhood defined by the selected kernel size. This has the effect of reducing impulsive image noise without degrading edges or smudging intensity gradients.

Morphological Operations

Morphological operations are used to eliminate or fill small and thin holes in objects, break objects at thin points or connect nearby objects. These operations generally smooth the boundaries of objects without significantly changing their area. The Image Processing Tool provides the following predefined morphological operations, each of which can only be applied to a 3x3 neighborhood: Dilate, Erode, Close and Open.

Dilate

The Dilate operation sets each pixel in the output image as the largest luminance value of all the input image pixels in the neighborhood defined by the selected kernel size. (Currently fixed to 3x3)

Erode

The Erode operation sets each pixel in the output image as the smallest luminance value of all the input image pixels in the neighborhood defined by the selected kernel size. (Currently fixed to 3x3)

Close

The Close operation is equivalent to a Dilate operation followed by an Erode operation. This has the effect of removing small dark particles and holes within objects.

Open

The Open operation is equivalent to an Erode operation followed by a Dilate operation. This has the effect of removing peaks from an image, leaving only the image background.

Histogram Operations

The action of a histogram operation depends on the histogram of the Input Image. The Image Processing Tool provides the following histogram operations, each of which can only be applied to an unsigned 8-bit image: Equalization, Stretching, Light Threshold and Dark Threshold.

Equalization

The Equalization operation enhances the Input Image by flattening the histogram of the Input Image.

Stretching

The Stretching operation stretches (increases) the contrast in an image by applying a simple piecewise linear intensity transformation based on the histogram of the Input Image.

Light Threshold

The Light Threshold operation changes each pixel value depending on whether they are less or greater than the specified threshold. If an input pixel value is less than the threshold, the corresponding output pixel is set to the minimum acceptable value. Otherwise, it is set to the maximum presentable value.

Dark Threshold

The Dark Threshold operation changes each pixel value depending on whether they are less or greater than the specified threshold. If an input pixel value is less than the threshold, the corresponding output pixel is set to the maximum presentable value. Otherwise, it is set to the minimum acceptable value.

Using the Image Processing Tool

Setup of the Image Processing Tool is carried out on three tab-selected panels: General, Processing, and Results. Steps for using and setting up the Image Processing Tool will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Image Processing Tool and enter the tool's input and output Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

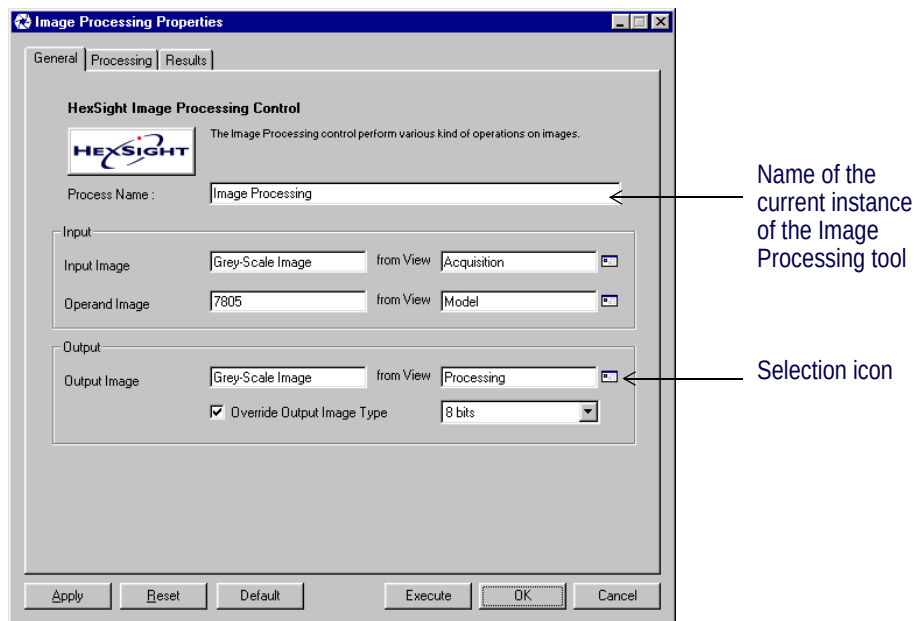


Figure 385 Image Processing— General Panel

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a

default name: HSIImageProcessing#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database to which an operation will be applied. The Image Processing can accept unsigned 8-bit, signed 16-bit and signed 32-bit images as Input Image.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Operand Image

An **Operand Image** is used by an operation that acts on two images. If and Operand Image is specified it will override the use of the constant specified for the operation. The Image Processing Tool can accept unsigned 8-bit, signed 16-bit and signed 32-bit images as Operand Image.

The Image processing tool applies logical and arithmetic operators, first to the **Input Image**, secondly to the **Operand Image**. Type the name of the Image in the **Operand Image** text box and the name of the View containing this Image in the associated **from View** text box.

- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Output Image

Output Image is the resulting Image that will be created in the runtime database by the tool.

- Type the name of the image in the **Output Image** text box and the name of the View containing this image in the associated **from View** text box.

- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database. Since this is an output image, it will overwrite the selected Image in the runtime database when the tool is executed.

Override Output Image Type

By default, the Image Processing Tool outputs all resulting images as unsigned 8-bit images. To change the type of output image, enable the **Override Output Image Type** check box and select an image type.

HexSight processes, other than the Image Processing Tool, can only use 8-bit images. Unsigned 16 or 32-bit images will be output to the runtime database but will not be available as input for HexSight processes other than additional Image Processing Tools.

Processing Panel

Use this panel to select the processing operator and to set parameters, if any, for the selected operator.

For more details on each of the processing operators, refer to the ***Understanding the Image Processing Tool***.

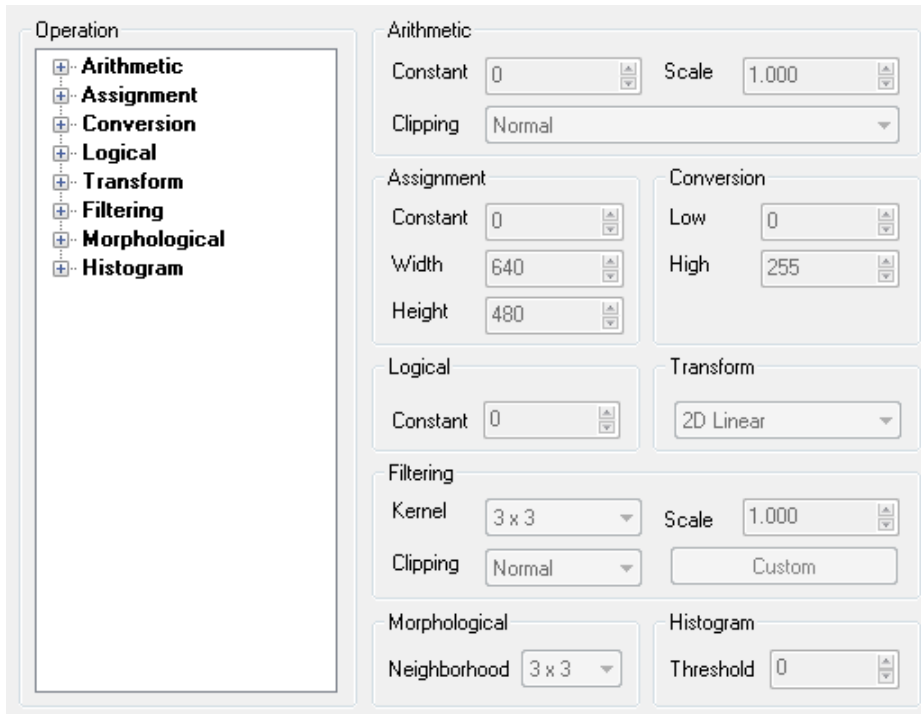


Figure 386 Image Processing Tool — Processing Panel

Operation

Use the **Operation** box to select an image processing operation.

- Use the - and + signs to collapse and expand the list choices
- Select an operator by clicking on its name in the list

Arithmetic

Use the **Arithmetic** box to set the parameters for an arithmetic operator.

- The value entered in the **Constant** text box is applied as Operand if you did not specify an Operand Image in the General panel.

- The value entered in the **Scale** text box multiplies the value of each pixel after the operation has been applied.
- Select one of two **Clipping** modes: Normal or Absolute. See ***Understanding Image Processing Tool Concepts*** for more details on these modes.

Assignment

Use this box to set the parameters for an assignment operation.

- The value entered in the **Constant** text box is applied as Operand if you did not specify an Operand Image in the General panel.
- Use the **Width** text box to set the width, in pixels, of the resulting Output Image.
- Use the **Height** text box set the height, in pixels, of the resulting Output Image.

Conversion

Use this box to set the parameters for a conversion operation.

- Use the **Low** text box to set the lower value of the input range to convert.
- Use the **High** text box set the higher value of the input range to convert.

Transform

Use this section to set the parameters for a transform operation.

- Available transforms are **Fast Fourier** and **Discrete Cosine**.
- Use the **Transform** combo box to select the format /scale in which the transform operation results will be output: **1D Linear**, **2D Linear**, **2D Logarithmic** or **Histogram**.

Logical

Use this section to set the parameters for a logical operation. The value entered in the **Constant** text box is applied as Operand if you did not specify an Operand Image in the General panel.

Filtering

Use this section to set the parameters for a filtering operation. Filtering operations do not apply an Operand.

- Use the **Kernel** combo box to select a kernel size for a predefined filter. If you have selected a **Custom** filter, you must specify the kernel size, and other kernel parameters in the **Custom Kernel Properties** dialog (Click **Custom**).
- The value entered in the **Scale** text box multiplies the value of each pixel after the operation has been applied.
- Select one of two **Clipping** modes: Normal or Absolute. Typically, the Absolute clipping mode is used for filter operations. See ***Understanding Image Processing Tool Concepts*** for more details on clipping modes.
- The **Custom** button opens the **Custom Kernel Properties** Dialog. See ***Creating a Custom Kernel***.

Creating a Custom Kernel

The **Custom** button opens a **Custom Kernel** dialog, which you will use to create the kernel for the custom filter operation. [Figure 387](#) illustrates a 3x3 kernel.

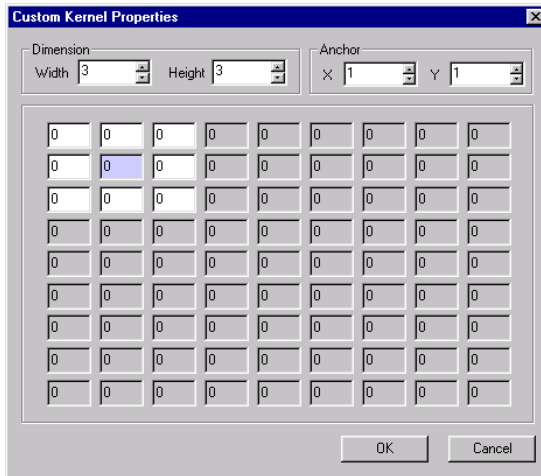


Figure 387 Image Processing Tool — Custom Kernel Dialog

- In the **Dimensions** box, enter values for width and height of the kernel. Grid boxes in white indicate kernel elements.
- Enter the required value in each white box of the kernel grid.
- In the **Anchor** box, enter the X and Y positions of the kernel anchor, with respect to the defined kernel. The box indicating the anchor position appears in color.

Morphological

Use the **Morphological** box to set the parameters for a morphological operation.

- Use the **Neighborhood** combo box to select the neighborhood that will be applied by the operation.

Histogram

Use the **Histogram** box to set the parameters for a histogram operation by entering the required **Threshold** value.

Results Panel

The Results panel provides information on the output of the Image Processing Tool. The display provides options for showing the **Input Image**, the **Operand Image** and the **Output Image**.

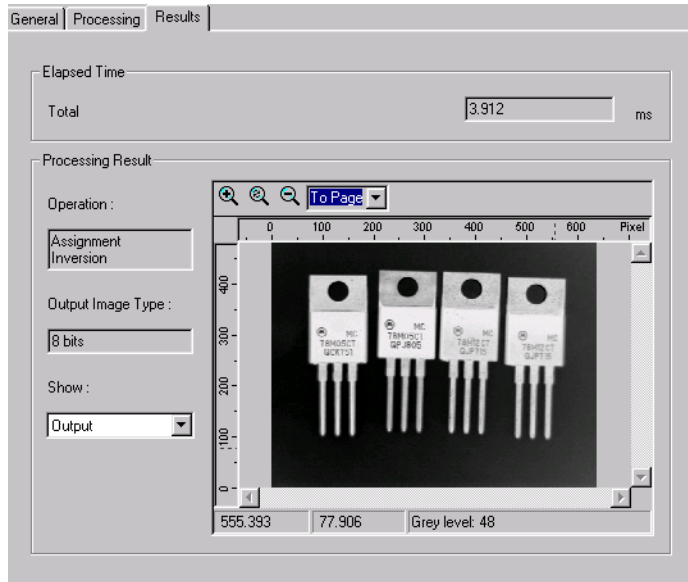


Figure 388 Image Processing Tool — Results Panel

Elapsed Time

This text box displays the time for the execution of the tool.

Processing Results

Use the **Show:** combo box to select the image shown in the display: Output Image (result), Input Image or Operand Image.

Display

The display rulers and status bar provide pixel values as well positional values in pixel units.

Image Histogram

The Image Histogram Tool computes greylevel statistics within a user-defined region of interest.

Image Histogram Chapter Outline

[Understanding Image Histogram Tool Concepts](#)

[Image Histogram Basics](#)

[Image Histogram Results](#)

[Using the Image Histogram Tool](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Image Histogram Tool Concepts

The Image Histogram tool is normally used in applications where the greylevel distribution of an image needs to be compared against an ideal known distribution. Typical cases for using this tool are verifying and the validating the camera iris adjustment or the lighting setup of an application.

[Figure 389](#) illustrates the effect of varying image brightness on the greylevel distribution.

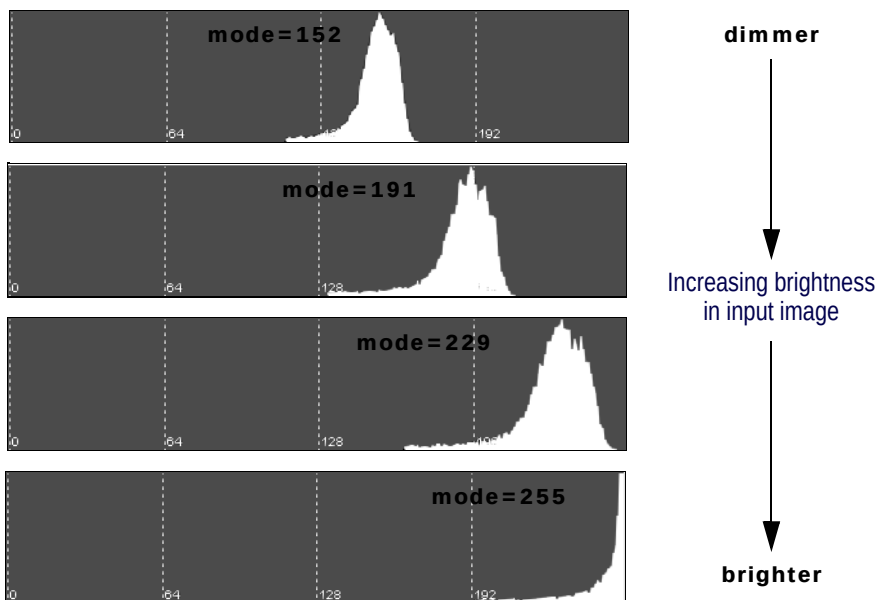


Figure 389 Effect of Image Brightness on the Histogram

Image Histogram Basics

The Image Histogram tool calculates greylevel statistics for a selected region of interest. The final histogram, for which the tool calculates the statistics ignores pixels that have been excluded by thresholds or tails.

Histogram

The Histogram provides the distribution of all the pixel values contained in the Rectangle region of interest. Pixels can be excluded from the distribution by thresholds or tail functions.

- Since HexSight uses 8-bit grey-scale images, pixel greylevel values in the histogram range from 0 (black) to 255 (white).

Region of Interest

The region of interest that contains the histogram is bounded by a Rectangle that can be freely rotated and translated. This Rectangle can be Model-based in the same manner as Inspection tools. See ***Tool Positioning Modes*** in the ***Inspection Tools*** chapter.

Thresholds

Thresholds are used to exclude a range of pixel values from the histogram. The **White** threshold excludes light pixels, having a greylevel value higher than the threshold value. The **Black** threshold excludes dark pixels, having a greylevel value lower than the threshold value.

- The example illustrated [Figure 390](#) demonstrates the use of a threshold to remove background pixels surrounding an object of interest.

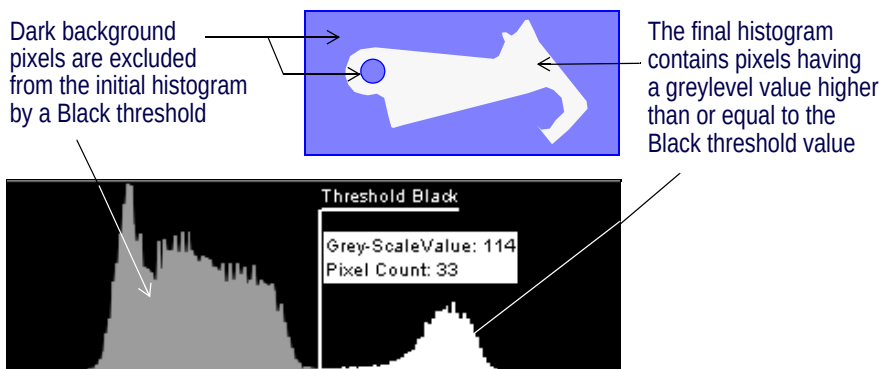


Figure 390 Applying a Threshold to Exclude Background Pixels

- Another reason for using thresholds is to remove frame grabber artifacts, such as bars on the edges of an image as illustrated in [Figure 391](#).

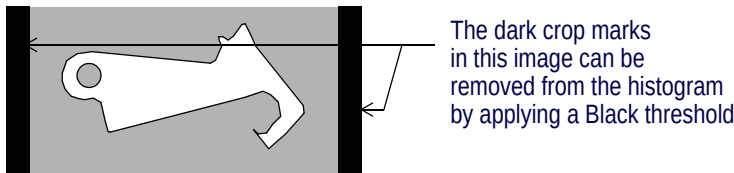


Figure 391 Applying a Threshold to Exclude Artifacts

Tails

Tail values are typically used to remove from the distribution, pixels with spurious greylevels, caused by noise in images. This will often allow you obtain a more uniform pixel distribution than fixed thresholds in cases where the brightness level changes from one image to another.

A tail specifies an amount pixels to be removed from the dark and light ends of the initial histogram. This value expresses a percentage of the total number of pixels in the histogram before tails are removed.

- The **Black** tail value sets the percentage of pixels to ignore at the dark end of the greylevel distribution in the tool's region of interest. After its creation, the histogram is scanned, starting from bin 0. The bins at the dark end of the histogram are then cleared until the amount of pixels defined by **Black** value is reached.
- The **White** tail value sets the percentage of pixels to ignore at the light end of the greylevel distribution in the tool's region of interest. After its creation, the histogram is scanned, starting from bin 255. The bins at the bright end of the histogram are then cleared until the amount of pixels defined by **White** value is reached.
- Note that if both a Black threshold and a Black tail are configured, the threshold is first applied and the tail is then removed from the resulting histogram. Likewise, if both a white threshold and a white tail are used, the tail is removed from the histogram after the threshold is applied.

Image Subsampling

The image subsampling function coarsely resamples the image in the tool's region of interest. This can be advantageously used when the image does not contain high frequency transitions or textures.

In such cases, the subsampled image is processed more quickly, without having a negative impact on the accuracy of the results.

Image Histogram Results

The results of the tool are given for the final histogram, that is the histogram defined by the tool's region of interest, after any thresholds or tails have been removed. Note that the tool region of interest can be subsampled prior to the application of tails or thresholds.

Statistics

Statistical results provided for the histogram are as follows.

- **Mean:** Mean of the greylevel distribution in the histogram.
- **Median:** Median of the greylevel distribution in the histogram.
- **Variance:** Variance of the greylevel distribution in the histogram.
- **Standard Deviation:** Standard deviation of the greylevel distribution in the histogram.
- **Mode:** Mode of the greylevel distribution in the histogram. The mode is the greylevel value which corresponds to the histogram bin with the highest number of pixels
- **Mode Pixel Count:** Number of pixels in the histogram bin which corresponds to the mode of the greylevel distribution in the histogram.
- **Minimum Greylevel Value:** Lowest greylevel value found in the histogram.
- **Maximum Greylevel Value:** Highest greylevel value found in the histogram.

- **Greylevel Range:** The range of greylevel values in the histogram; this is equal to [**Maximum Greylevel Value** - **Minimum Greylevel Value** + 1]
- **Tail Black Greylevel Value:** represents the darkest greylevel value that remains in the histogram after the tail is removed.
- **Tail White Greylevel Value:** Represents the brightest greylevel value that remains in the histogram after the tail is removed.

Histogram Data

The tool interface provides a grid showing the number of pixels in each histogram bin, from 0 to 255.

Pixel Data

- **Histogram Pixel Count:** The number of pixels in the histogram. Equal to the sum of all of the histogram bins.
- **Image Pixel Count:** Number of pixels in the tool's region of interest.
- **Image Width:** The length, in pixels, along the X-axis, of the tool's region of interest.
- **Image Height:** The height, in pixels, along the Y-axis, of the tool's region of interest.

Using the Image Histogram Tool

Setup of the Image Histogram is carried out on five tab-selected panels: General, Location, Transform, Configuration and Results. Steps for using and setting up the Image Histogram will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Image Histogram and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

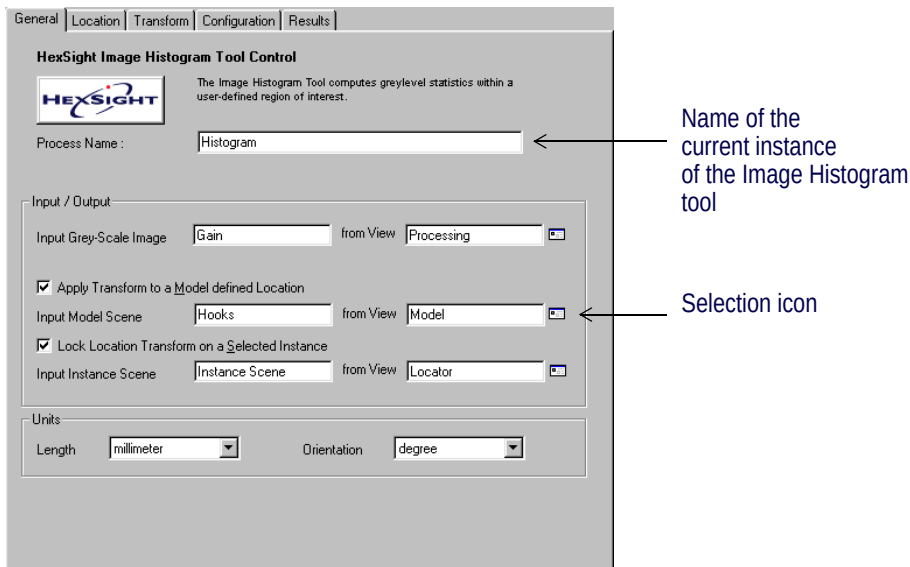


Figure 392 Image Histogram — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**
- **Express Position and Size using Calibrated Units**

[Table 40](#) provides an overview of the required inputs and settings for the four positioning modes.

Table 40 Positioning Modes: Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 40 Positioning Modes: Inputs and Settings

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSIImageHistogram#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes.

When enabling Model-based positioning, you must also specify an input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Locator. This Scene is provided by a Locator configured to output its Instance Scene in the runtime database.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

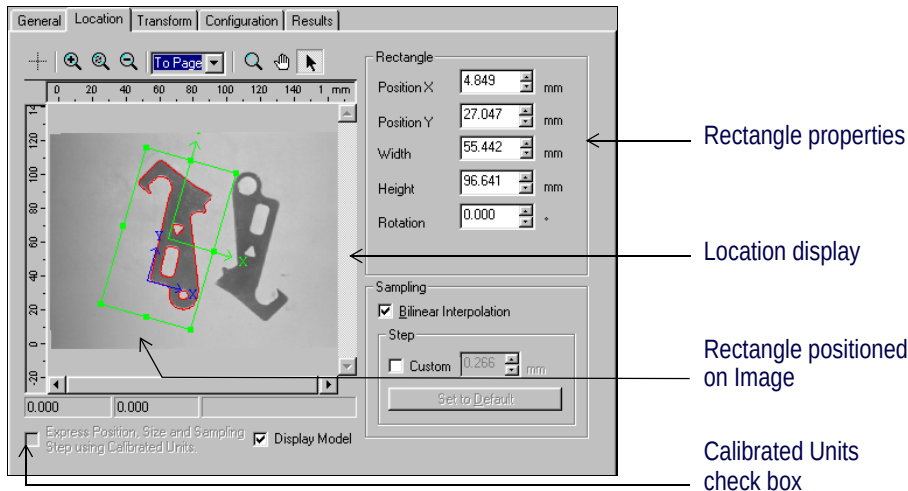


Figure 393 Image Histogram — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the tool's region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is

executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

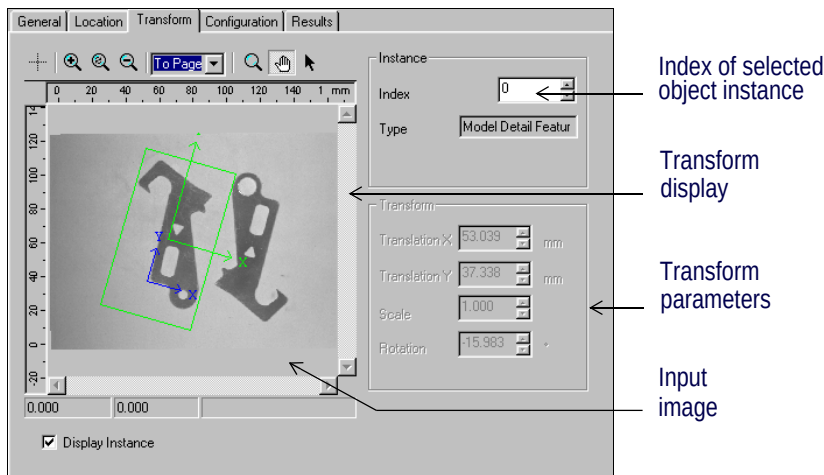


Figure 394 Image Histogram — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, the index ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X, Translation Y, Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to set parameters for building the Histogram. This panel provides two displays that illustrate the effect of the tool's configuration.

Thresholds

Thresholds exclude pixels from the histogram according their greylevel value.

- **Black** sets the threshold for excluding dark pixels. All pixels with a greylevel value lower than the threshold value are excluded from the histogram.
- **White** sets the threshold for excluding light pixels. All pixels with a greylevel value higher than the threshold value are excluded from the histogram.

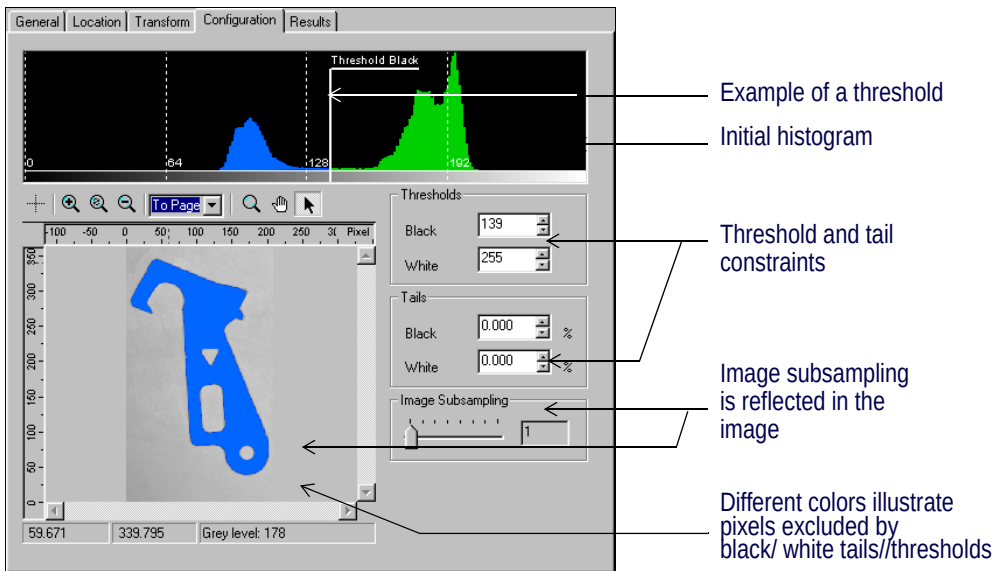


Figure 395 Image Histogram — Configuration Panel

Tails

Tails specify a percentage of pixels to be ignored from the dark and light ends of the histogram.

- A **Black** tail specifies the amount of dark pixels to exclude from the histogram, starting from the dark end of the histogram distribution (**0**). The amount of pixels to exclude is expressed as a percentage of the total number of pixels in the tool's region of interest.
- A **White** tail specifies the amount of light pixels to exclude from the histogram, starting from the light end of the histogram distribution (**255**). The amount of pixels to exclude is expressed as a percentage of the total number of pixels in the tool's region of interest.

Image Subsampling

Use the **Image Subsampling** slider to set a subsampling level from **1**, which means no subsampling, to **8**, where a subsampled pixel represents a tile of 8 by 8 pixels in the original image.

Image sampling in can significantly improve execution speed. It is useful for cases in which the image does not have high frequency transitions or textures and in which the averaging process does not significantly affect the statistics.

Histogram Displays

Both displays in this panel, the histogram and the image, use the same color scheme to illustrate the pixels excluded from the histogram.

- Blue: pixels excluded by a **Black** threshold.
- Yellow: pixels excluded by a **White** threshold.
- Light blue: pixels excluded by a **Black** tail percentage.
- Orange: pixels excluded by a **White** tail percentage.

Results Panel

The Results panel shows the results of the Image Histogram process.

Results Display

The Histogram display represents the final histogram, after thresholds and tails have been applied.

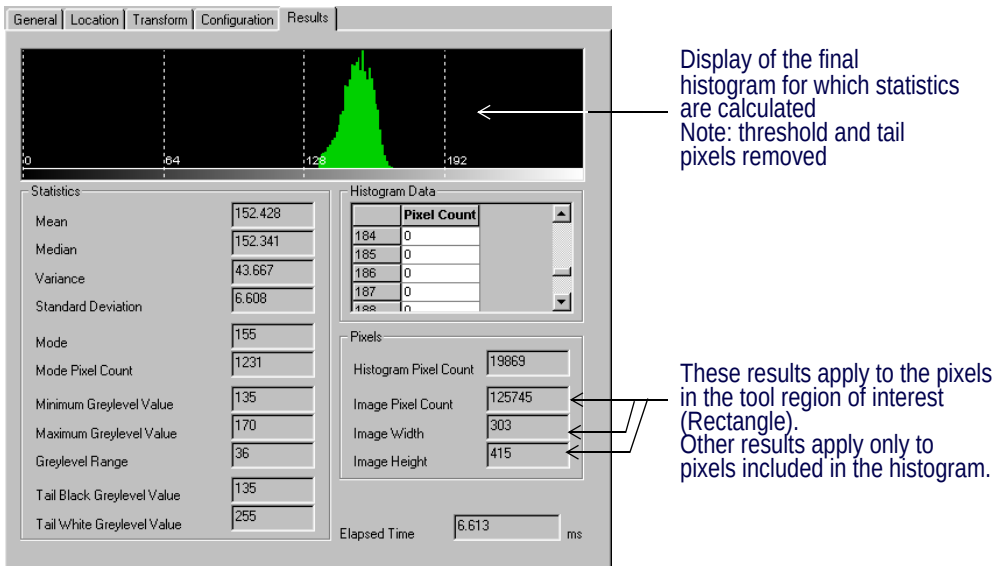


Figure 396 Image Histogram — Results Panel

Statistics

The histogram statistics provided by the tool are as follows:

- The **Mean** of the greylevel distribution in the histogram.
- The **Median** of the greylevel distribution in the histogram.
- The **Variance** of the greylevel distribution in the histogram.
- The **Standard Deviation** of the greylevel distribution in the histogram.
- The **Mode** of the greylevel distribution, which corresponds to the greylevel value for which there is the highest number of pixels.
- The **Mode Pixel Count** is the number of pixels in the histogram that corresponds to the mode value of the greylevel distribution.
- The **Minimum Greylevel Value** is the lowest greylevel value found in the histogram.

- The **Maximum Greylevel Value** is the highest greylevel value found in the histogram.
- The **Greylevel Range** specifies the range of greylevel values in the histogram; this is equal to [**Maximum Greylevel Value** - **Minimum Greylevel Value** + 1].
- The **Tail Black Greylevel Value** represents the darkest greylevel value that remains in the histogram after a **Black** tail is removed.
- The **Tail White Greylevel Value** represents the lightest greylevel value that remains in the histogram after a **White** tail is removed.

Histogram Data

Histogram data is provided in the interface as a grid showing the number of pixels (pixel count) for each greylevel value in the histogram. Each of these greylevel values is also referred to as a bin.

Pixels

The **Pixels** frame provides additional data for the histogram:

- **Histogram Pixel Count**: Total number of pixels in the histogram.
- **Image Pixel Count**: Number of pixels in the histogram.
- **Image Width**: X-axis length in pixels of the histogram.
- **Image Height**: Y-axis length in pixels of the histogram.

Elapsed Time

Elapsed Time shows the total execution time of the Image Histogram.

Image Sharpness Tool

The Image Sharpness Tool computes the sharpness of preponderant edges in a user-defined region of interest.

Image Sharpness Tool Chapter Outline

[Understanding Image Sharpness Concepts](#)

[Image Sharpness Basics](#)

[Image Sharpness Results](#)

[Using the Image Sharpness Tool](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Results Panel](#)

Understanding Image Sharpness Concepts

The Image Sharpness tool is basically used for verification or validation of the image sharpness of an image before it is processed by other tools. It can also be used as a building block for implementing an auto focus procedure which consists of a motorized focus lens and uses the sharpness value to close the loop.

Image Sharpness Basics

The Image Sharpness process operates by first identifying a set of points with high local grey-scale variations at various points and applying an autocorrelation method to calculate an average sharpness factor from these points.

Candidate Points

Candidate Points are the points with the highest local grey-scale variation in the region of interest. These points are candidates at which a sharpness measurement will be made if the local variation is sufficient. The number of candidate points is by default, automatically set by the tool, based on size of the region of interest.

When the tool is executed, it first scans the region of interest and identifies a number of candidate points where the local standard deviation is the highest. It then evaluates the sharpness at each of the candidate location that has a local standard deviation above the **Standard Deviation Threshold**. The locations where the sharpness is actually measured become Measurement Points.

- Candidate points are by default set automatically by the tool. When the default, and recommended, **Automatic** setting is enabled, the tool uses 500 Candidate Points for a region of interest over 320x240 pixels in size. If the area is smaller than 320x240, the number of Candidate Points is equal to: $\text{width} \times \text{height} / (500 / 320 \times 240)$
- The number of candidate points can be set manually by entering a value for the **Candidate Point Count** parameter.

Standard Deviation Threshold

Standard Deviation Threshold sets the minimum required standard deviation required for a Candidate Point to be used as a Measurement Point for calculating the average image sharpness.

When the tool is executed, it scans the region of interest and identifies a number of candidate locations, set by **Candidate Points Count**, where the local standard deviation is the highest. Points having a standard deviation equal to or above the threshold are used by the tool as the measurement points for calculating the average image sharpness.

Sharpness Operator

The Sharpness Operator is a processing operation that evaluates the blurriness at a Measurement Point using a local autocorrelation method.

The default kernel size of **5** should be appropriate in typical applications. However, the **Kernel** should be larger than the number of pixels over which a typical contrast is spread.

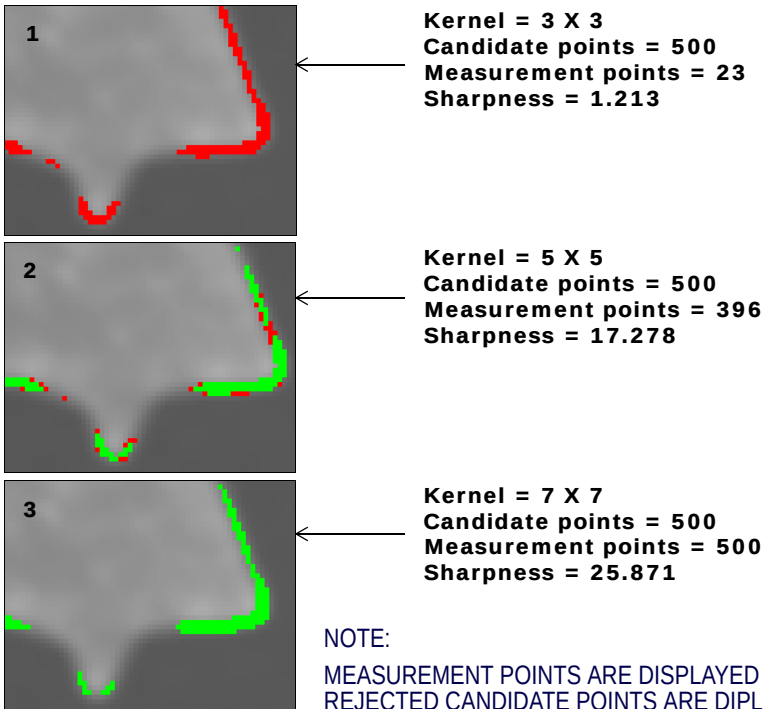


Figure 397 Effect of Kernel Operator on the Sharpness Result

A larger kernel may be used for blurrier images, for example in the case where the blurriness of the contrast is larger than the default kernel value. This is illustrated in [Figure 397](#), where the contrast is about 6-8 pixels wide. Note the difference in values obtained with different kernel sizes. With a 7X7 kernel, all the candidate points are used as measurement points. Larger kernels subsequently have almost no impact on the Sharpness result.

A smaller kernel size, **2** or **3** for example, may be helpful for images with fine details or for images constituted of fine high-frequency textures.

Image Sharpness Results

Besides the output of the Sharpness factor, the Image Sharpness tool provides a history of sharpness results obtained in prior executions of the tool. This history of results can be useful for implementing an auto focus procedure.

Sharpness Factor

The Sharpness factor output by the tool serves as an indication of the average sharpness in the input image. The range for this factor is 0 to 1000, where 1000 is the maximum attainable sharpness and 0 indicates a totally blurred image.

Sharpness History

The Image Sharpness tool stores and outputs the history of the last Sharpness values computed by the tool. The interface provides a visual display (viewer) of the Sharpness History. The interface also provides scale settings and a reset command for the viewer.

Sharpness Peak

The Sharpness Peak is the maximum average sharpness result computed by the tool. This value is reset to zero when the history is reset.

Using the Image Sharpness Tool

Setup of the Image Sharpness Tool is carried out on four tab-selected panels: General, Location, Transform, Configuration and Results. Steps for using and setting up the Image Sharpness Tool will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Image Sharpness Tool and enter the tool's input Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

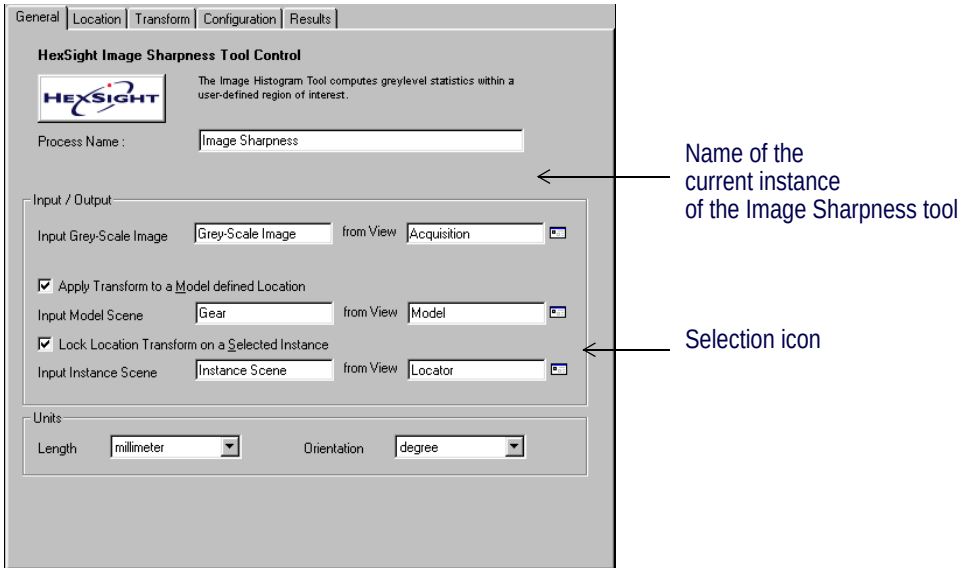


Figure 398 Image Sharpness Tool — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-based automatic. See the **Inspection Tools** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**

- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 41 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a

default name: HSIImageSharpnessTool#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a HexSight Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that provides the transform parameters for instances of objects found by a HexSight Locator. This Scene is provided by a Locator configured to output its Instance Scene to the runtime database.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

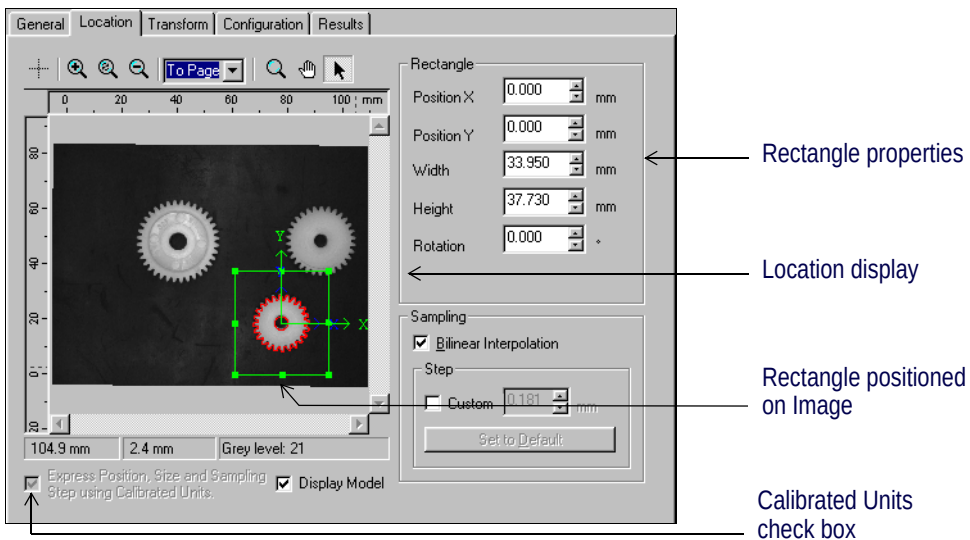


Figure 399 Image Sharpness Tool — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.

- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

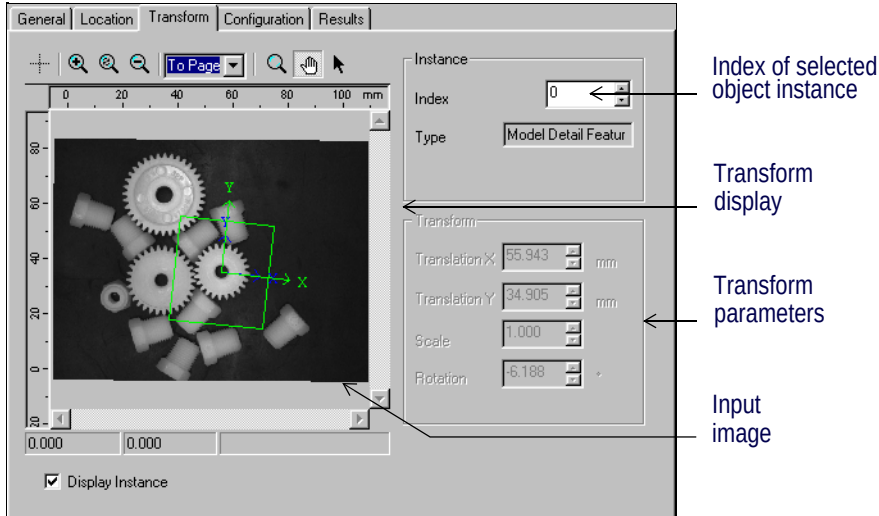


Figure 400 Image Sharpness Tool — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box.
- Alternately, you can use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure the parameters for determining image sharpness in the Rectangle region of interest.

The screenshot shows the 'Configuration' tab of the Image Sharpness Tool. The 'Measurement Points' section includes a 'Candidate Points Count' spinner set to 7, a 'Standard Deviation Threshold' spinner set to 20.000, and a checked 'Automatic' checkbox. The 'Sharpness Operator' section includes a 'Kernel Size' spinner set to 5.

Figure 401 Image Sharpness Tool — Configuration Panel

Measurement Points

Measurement Points are the points used to calculate the average Sharpness result. Only the **Candidate Points** that meet the **Standard Deviation Threshold** are retained as measurement points.

Candidate Points Count

Sets the maximum number of points that can be used by the tool to calculate the image sharpness.

- When **Automatic** is enabled, the **Candidate Points Count** value is determined and set by the tool, according to the dimension of the tool's region of interest. This is the recommended setting.
- When **Automatic** is disabled, you must manually provide a value **Candidate Points Count**.

Standard Deviation Threshold

This threshold sets the minimum allowable Standard Deviation for a candidate points to be accepted as a measurement point.

Sharpness Operator

The sharpness process operation uses an operator defined by the **Kernel Size** parameter. The default setting of 5 (a 5X5 kernel) is generally sufficient for most cases.

Results Panel

The Results panel shows the results of the Image Sharpness Tool process, including a visual display of the sampled Image.

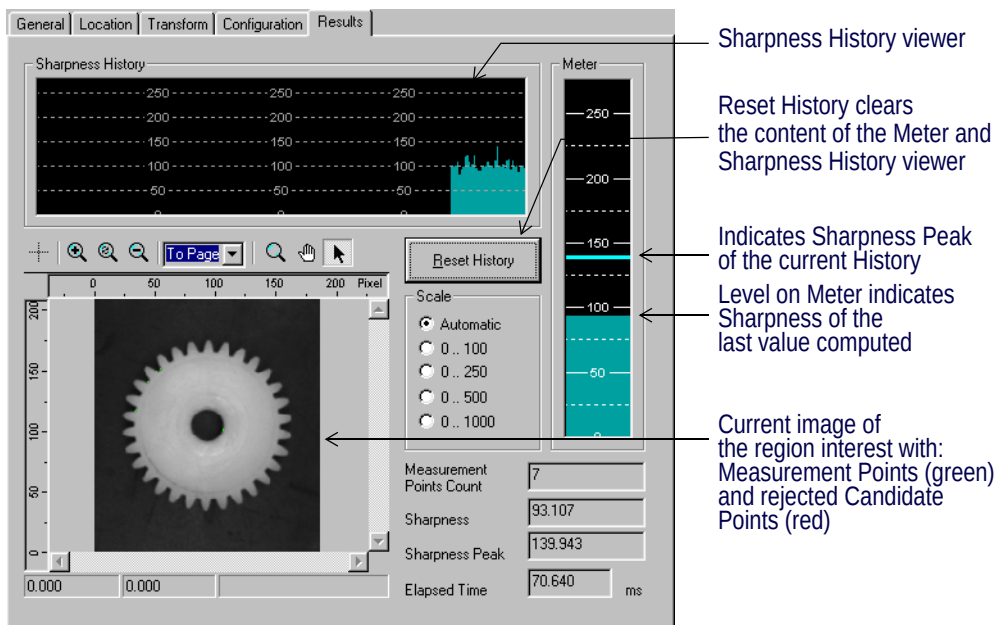


Figure 402 Image Sharpness Tool — Results Panel

Sharpness History

The Sharpness History viewer represents the history of sharpness results for prior executions of the tool since the last reset. The scale lines in the display depend on the Scale setting.

Meter

The Meter provides a gauge-type display of the current average **Sharpness** value.

The light blue bar indicates the average **Sharpness** value computed by the tool since the history was reset.

Reset History

Use this button to reset the sharpness history and clear all information from both the **Meter** and the **Sharpness History** displays.

Scale

Scale sets the range of values that can be displayed in the **Meter** display and the **Sharpness History** viewer.

- When **Automatic** is enabled, the tool automatically sets the scale.
- With a custom **Scale** range, all results are clipped back to the maximum scale value. For example, with a scale of 0-100, all Sharpness results over 100 will be displayed as a value of 100.

Results Display

The image display shows the last processed region of interest and also displays the following:

- Green points indicate the **Measurement Points** used to calculate the average sharpness.
- Red points indicate **Candidate Points** that were rejected because their local standard deviation value was lower than the **Standard Deviation Threshold**.

Results

The results provided by the Image Sharpness Tool are the following:

- **Measurement Points Count** is the number of points actually used to measure the average sharpness for the current region of interest. This can be less than the number of **Candidate Points** set in the Configuration panel

- **Sharpness** is the average sharpness value calculated for the current instance. The sharpness value ranges from a maximum of 1000, indicating a very sharp image, to 0 indicating a very blurry image.
- **Sharpness Peak** is the maximum average sharpness value computed by the tool since the history was reset.
- **Elapsed Time** shows the total execution time of the Image Sharpness Tool.

Sampling Tool

The Sampling Tool creates an image sampled from a region of interest defined on a larger Image. Like other HexSight inspection tools, the Sampling Tool can be configured to use Model-based, as well as Image-based positioning. A typical use of the Sampling Tool is the implementation of custom Model-based inspection processes by creating Model-based sampled images that can be accessed by custom C++ algorithms through the HexSight Class Library. The Sampling Tool can also be used to implement hierarchical searches by first locating a large object with a HexSight Locator and then searching a Model-based sampled image with another HexSight Locator for smaller features on that object.

Sampling Tool Chapter Outline

[Understanding Sampling Tool Concepts](#)

[Sampling Tool Basics](#)

[Sampling Tool Results](#)

[Using the Sampling Tool](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Results Panel](#)

Understanding Sampling Tool Concepts

Sampling Tool Basics

Region of Interest

In the context of the HexSight Sampling Tool, sampling refers to the procedure of creating a sub-image from a region of interest defined on a larger Image. The region of interest is bounded by a Rectangle that can be freely rotated and translated.

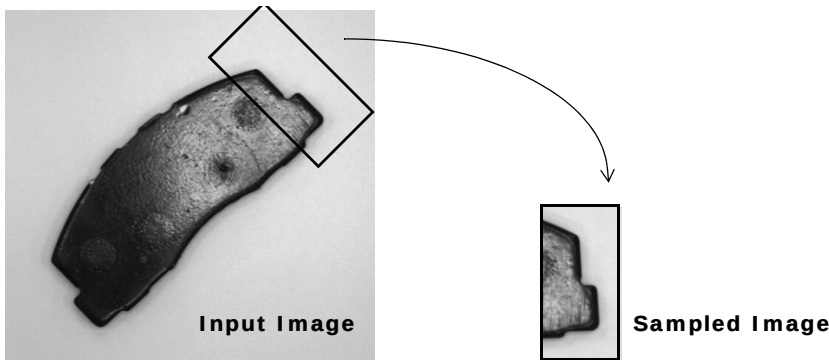


Figure 403 Sampling Tool — Region of Interest

Interpolated Sampling

The sampling process scans each pixel position in the sampled Image and applies the inverse transform of the Rectangle to determine where this sampled pixel lands in the original Image.

In its simplest form, the sampling process takes the value of the closest pixel in the input Image and assigns it directly to the sampled pixel. The pixel grid of the sampled Image is rarely aligned with that of the input Image and uninterpolated sampling can cause imprecisions characterized by jaggedness.

Bilinear interpolated sampling is recommended for applications requiring subpixel accuracy. With this method, instead of assigning to the sampled pixel the value of the closest pixel in the input image, its value is computed by interpolating the value of the four closest pixels in the input image. Although

slower than uninterpolated sampling, bilinear interpolated sampling is recommended for most applications.



Uninterpolated sampling



Bilinear interpolated sampling

Figure 404 Sampling Tool — Bilinear Interpolation

Sampling Tool Results

Calibration Coordinate System

Pixels created by the Sampling Tool in the sampled Image are all the same size and are square. Also, this Image is calibrated by the Sampling Tool using either the World, Object, Tool or Image coordinate system. The calibration coordinate system does not affect how the sampled Image is created. It only defines the calibrated position of each pixel in the sampled Image.

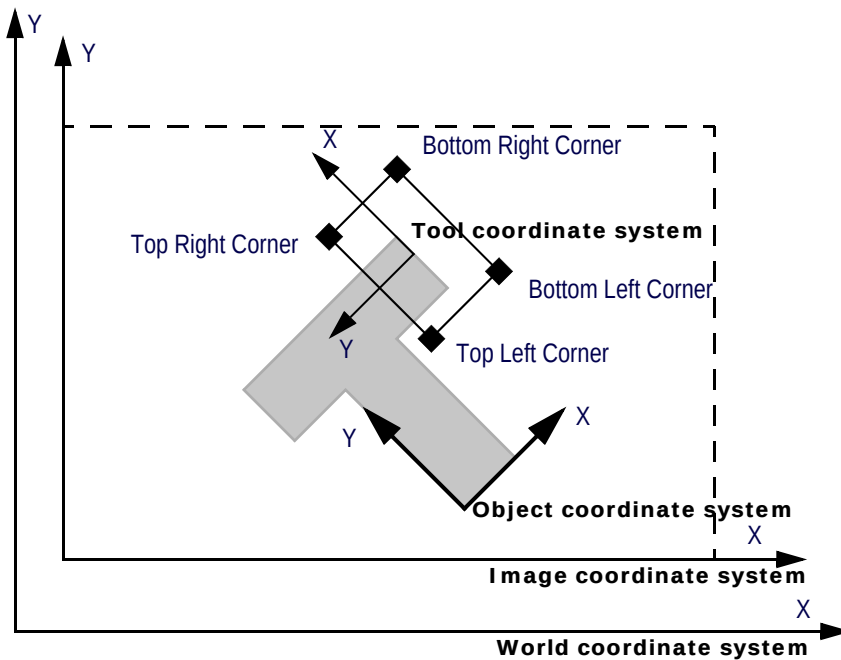


Figure 405 Sampling Tool — Calibration Coordinate Systems

Sampled Image Properties

In addition to the uncalibrated width and height of the sampled Image, expressed in pixels, the Sampling Tool computes the following calibrated properties: pixel width, pixel height, Image width, Image height and the position of the four corners of the Image bounding box in the calibration coordinate system.

Uncalibrated width and height respectively represent the number of pixels in a row and a column of the sampled Image. Since pixels are square in the sampled Image, pixel width and height are always the same. Calibrated width and height are respectively obtained by multiplying uncalibrated width by pixel width and the uncalibrated height by pixel height.

The position of the four corners of the sampled Image's bounding box are always expressed in calibrated units. These corners are always referred to as: bottom left, bottom right, top left and top right, with respect to their position

in the Tool coordinate system. So, since the tool can be rotated with respect to the selected calibration coordinate system, the bottom left corner, for example, is not necessarily the point with the leftmost and bottommost calibrated coordinates. See [Figure 405](#) for an example.

Using the position of the four corners of the bounding box, along with the calibrated pixel width and height, it is possible to compute the calibrated position of any point in the sampled Image with respect to the selected calibration coordinate system. This can be useful when implementing a custom Model-based inspection algorithm in C++ using images produced by the Sampling Tool.

Using the Sampling Tool

Setup of the Sampling Tool is carried out on four tab-selected panels: General, Location, Transform, and Results. Steps for using and setting up the Sampling Tool will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Sampling Tool and enter the tool's input and output Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

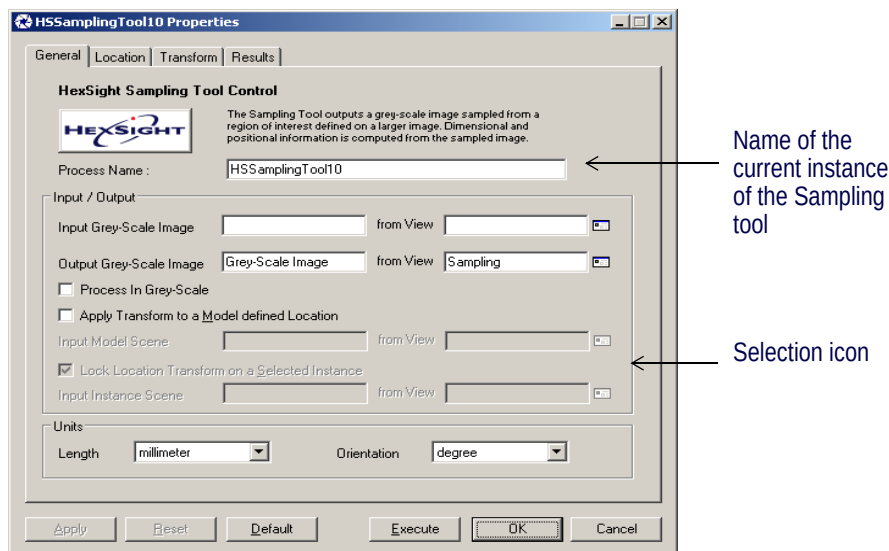


Figure 406 Sampling Tool — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 42 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 42 Inputs and Settings Required for Different Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model-Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSSamplingTool#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Output Image

Output Image is the Image that will be created in the runtime database by the tool. Since this is an output Image, it will overwrite the selected Image in the runtime database when the tool is executed.

- Type the name of the Image in the **Output Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Process In Grey-Scale

When selected, forces a color image to be processed as a grey-scale image. Selecting this mode can decrease processing time of color image, when appropriate.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.

- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Locator. This Scene is provided by a HexSight Locator configured to output its Instance Scene in the runtime database. See the **Locator** chapter of this guide for more details.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

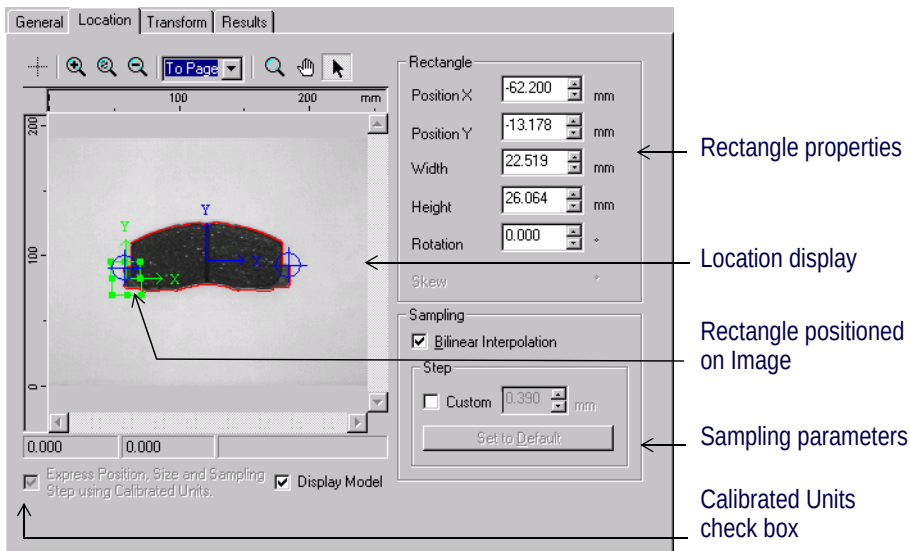


Figure 407 Sampling Tool — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.

- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.



Figure 408 Sampling Tool — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest. When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Results Panel

The Results panel shows the results of the Sampling Tool process, including a visual display of the sampled Image. Read only text boxes provide dimensional and positional informations on this Image.

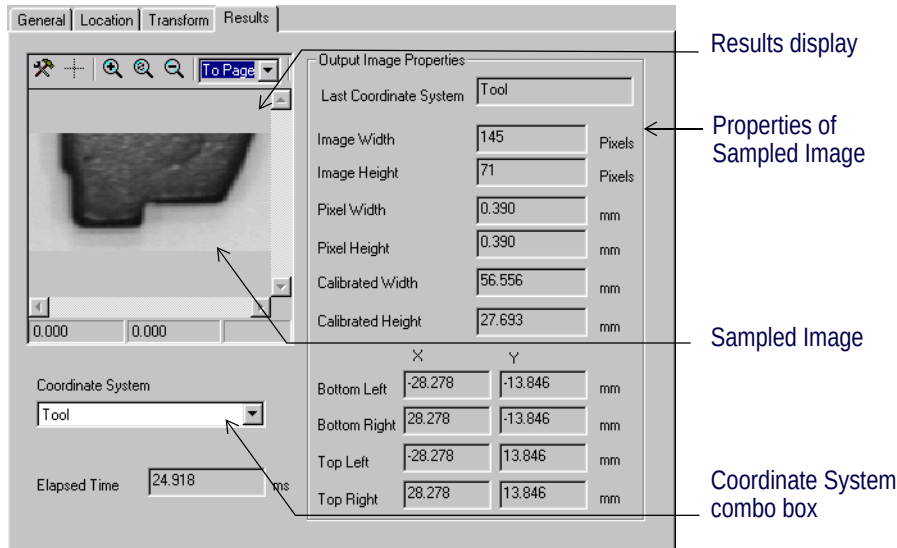


Figure 409 Sampling Tool — Results Panel

Results Display

The **Results Display** shows the sampled Image. Use the Calibrate button of the **Results Display** to toggle between calibrated and uncalibrated display modes.

Coordinate System

Use the **Coordinate System** combo box to define the coordinate system used to calibrate the sampled Image. When a new coordinate system is selected, it will only be applied to the sampled Image at the next execution of the tool.

- The coordinate system does not affect the size of the sampled Image nor its pixel values; only the calibration information of the sampled

Image is dependent on this setting. Whatever coordinate you select, the sampled Image always appears the same when displayed in the uncalibrated mode.

- When displayed in the calibrated mode, the sampled Image may appear rotated, depending on the selected coordinate system.

Elapsed Time

Elapsed Time shows the total execution time of the Sampling Tool.

Output Image Properties

The **Output Image Properties** frame displays the read only properties of the Image output by the sampling tool. These properties are described below.

- **Last Coordinate System** shows the coordinate system used to calibrate the sampled Image the last time the tool was executed.
- **Image Width** shows the uncalibrated width of the sampled Image; i.e. the number of pixels in each row of the Image.
- **Image Height** shows the uncalibrated height of the sampled Image; i.e. the number of pixels in each column of the Image.
- **Pixel Width** shows the calibrated width of a single pixel in the sampled Image. Pixels in the sampled image are square, thus pixel width is always the same as pixel height. Pixel width is expressed in the **Length Units** selected in the General panel.
- **Pixel Height** shows the calibrated height of a single pixel in the sampled Image. Pixels in the sampled image are square, thus pixel height is always the same as pixel width. Pixel height is expressed in the **Length Units** selected in the General panel.
- **Calibrated Width** shows the total calibrated width of the sampled Image. Calibrated width is expressed in the units selected with the **Length Units** combo box in the General panel.

- **Calibrated Height** shows the total calibrated height of the sampled Image. Calibrated height is expressed in the **Length Units** selected in the General panel.
- **Bottom Left** shows the calibrated X-Y coordinates of the bottom left corner of the sampled Image's bounding box with respect to the selected coordinate system. X-Y coordinates are expressed in the **Length Units** selected in the General panel
- **Bottom Right** shows the calibrated X-Y coordinates of the bottom right corner of the sampled Image's bounding box with respect to the selected coordinate system. X-Y coordinates are expressed in the **Length Units** selected in the General panel.
- **Top Left** shows the calibrated X-Y coordinates of the top left corner of the sampled Image's bounding box with respect to the selected coordinate system. X-Y coordinates are expressed in the **Length Units** selected in the General panel.
- **Top Right** shows the calibrated X-Y coordinates of the top right corner of the sampled Image's bounding box with respect to the selected coordinate system. X-Y coordinates are expressed in the **Length Units** selected in the General panel.

Evaluator Tool

The Evaluator tool converts a specific result from any other tool as a Pass or Fail value, in order to trigger an event with an upcoming OutputTool, depending on the decision result. Comparisons and mathematical operations are available. Multiple Evaluator tools can be used to obtain a final decision.

Note that a result of **True** is required to schedule an event in the OutputTool. If a trigger should be done on a **False** result, modify the test or use an additional Evaluator Tool to negate the value using the **NOT** operator. Additionally, if a result is 0, it is considered **False**, while a result different than 0 is considered **True**.

Evaluator Tool Chapter Outline

[Using the Evaluator Tool](#)

[General Panel](#)

[Parameters Panel](#)

[Results Panel](#)

Using the Evaluator Tool

The Evaluator Tool is composed of three tab-selected panels: General, Parameters, and Results.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the tool.

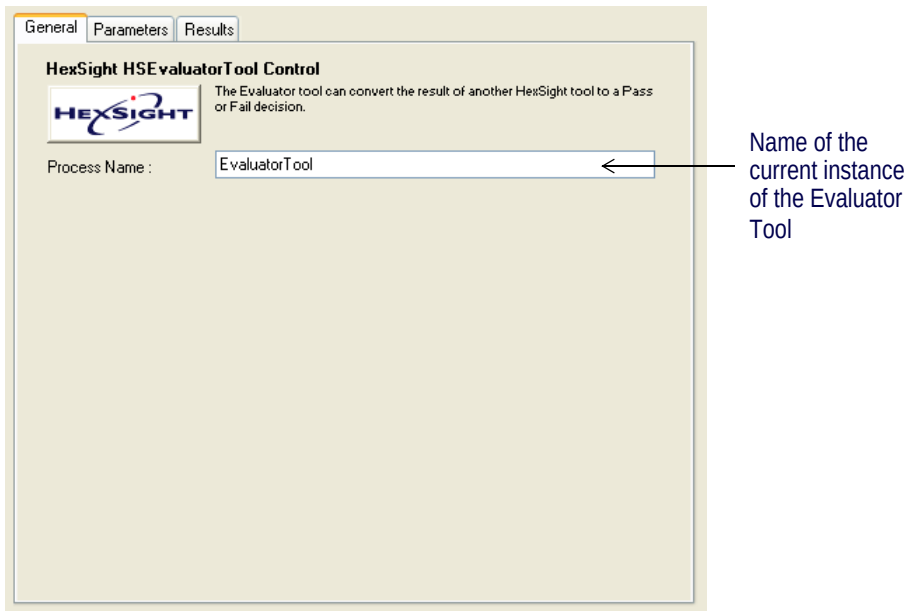


Figure 410 EvaluatorTool — General Panel

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSEvaluatorTool#, where # is replaced by its rank number in the list of instantiated processes.

Parameters Panel

This panel is divided in three sections: the left operand, the operation and the right operand. The operands can either be a constant value or the output of any other tool. The available tools will appear in the combo box, selection will update the available results to analyze. Comparison operators can be used to compare both operands. If a mathematical operation is used instead, the value computed can also be used as input in a subsequent Evaluator Tool. This features allows for more complex equations than just a binary operation.

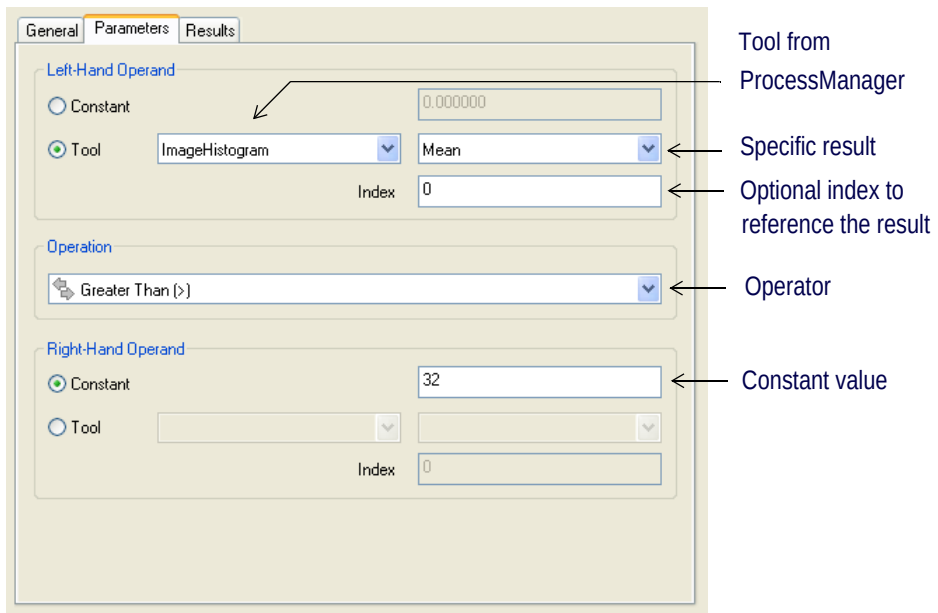


Figure 411 Evaluator Tool — Parameters Panel

Operations

Multiple operations are available. They can either return a Boolean value to be used as a Pass or Fail result, such as comparison operators (Greater than, Less than, etc), or return a value that can be used as input in other Evaluator Tools (Addition, Division, etc). Most of these operations are binary, which require two operands, and some are unary, such as the NOT operator, that only acts on the Left operand.

Results Panel

The Results panel shows the results of the Evaluator process.

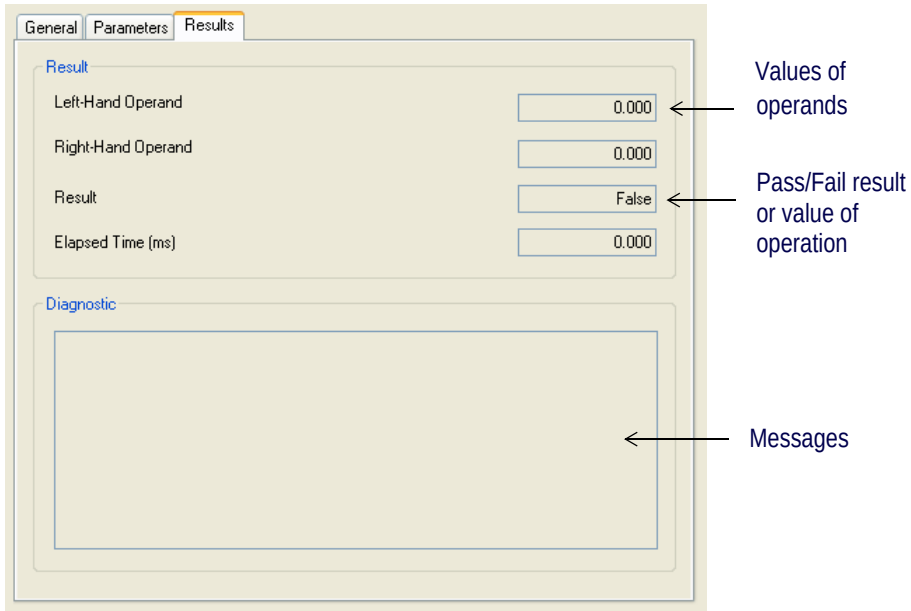


Figure 412 Evaluator Tool — Results Panel

Left-Hand Operand

This is the result of the evaluation of the top operand in the Parameters panel.

Right-Hand Operand

This is the result of the evaluation of the bottom operand in the Parameters panel.

Result

Result is the computed value of the evaluation. It can be Boolean or Float, depending on the operation executed. As mentioned, a value of 0 is also considered **False (Fail)**, and a value different than 0 is **True (Pass)**.

The image shows a screenshot of the Evaluator Tool interface. It has a light beige background with a blue header bar. Below the header, there is a section titled 'Result' in blue. This section contains four rows, each with a label on the left and a text input field on the right. The labels are 'Left-Hand Operand', 'Right-Hand Operand', 'Result', and 'Elapsed Time (ms)'. The input fields contain the values '1.000', '1.000', '2.000', and '0.264' respectively. An arrow points from the text 'Value of operation' to the 'Result' input field.

Result	
Left-Hand Operand	1.000
Right-Hand Operand	1.000
Result	2.000
Elapsed Time (ms)	0.264

Value of operation

Figure 413 Evaluator Tool — Operation resulting in a value

Elapsed Time

Elapsed Time shows the total execution time of the Evaluator process.

Diagnostic

When the Evaluator Tool encounters an issue, a message may appear to help the user solve the problem.

Output Tool

The Output Tool can schedule a digital output of a Gocator based on the result of another HexSight tool, at a specific moment in time. The decision to schedule the digital output is usually the result of an Evaluator Tool. Only a **True** result (or a value different than zero) will schedule the event. To obtain the proper decision, it is possible to use multiple Evaluator Tool for more complex operations (see [Evaluator Tool](#)).

Output Tool Chapter Outline

[Using the Output Tool](#)

[General Panel](#)

[Parameters Panel](#)

[Results Panel](#)

Using the Output Tool

The Output Tool is composed of three tab-selected panels: General, Parameters, and Results.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the tool and select a reference image.

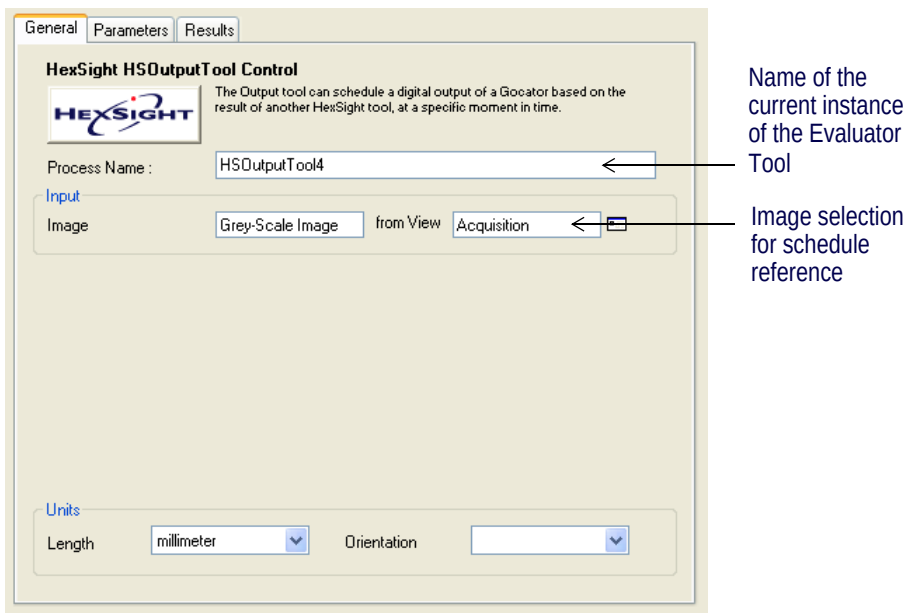


Figure 414 Output Tool — General Panel

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSDOutputTool#, where # is replaced by its rank number in the list of instantiated processes.

Input image

The Input image is the image from the runtime database used for schedule reference. Every captured image contains a Time Stamp and an Encoder Position (if available at the time of the acquisition). These reference values are used to compute the schedule time or position to trigger an event on the device.

Parameters Panel

This panel is divided in three sections: the schedule condition, the schedule mode and the device parameters.

The screenshot shows the 'Parameters' tab of the 'Output Tool' interface. It is divided into three sections:

- Schedule Condition:** Contains three fields: 'Tool' (dropdown menu set to 'EvaluatorTool'), 'Function' (dropdown menu set to 'Result'), and 'Index' (text input set to '0').
- Schedule Mode:** Contains one field: 'Encoder or Time offset' (text input set to '0' with a unit of μs).
- Gocator:** Contains two fields: 'Gocator IP' (a 'Discover' button followed by a text input showing '192 . 168 . 1 . 10') and 'Digital Output' (a dropdown menu set to '0').

Annotations on the right side of the image point to specific fields:

- 'Tool from ProcessManager' points to the 'Tool' dropdown.
- 'Specific result' points to the 'Function' dropdown.
- 'Optional index to reference the result' points to the 'Index' text input.
- 'Constant offset value, in ticks or in μs .' points to the 'Encoder or Time offset' text input.
- 'Gocator IP address' points to the 'Gocator IP' text input.
- 'Digital output index' points to the 'Digital Output' dropdown.

Figure 415 Output Tool — Parameters Panel

Schedule Condition

The schedule condition is the output of another tool, usually an Evaluator Tool. The schedule will occur if and only if the result of this function is **True** or different than zero.

Schedule Mode

The schedule mode defines the offset from the acquisition stamp in the currently set domain. Note that a running Gocator cannot change its domain, hence it is necessary to set the domain accordingly prior to starting the device. The offset is expressed in millimeters if the domain is in encoder value, otherwise it should be in microseconds for a time domain.

Gocator

The Gocator device parameters are set in the last section. The IP address is required, along with the index of the digital output to trigger. The Discovery button can locate Gocators on the subnet, for easier setup.

Results Panel

The Results panel shows the results of the Output process.

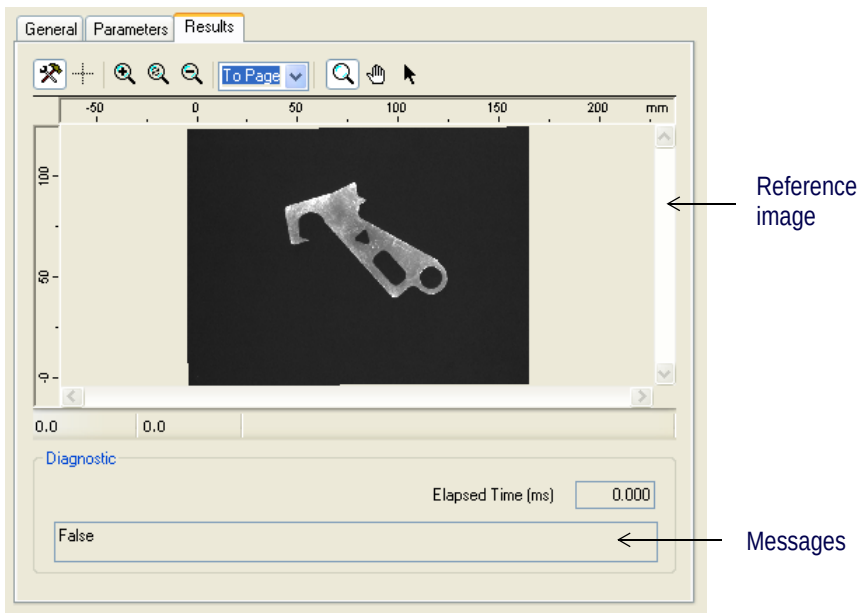


Figure 416 Evaluator Tool — Results Panel

Elapsed Time

Elapsed Time shows the total execution time of the Output process.

Diagnostic

Result of the last schedule command. A positive response is shown if the trigger event was properly scheduled. If the Output Tool encountered an issue during the schedule command, a message may appear to help the user solve the problem. Usual problems are: Gocator unreachable, Gocator not running, Schedule condition not fulfilled, or Offset too small resulting in a schedule in the past.

Barcode Reader

The Barcode Reader reads and extracts information from 1D symbologies commonly known as bar codes.

Barcode Reader Chapter Outline

[Understanding Barcode Reader Concepts](#)

[Region of Interest](#)

[General Properties](#)

[Detection Parameters](#)

[Learn Process](#)

[Reading Multiple Bar Codes](#)

[Using the Barcode Reader](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Advanced Panel](#)

[Results Panel](#)

Understanding Barcode Reader Concepts

This section provides the basic concepts of the Barcode Reader process.

Region of Interest

The tool's Rectangle is the region of interest within which the process finds and reads bar code instances. The positioning of the tool Rectangle is presented in detail in the Symbology Tools chapter, under Tool Positioning.

- Restricting the size of the Rectangle will generally decrease the execution time of the tool.
- The Rectangle region of interest must exceed the bar code by 25 pixels or more on all sides. Setting the Rectangle sides within this 25-pixel limit may cause both the learning and decoding to fail.
- Up to five bar code instances can be found and decoded within the region of interest. For more details on this subject see the **Reading Multiple Bar Codes** section in this chapter.
- The Locator tool can be advantageously used to locate an object upon which you need to read a bar code. However, if input images contain only one bar code it may be more time efficient to locate the symbology with the Barcode Reader, without using a Locator tool.

General Properties

Barcode properties can be set manually, however these properties can also be auto-detected and set by a successful Learn process (see the **Learn Process** section). Figure 417 illustrates the elements of a bar code symbology.

Code Height and Length

The maximum and minimum height and length of the bar code are always expressed in pixels. These properties are auto-configured after a successful Learn.



Figure 417 Elements of a Bar Code Symbology

Number and Width of Bars

The maximum and minimum number of bars in a bar code, as well as the maximum and minimum width of the bars, are auto-configured after a successful **Learn**.

Detection Parameters

Besides the properties inherent to the bar code itself, the bar code reading process is also affected by the following detection parameters. Some of these parameters may be auto-detected and set by the Learn process. The Learn process may also reset some parameters to their default values.

Probe Spacing

Probe spacing describes physical intervals at which the search for a bar code is carried out. Probing is carried out either horizontally, vertically, or both depending on the selected **Search Direction**. Vertical and horizontal probe spacing is configured by a successful Learn.

Search Direction

By default, the tool applies the default **Horizontal->Vertical** search direction. In this case, the tool first searches horizontally for symbology instances then, if none is found, it searches vertically. Selecting **Vertical->Horizontal** causes the opposite search process. The search direction can

also be set to only **Vertical** or **Horizontal**, which typically provides a faster execution time than a dual-direction search.

Minimum Quiet Zone

The quiet zone is a clear area found to either side of a bar code. The minimum required size, in pixels, of the quiet zone varies according to the bar code type.

- If any of the spaces between two adjacent bars in the bar code exceeds the **Minimum Quiet Zone** length, the Bar Code Reader may interpret this space as a quiet zone and therefore stop the reading process. In such cases, the **Minimum Quiet Zone** should be increased.
- The default minimum bar code quiet zone length is 12 pixels. If for any reason the quiet zone in the image is less (because of camera setup restrictions, for example), the **Minimum Quiet Zone** can be lowered in order for the algorithm to accept shorter space as quiet zone.
- Since the Learn process uses the default Quiet Zone value, changing this value may cause the Learn to fail. If a value other than the default value is required for runtime, change this value AFTER executing a Learn.

Minimum Edge Strength

This parameter sets the minimum greylevel value required for detection of edges within the bar code symbology.

- For low contrast bar codes, especially BC412 symbologies, the edge strength value can be set between 5 and 18.
- The edge strength can be raised to a higher value when the bar code has very high contrast.
- With higher edge strength, the program will ignore most of low contrast objects and get to the bar code area quicker.
- If the bar code's contrast is stable and unlikely to change, a value of 36 should provide optimum run performance.

- Since the Learn process uses the default Edge Strength value, changing this value may cause the Learn to fail. If a value other than the default value is required for runtime, change this value AFTER executing a Learn.

Barcode Types

Most bar code types can be auto-detected and enabled by a successful Learn. However, BC412, Postnet, or Pharmacode symbologies are not detected and must be enabled *before* performing a Learn.

The Barcode Reader supports the following bar code types:

- Code 128
- UPC-E
- Code 39
- UPC Supplemental
- Code 93
- PDF-417
- Interleaved 2 of 5
- BC 412
- Codabar
- Postnet
- UPC/EAN
- Pharmacode

Barcode-Specific Properties

Specific properties exist for BC412 bar codes. Enabling any of these properties will not have any effect on other types bar code except BC 412:

- **Checksum Enabled:** When enabled, a BC412 bar code must contain a checksum character to be correctly decoded. When disabled, BC412 bar codes without a checksum character can also be decoded.
- **Decode Left To Right:** When enabled, a BC412 bar code is decoded from left to right. When disabled, a BC412 bar code is decoded from right to left. This mode is only useful when **Start/Stop Enabled** is disabled. This mode should not be used for SEMI BC412 bar codes that contain both a Start/Stop pattern and a checksum character.
- **Start/Stop Enabled:** When enabled, a BC412 bar code must contain a start/stop pattern to be correctly decoded. When disabled, BC412 bar codes without a a start/stop pattern can also be decoded.
- **Polarity:** This sets the polarity of the bar code with respect to its background to either **Light On Dark** or **Dark on Light**.

Learn Process

All parameters for bar code detection and reading can be can be set by the user, either programmatically or through the interface. To assist the user in configuring parameters, the Barcode Reader provides a Learn process that automatically detects and sets the parameters required to process a given symbology.

When the **Learn** process is executed, the Data Matrix Reader attempts to locate and decode a symbology in the current input image.

- The Learn should be carried out on a bar code image that is typical of the bar code instances the tool will be required to decode in runtime.
- A successful Learn (**Learn Succeeded**) optimizes and sets the following parameters: Horizontal Probe Spacing, Vertical Probe Spacing, Maximum Bar Width, Minimum Bar Width, Maximum Code Height, Minimum Code Height, Maximum Code Length, Minimum Code Length, and Maximum Number of Bars.
- An unsuccessful Learn (**Learn Failed**) indicates image quality is inadequate, that parameters need to be changed or both.

- You can only carry out a Learn for one bar code instance at a time. Each time you carry out a new Learn, parameters, including the bar code type, are reset accordingly.
- If you know the parameters that are required, you can set them programmatically, or through the interface without carrying out a Learn process.
- A Learn detects the bar code type. However, the Learn process will not automatically detect BC412, Postnet, or Pharmacode symbologies. For these types of bar code, you must set the bar code type before executing a Learn.

Preprocessing and Image Quality

Good image quality is significant factor in the successful learning and reading bar codes. Since ideal conditions are not always possible, the Barcode Reader can apply one of several image-processing algorithms prior to the decoding process. The selected preprocessing algorithm is also applied prior to a Learn. See the following sections for more details on image preprocessing: [Preprocessing](#) in the [Using the Barcode Reader](#) section and the ProcessingAlgorithm property in the **Reference Manual**.

Reading Multiple Bar Codes

A single Barcode Reader tool can process up to five bar code instances in each input image. These instances can be of different types, providing that the configured parameters allow the decoding of the different types.

Multiple Barcode Reader Tools

If you need to process more than one *type* of bar code in an application, we recommend that you instantiate one Barcode Reader tool for each type of bar code you need to read. This allows you to set (manually or through a Learn) the optimal detection parameters for each bar code type.

Single Barcode Reader Tool

If you use only one Barcode Reader tool to process more than one type of bar code, you will have to set all parameters "broadly" enough to deal with the range of bar code types covered by your application. To do this, learn each bar code separately and record its bar code properties. Next, set parameters (manually or programmatically) so that they take into account the properties learned from each bar code.

Using the Barcode Reader

Setup of the Barcode Reader is carried out on four tab-selected panels: General, Location, Transform, and Results. Steps for using and setting up the Barcode Reader will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Barcode Reader and enter the tool's input and output Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

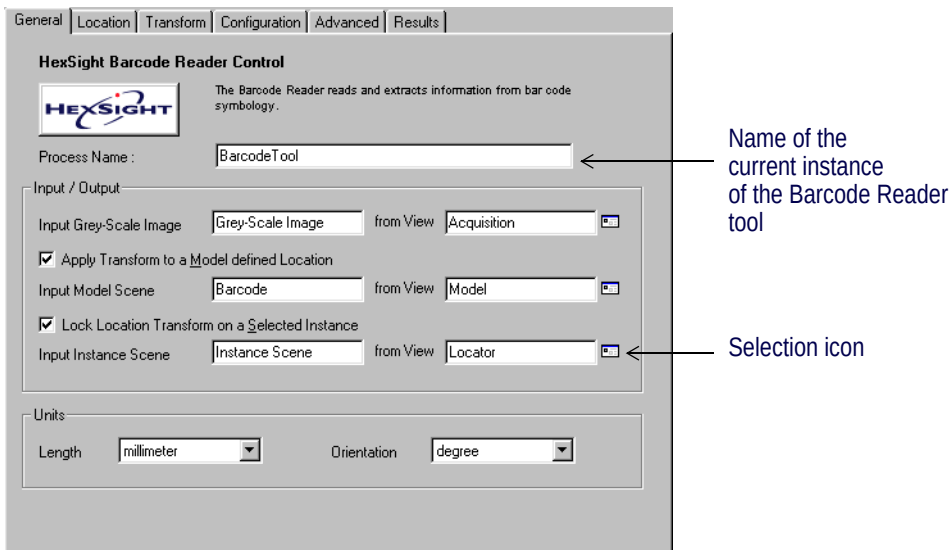


Figure 418 Barcode Reader — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the *Inspection Tools* chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes

Table 43 Requirements for Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 43 Requirements for Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSBarcodeReader#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Locator. This Scene is provided by a HexSight Locator configured to output its

Instance Scene in the runtime database. See the **Locator** chapter of this guide for more details.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note:Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

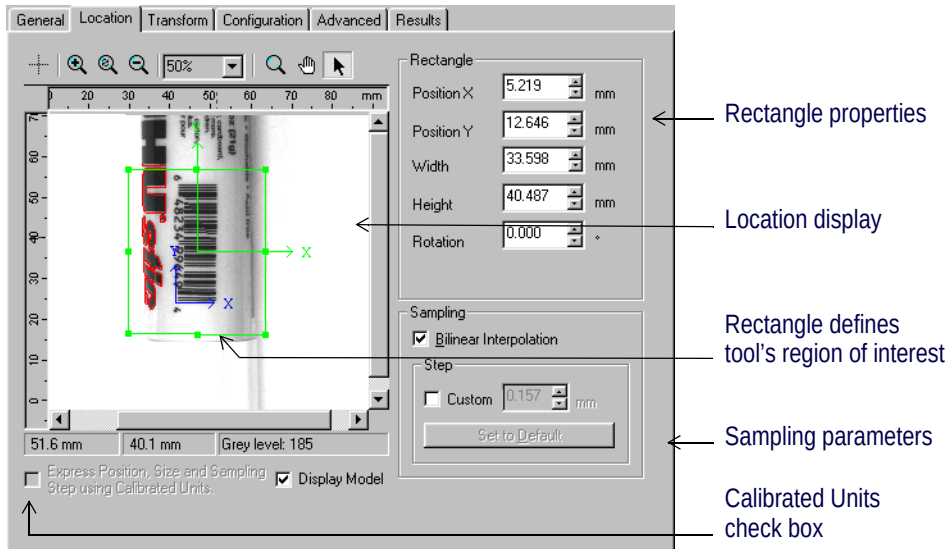


Figure 419 Barcode Reader — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is ignored when the tool is Imaged-based.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

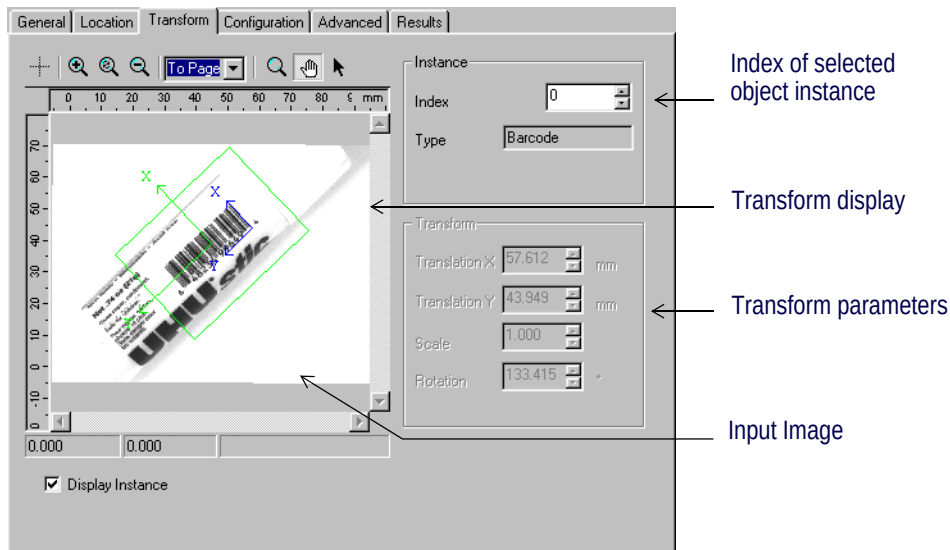


Figure 420 Barcode Reader — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected

since the instance processed does not match the type of object for which the tool is configured.

- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest. When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure general bar code parameters either manually or automatically, through the Learning process. This panel also provides an image processing function.

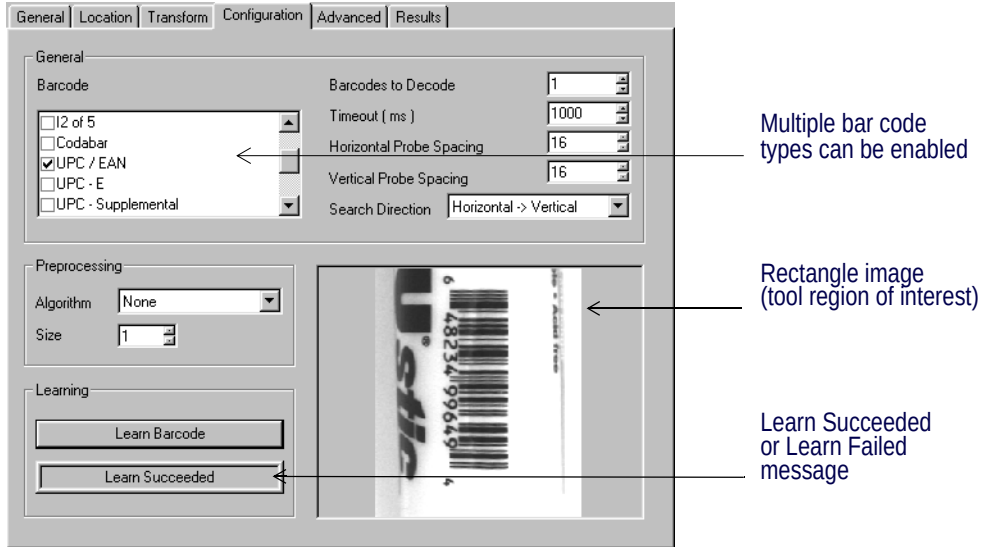


Figure 421 Barcode Reader — Configuration Panel

General Properties

General properties can be automatically set by carrying out a successful Learn, using the **Learn Barcode** command. This is recommended if you do not know beforehand the parameters required for locating and decoding the bar codes in the current application

Barcode Type

Most bar code types are automatically detected when you execute **Learn Barcode**, whether or not a **Barcode** type check box has been enabled beforehand. However, to read **BC412**, **Postnet**, or **Pharmacode** symbologies, you must enable the appropriate check box BEFORE executing a Learn.

Barcodes to Decode

This parameter sets the maximum number of bar code instances that can be decoded a single input image. The maximum value is 5. For optimum performance, restrict this value to the minimum number of bar codes

instances you need to decode in an image; typically this will be a single instance.

Timeout

Timeout sets the time limit after which the Barcode Reader aborts its process.

Horizontal and Vertical Probe Spacing

The Barcode Reader processes (probes) image pixels horizontally and vertically, at regular the intervals that are defined by the **Spacing** parameter. These two parameters are auto-configured after a successful Learn

- Lower values incur longer execution times but provide a more thorough processing.
- Higher values can speed up execution but result in missed bar codes.

Search Direction

Use this parameter to set the direction of the search.

- **Horizontal -> Vertical**: Bar codes are scanned horizontally then vertically.
- **Vertical -> Horizontal**: Bar codes are scanned vertically then horizontally.
- **Horizontal**: Bar codes are scanned horizontally only.
- **Vertical**: Bar codes are scanned vertically only.

Preprocessing

For poor symbology images, a properly chosen preprocessing step may enhance the image quality and reduce the noise to improve the reading robustness. Preprocessing parameters are applied to the image before any other operations, including **Learn Barcode**.

By default, no preprocessing **Algorithm** is applied (**None**). The **Size** text box sets the parameter for the selected preprocessing algorithm where applicable. Available preprocessing options are:

- **Invert**: Reverses the polarity of the image. For example, changing a light on dark bar code to a dark on light bar code. The **Size** parameter is ignored.
- **Erode**: This option enables you to increase dark areas (of a dark on light bar code) The **Size** parameter sets the value of the neighborhood applied by the operation.
- **Dilate**: This option enables you to increase light areas (of a light on dark bar code). The **Size** parameter sets the value of the neighborhood applied by the operation.
- **Open**: This option enables you to remove minor light defects of dark areas. The **Size** parameter sets the value of the neighborhood applied by the operation.
- **Close**: This option enables you to remove minor dark defects of light areas. The **Size** parameter sets the value of the neighborhood applied by the operation.
- **Gaussian**: The Gaussian operation acts like a low pass filter and helps remove noise and has overall blurring effect on the image. The **Size** parameter sets the value of the kernel applied by the operation.
- **Sharpen**: This has the effect of sharpening the image, especially edges. The **Size** parameter sets the value of the kernel applied by the operation.
- **Sharpen Low**: This has the effect of sharpening and smoothing the image at the same time. The **Size** parameter sets the value of the kernel applied by the operation.
- **Median**: The Median operation reduces impulsive image noise without degrading edges or smudging intensity gradients. The **Size** parameter sets the value of the kernel applied by the operation.

- **Histogram Stretching:** This operation increases the contrast in the image. The **Size** parameter is ignored.

Learn Barcode

Learn Barcode provides a method to automatically set the parameters for a specific symbology.

- When the Learn is successful (i.e. **Learn Succeeded** message appears), the Barcode Reader sets the parameters to the values required to read the bar code in the current image.
- When Learn Barcode fails (i.e. **Learn Failed** message appears), you will need to make adjustments before reattempting the Learn. Most often the failure indicates a problem with image quality.
- To read BC412, Postnet, or Pharmacode, you need to check the **Barcode** type in General properties *before* performing **Learn Barcode**.

Advanced Panel

The properties and parameters that can be set in this panel are mostly a function of the intrinsic bar code characteristics.

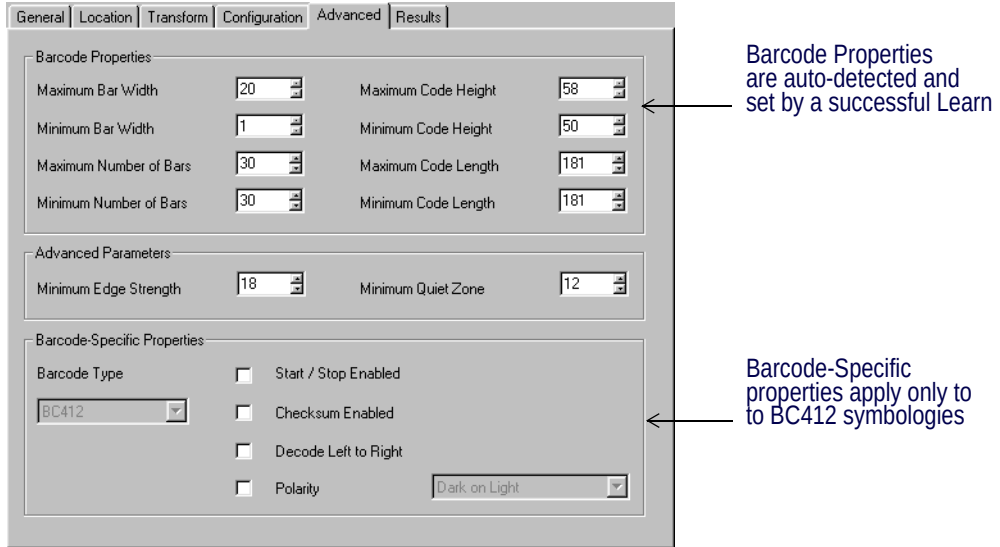


Figure 422 Barcode Reader — Advanced Panel

Barcode Properties

The following properties are auto-configured after a successful Learn. If any of these values may vary in real time images, change the learned value before runtime.

- **Maximum Bar Width:** Maximum width of bars in pixels.
- **Minimum Bar Width:** Minimum width of bars in pixels.
- **Maximum Number of Bars:** Maximum number of bars in the bar codes to be decoded.
- **Minimum Number of Bars:** Minimum number of bars in the bar codes to be decoded.
- **Maximum Code Height:** Maximum height of bar codes to be decoded.
- **Minimum Code Height:** Minimum height of bar codes to be decoded.
- **Maximum Code Length:** Maximum length of bar codes to be decoded.
- **Minimum Code Length:** Minimum length of bar codes to be decoded.

Advanced Parameters

These parameters are not auto-configured after a successful Learn. Since the Learn uses the default values for these parameters, changing their values may cause the Learn to fail. In this happens, reset default values before re-attempting a Learn, then re-adjust these parameter values.

Minimum Edge Strength

This value sets the required greylevel value for the edges in the bar code. The default value is typically sufficient for most cases. The allowable range is from 5 to 100.

- The edge strength can be raised to a higher value when the bar code has very high contrast. With higher edge strength, the Barcode Reader will ignore most of low contrast objects and get to the bar code area quicker.
- If the bar code's contrast is unlikely to change (e.g., when a stable lighting source is employed), the level can be raised to 36 to produce the optimum run performance.

Minimum Quiet Zone

This value sets the minimum length in pixels of the quiet zone, which is the clear border surrounding the symbology.

- The default minimum quiet zone length of 12 pixels is generally sufficient for most cases.
- This length *must* be greater than the space between any two adjacent bars in the bar code.

Barcode-Specific Properties

Barcode-specific properties are currently implemented only for the BC412 symbologies. The status of these properties is ignored for any other type of bar code.

Start / Stop Enabled

This parameter only takes effect when the BC412 barcode contains a start and stop pattern.

Checksum Enabled

This parameter only takes effect when the BC412 barcode contains a checksum character.

Decode Left to Right

This parameter enables left-to-right decoding of the symbology. When disabled, the symbology is decoded from right to left.

Note:Decode Left to Right is applicable only when Start/Stop Enabled is disabled. This mode should not be used for SEMI BC412 bar codes that contain both a Start/Stop and a Checksum.

Polarity

When the polarity is known, select either **Dark on Light**, or **Light on Dark** from the drop-down list. When polarity is not known, disable **Polarity**.

Results Panel

The Results panel shows the results of the Barcode Reader process.

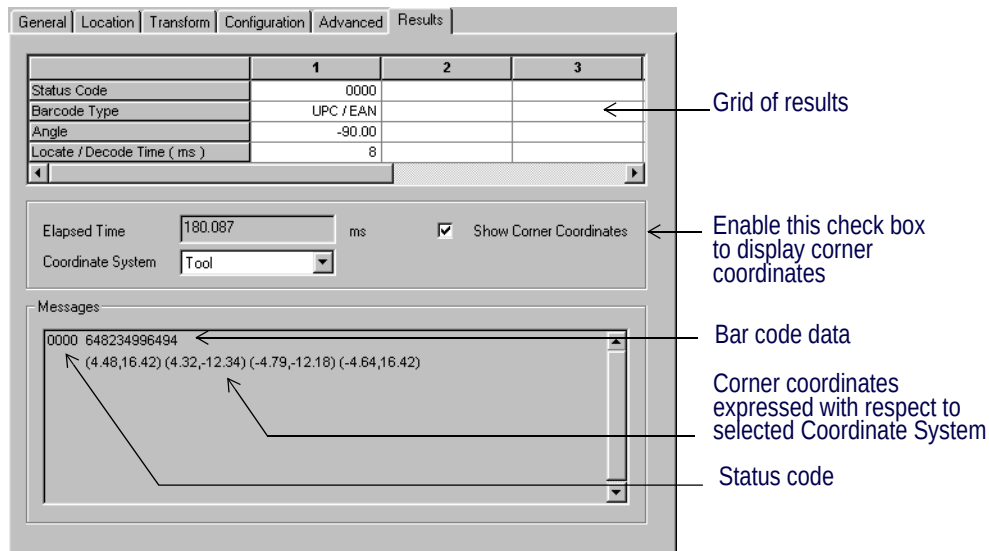


Figure 423 Barcode Reader — Results Panel

Grid of Results

The grid of results displays the following for each bar code instance read by the tool.

- **Status Code:** The following table describes the possible status codes.

Table 44 Status Codes in Results Messages

Numeric Code	Description
0000	Success
1000	No Execute Performed
3048	Symbology Decode Failed
4601	No Edge Candidate Found
4604	First Edge Not Found or Too Small
4605	Second Edge Not Found
4611	Third Edge Not Found
4612	Fourth Edge Not Found

Table 44 Status Codes in Results Messages

Numeric Code	Description
4620	Symbology Corner Out of Image
4621	Size Test Failed
4622	Unable to Find Row/Column
4633	Border Match Failed
4630	Inspection Timed Out
4301	Trigger Overrun
4303	String Mismatch
4304	Angle Match Failed
4305	Excessive Damage
4310	Process Overrun
6001	OCR Conformance Failure
6002	OCR General Error
9999	Dongle Missing

- **Barcode Type:** The name of the bar code type.
- **Angle:** The angle of orientation of a bar code is the counterclockwise angle between the bar code's X-axis and the X-axis of the Rectangle region of interest. See examples in [Figure 424](#).

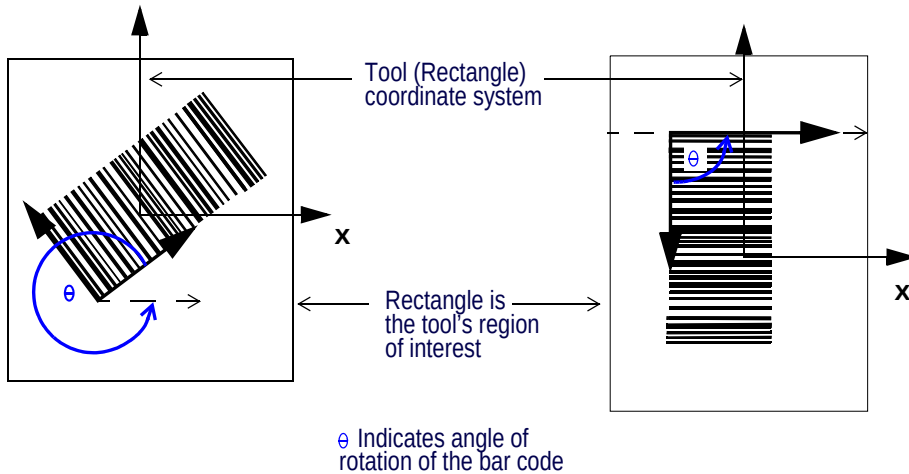


Figure 424 Bar Code Angle of Orientation

- **Locate / Decode Time:** This is the time required to read the given symbology instance, expressed in milliseconds.

Elapsed Time

Elapsed Time shows the total execution time of the Barcode Reader tool.

Coordinate System

Use the **Coordinate System** combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image**, **World**, **Object**, and **Tool**.

Show Corner Coordinates

When **Show Corner Coordinates** is enabled, the **Message** box displays the X-Y coordinates for the four corners of each decoded symbology instance. These coordinates are expressed with respect to the selected **Coordinate System**: **Tool**, **Image**, **Object** or **World**.

- When the World, the Object or the Tool coordinate system is selected, corner coordinates are expressed in the user-selected length units.

- Coordinates are expressed in pixels when the Image coordinate system is selected.
- In the exceptional case where the tool is not configured to work with calibrated units, coordinates in the Tool coordinate system are expressed in pixels but values cannot be returned in the World or in the Object coordinate system.

Messages

The **Message** box provides information for each decoded symbology instance: the Status code, the decoded alphanumeric data, and, when **Show Corner Coordinates** is enabled, the corner coordinates of each decoded bar code.

Data Matrix Reader

The Data Matrix Reader reads and extracts information from 2D symbologies commonly known as data matrixes.

Data Matrix Reader Chapter Outline

[Understanding Data Matrix Reader Concepts](#)

[Region of Interest](#)

[Elements of a Data Matrix](#)

[General Properties](#)

[Advanced Parameters](#)

[Learn Process](#)

[Preprocessing and Image Quality](#)

[Reading Multiple Data Matrixes](#)

[Using the Data Matrix Reader](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Assisted Learn Interface](#)

[Advanced Panel](#)

[Results Panel](#)

Understanding Data Matrix Reader Concepts

This section provides details on the basic concepts of the Data Matrix Reader process. Many of the properties required for detection of a data matrix can be automatically detected and optimized by the integrated Learn function.

Region of Interest

The tool's Rectangle is the region of interest within which the process finds and reads data matrix instances. The positioning of the tool Rectangle is presented in detail in the Symbology Tools chapter, under Tool Positioning.

- Restricting the size of the Rectangle will generally decrease the execution time of the tool.
- The region of interest should at least exceed the matrix by 25 pixels or more on all sides. Setting the Rectangle sides within this 25 pixel limit may cause both the learning and decoding to fail.
- Up to five data matrix instances can be found and decoded within the region of interest. For more details on this subject see the **Reading Multiple Data Matrixes** section in this chapter
- The Locator tool can be advantageously used to locate the objects upon which you need to read a data matrix. However, if input images contain only one data matrix it may be more time efficient to locate the symbology with the Data Matrix Reader, without using a Locator tool.

Elements of a Data Matrix

Each data matrix symbology contains a data storage area, constituted of nominally square modules, set out in a regular array. The data region is surrounded by a Finder Pattern, and a Quiet Zone border. Each symbol typically has a borders that can be solid or broken. Figure 25 illustrates the elements of a data matrix symbology.

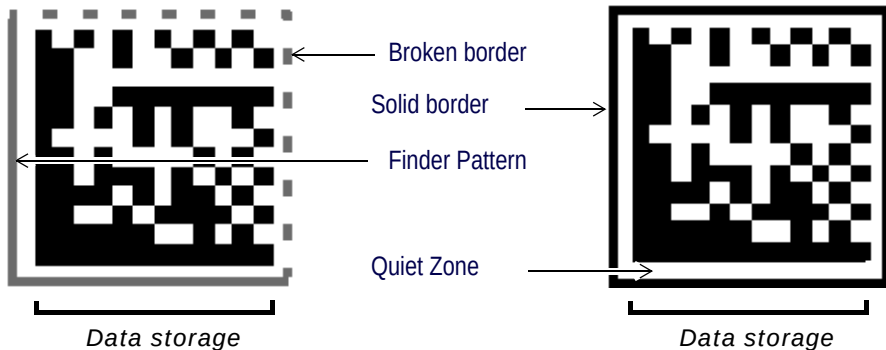


Figure 425 Elements of a Data Matrix Symbology

- **Solid Border:** The Data Matrix Reader uses the solid border to calculate the rotation of a data matrix symbol.
- **Broken Border:** The Data Matrix Reader uses the broken border to identify the number of rows and columns in a symbol.
- **Data Storage Area:** The data storage area contains the binary information that is encoded in the Data Matrix.
- **Quiet Zone:** The Quiet Zone is a clear white space surrounding the data matrix, without clutter or text of any kind. The width of the quiet zone must be at least the size of one cell. The quiet zone should be at least 10% of the width or height of the symbol in order to achieve an optimum read rate.
- **Finder Pattern:** The Finder Pattern is a perimeter to the data region and is one module wide. Two adjacent sides (left and lower sides) form an L-shaped boundary consisting of solid dark lines. These sides determine physical size, orientation and symbol distortion. The two opposite sides are made up of a broken borders and are used to define the cell structure of the symbol as well as assist in determining physical size and distortion.

General Properties

Properties listed as **General Properties** in the user interface are those related to the physical characteristics of data matrixes. These properties are all auto-detected and set by a successful Learn (see the **Learn Process** section) but only when the property has been enabled.

A property should be enabled when its value is known and is not expected to vary at runtime. Enabling properties generally improve execution time. For example, if the **Polarity** property is *disabled*, the tool searches for all possible polarities, however if the **Polarity** property is *disabled*, the tool restricts its search to symbols having the configured **Polarity** value.

Height and Width

When these properties are enabled, they respectively set the required height and width, pixels, for data matrixes to be decoded. When these properties are disabled, no height and/or width constraints are applied when detecting valid matrix instances.

Number of Rows and Columns

When these properties are enabled, they respectively set the required number of rows and columns for data matrixes to be decoded. When these parameters are disabled, the tool does not require a specific number of rows or columns to when detecting valid matrix instances.

Polarity

When this property is enabled, the tool will only decode matrixes of the selected polarity: **Light On Dark** (light matrix on a dark background or **Dark On Light** (dark matrix on a light background). When **Polarity** is disabled the tool searches for instance of either polarity.

ECC Level

Data matrix symbologies are identified by their ECC (Error Correction Code) type. There are two basic ECC types:

- **ECC 000** to **ECC140** symbols consisting of an odd number of rows and columns, from 9x9 to 49x49, not including quiet zones. The upper-right corner module is dark in the case of a dark matrix on a light background; the upper corner is light in a light on dark matrix.
- **ECC 200** symbols consist of an even number of rows and columns, from 10x10 to 144x144, not including the quiet zone. Some may be rectangular with sizes from 8x8 to 16x48. In all ECC 200 symbologies, the upper right corner being is light in the case of a dark matrix on a light background; the upper corner is dark in a light on dark matrix.

When **ECC Level** is enabled, the tool will only decode matrixes of the selected ECC type. When this property is disabled the tool searches for matrixes of any ECC type.



Figure 426 Data Matrix Examples with Different ECC Levels

Image Style

When this parameter is enabled, the tool will only decode matrixes of the selected style: **Mirror** (default value) or **Normal**. When **Image Style** is disabled the tool searches for instances of either style.

Orientation

The value of the **Orientation** parameter is a nominal value. Allowable values range from 0 to 315, in increments of 45 degrees. The angle of orientation of a data matrix is the counterclockwise angle between the X-axis of the data matrix (as defined by its finder pattern) and the X-axis of the Rectangle region of interest.

Variation from the nominal **Orientation** value can be set by the **Angle Variation** parameter.

Advanced Parameters

Some advanced parameters are auto-configured by a successful Learn.

Average Element Size

This parameter specifies the average size, in pixels, of the cells in the data matrix. The value set by a successful Learn reflects the actual cell size of the data matrix and is the most suitable value for a robust performance.

- For optimal performance, the field of view should be set so that the learned **Average Element Size** value falls between 5 and 10 pixels.
- With relatively large and well contrasted matrixes, the value can be lowered slightly to speed up the read cycle.
- A value lower than 4 pixels will increase the processing time significantly and the robustness of the algorithm may degrade.
- When using the **Local Threshold** or **Adaptive Threshold** methods, **Average Element Size** should be set to a value greater than 5.

Minimum Edge Strength

This parameter sets the minimum greylevel value required for detection of edges within the data matrix symbology. The **Minimum Edge Strength** value is auto-detected and configured by a successful Learn.

- For extremely low-contrast images, the edge strength value can be set to a value between 5 and 18. A value of 18 allows the program to read the data matrix with contrast as low as 10%.
- If the data matrix's contrast is stable and unlikely to change, the level can be raised to 36 to produce the optimum run performance.
- The edge strength can be raised to higher values when the matrix is of relatively high contrast and the background clutter is of relatively low contrast.

- With higher edge strength, the program will ignore most of the low contrast objects and get to the higher contrast matrix area quicker.

Samples Per Cell

The Data Matrix Reader samples and averages portions of each cell to determine whether the cell is light or dark. **Samples Per Cell** sets the number of samples taken for each matrix cell.

- The value of this parameter is auto-configured by a successful Learn.
- When using the **Local Threshold** or **Adaptive Threshold** methods, **Samples Per Cell** should be set to a value greater than 5.

Border Integrity

Border Integrity specifies the percentage of undamaged border required for a data matrix to be read. For example, a value of 80% will allow the tool read data matrixes having up to 20% of mismatch or damage in their border.

Angle Variation

Angle Variation sets maximum allowable variation, in degrees, of the angle of the data matrix to read, with respect to the angle set by the **Matrix Orientation** parameter. The default value is +/- 23 degrees.

- **Angle Variation** is ignored if **Matrix Orientation** is disabled.
- Minimizing the angle variation can provide a minimal improvement in speed. For example, if the data matrixes to be decoded do not change their orientation by more than +/- 10 degrees, the **Angle Variation** should be set to 10.

Size Variation

The size of a data matrix is defined by the **Height** and **Width** parameters. The **Size Variation** parameter sets the allowable variation in size that will allow the tool to still locate and decode data matrix instances. By default, this parameter is set to its maximum value of 10%.

- This parameter has no effect if both the **Height** and **Width** parameters are disabled.
- Minimizing this value can help improve the speed and accuracy.
- When the variation of symbologies to be inspected exceeds 10%, both **Height** and **Width** should be disabled.

Width / Height Ratio

Enabling **Height / Width** ratio instructs the tool search for only for data matrixes having the specified aspect ratio; the allowable range is 0.3 to 5.0.

- This parameter has no effect when either the **Height** or **Width** parameters, or both, are enabled.
- When neither **Height**, **Width**, nor **Width / Height Ratio** are enabled, the tool will search for symbologies of any size and aspect ratio.

Fine Tune Method

Fine-tune methods provide options to enhance the performance of the data matrix and overcome specific decoding problems.

Position Enhance

This option enables an extra enhancement step for reading high density data matrixes with low resolution.

Intensity Enhance

Enabling this option can help decode matrixes in which there exists a dramatic intensity variation over the border area; for example, poorly-marked data matrixes having some cells that are almost invisible compared to the rest of the cells. In such cases, a failure to decode the data matrix is generally identified by an "...Edge Not Found" status code: 4604, 4605, 4611 or 4612.

No Quiet Zone Clutter

Enabling this option, speeds the reading of a data matrix that has a sufficient, uncluttered quiet zone.

Quiet zone clutter is typically associated with low quality inkjet or dot-peen marks or with data matrixes that are imaged at an angle.

Timing Borders

Enabling this option enables the tool to locate and decode defective data matrixes that are missing the top timing border or the right timing border.

To learn this type of data matrix, you must either use a defective label or draw the Learn region of interest without including the matrix border. You must provide the correct **Number of Rows**, **Number of Columns**, and data matrix **Polarity** and enable the appropriate **Fine-Tune Method: No Top Timing Border** or **No Right Timing Border**.

Allow Steep Angle

Enabling this method improves the tools capacity to read a data matrix with severe geometrical distortion. This may occur camera's focal plane cannot be set parallel to the surface of the data matrix label, resulting in a steep angle between the focal plane and the label surface.

Allow Severe Damage

Enabling the option will increase the robustness of the software for reading data matrixes with severe border damage. To use this option, first perform a successful Learn on a less damaged label, then enable the option to read the damaged labels in runtime.

Search Speed

The default **Normal**, speed is the most robust search speed. For a high-contrast, well-formed data matrix, selecting **Overdrive** or **Turbo** mode may reduce the execution time.

Search Direction

By default, the tool applies the default **Horizontal->Vertical** search direction. In this case, the tool first searches horizontally for symbology instances then, if none is found, it searches vertically. Selecting **Vertical->Horizontal** causes the opposite search process. The search direction can

also be set to only **Vertical** or **Horizontal**, which typically provides a faster execution time than a dual-direction search.

Warp Method

The default is **Fast** method generally provides satisfactory results with much higher speed.

- For very poor quality images, **Slow** may provide more robustness.
- The **Slow** method should be selected when the tool is configured to use the **Local Threshold** or **Adaptive Threshold** methods.

Threshold Method

Three threshold methods are available: **Global**, **Local**, and **Adaptive**. These three methods are not exclusive and can be enabled simultaneously.

- The **Global** method is usually faster than both the **Local** and **Adaptive** methods and is generally sufficient for reading high quality marks.
- The **Local** method is recommended for symbologies in which the spacing between dots is uneven, such as dot-peen marks or inkjet printer marks.
- When the background of the matrix is uneven due to poor marking or lighting, selecting **Adaptive** may produce better decode results.
- For the **Local** and **Adaptive** methods to work efficiently, **Average Element Size** and **Samples Per Cell** should both be set to a value higher than 5 and the **Warp Method** should be set to **Slow**.

Learn Process

All parameters for data matrix detection and reading can be set by the user, either programmatically or through the interface. To assist the user in configuring parameters, the Data Matrix Reader provides a Learn process that automatically detects and sets the parameters required to process a given symbology.

When the **Learn** process is executed, the Data Matrix Reader attempts to locate and decode a symbology in the current input image.

- The Learn should be carried out on a bar code image that is typical of the bar code instances the tool will be required to decode in runtime.
- A successful (**Learn Succeeded**), optimizes and sets the following parameters: Height, Width, Number of Columns, Number of Rows, Polarity, ECC Level, Orientation, Image Style, Minimum Edge Strength, Average Element Size, Samples per Cell, Width/Height Ratio, and the Threshold method.
- An unsuccessful Learn indicates image quality is inadequate, that parameters need to be changed or both.
- You can only carry out a Learn for one data matrix instance at a time. Each time you carry out a new Learn, parameters, including the data matrix type, are reset accordingly.
- If you know the parameters that are required, you can set them programmatically, or through the interface without carrying out a Learn process.

Preprocessing and Image Quality

Good image quality is significant factor in obtaining a successful Learn. However, the learned data matrix should be typical of the symbologies that will be processed by the application.

Since ideal conditions are not always possible, the Data Matrix Reader can apply one of several image processing functions before applying any other functions, including a Learn.

Reading Multiple Data Matrixes

The Data Matrix Reader can process up to 5 data matrixes in each input image. These instance can be of different types, providing that the configured parameters allow the decoding of the different types.

Multiple Data Matrix Reader Tools

If you need to process more than one type (ECC level) of data matrix in an application, we recommend that you instantiate one Data Matrix Reader tool for each type of matrix you need to read. This allows you to set the optimal detection parameters for each matrix type.

Single Data Matrix Reader Tool

If you use only one Data Matrix Reader tool to process more than one type of data matrix, you will have to set all parameters "broadly" enough to deal with the range of matrix types covered by your application.

To do this, learn each matrix separately and record its properties. Next, set parameters (manually or programmatically) so that they take into account the properties learned from each data matrix.

Using the Data Matrix Reader

Setup of the Data Matrix Reader is carried out on four tab-selected panels: General, Location, Transform, and Results. Steps for using and setting up the Data Matrix Reader will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the Data Matrix Reader and enter the tool's input and output Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

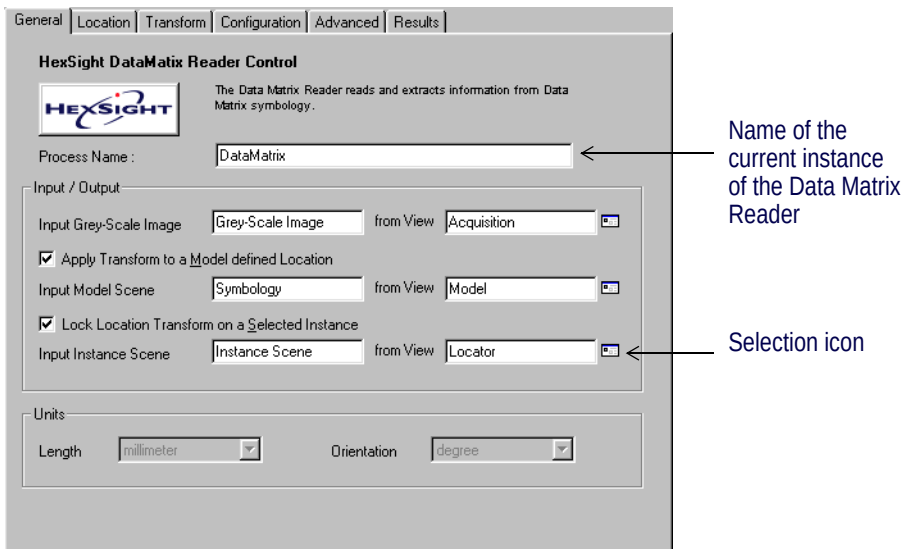


Figure 427 Data Matrix Reader — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the **Inspection Tools** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes

Table 45 Requirements for Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 45 Requirements for Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSDataMatrix#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Data Matrix Reader configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Data Matrix Reader. This Scene is provided by a HexSight Data Matrix Reader

configured to output its Instance Scene in the runtime database. See the **Data Matrix Reader** chapter of this guide for more details.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

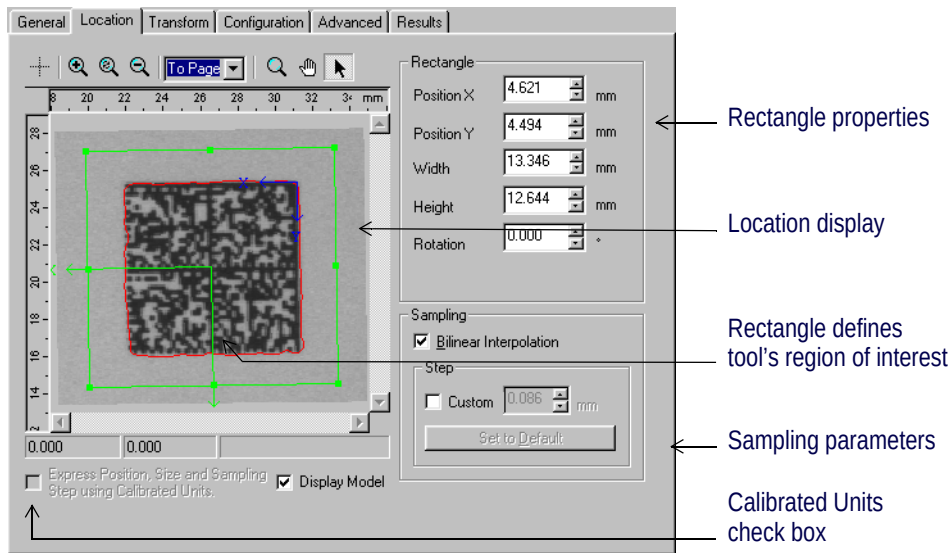


Figure 428 Data Matrix Reader — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is ignored when the tool is Image-based.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.
- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

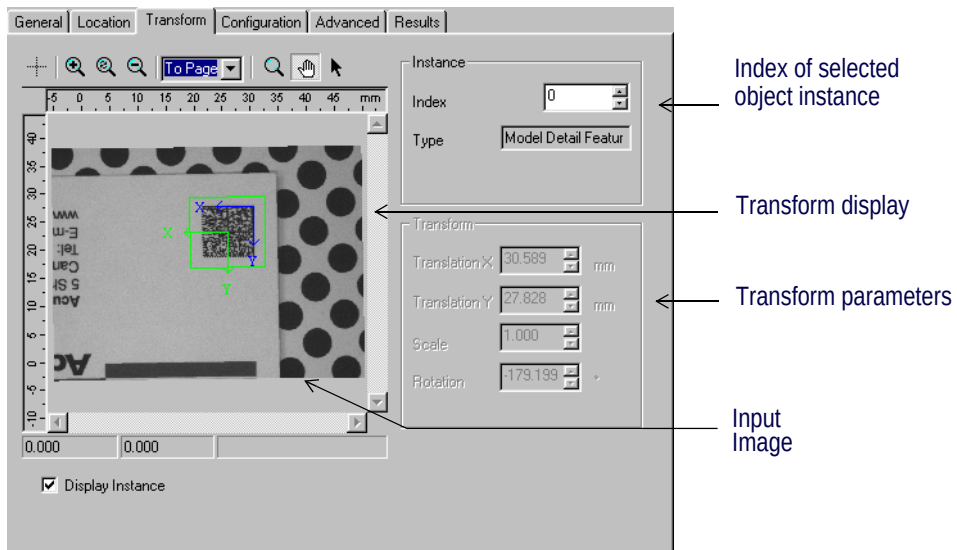


Figure 429 Data Matrix Reader — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected

since the instance processed does not match the type of object for which the tool is configured.

- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest. When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.
- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

Use this panel to configure general data matrix parameters either manually or automatically, through the Learning process. This panel also provides an image processing function.

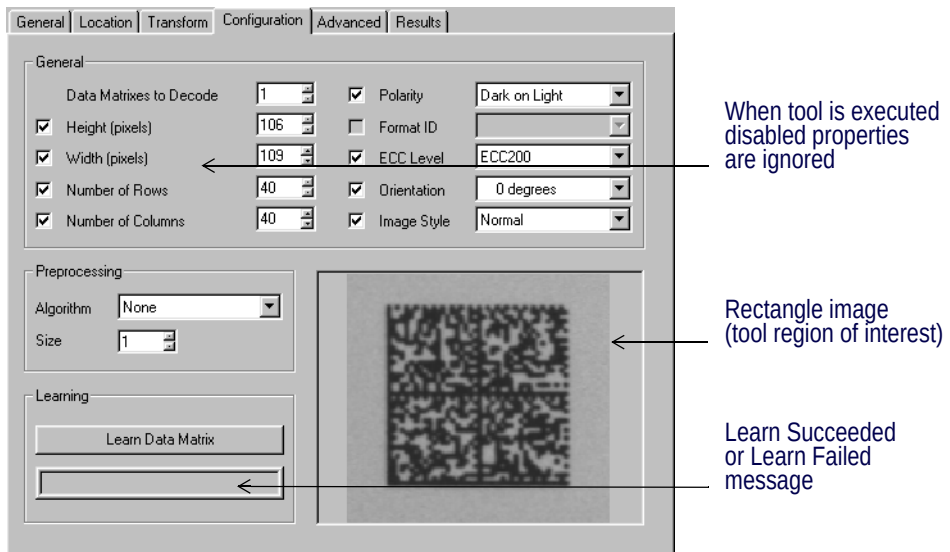


Figure 430 Data Matrix Reader — Configuration Panel

General Properties

General properties can be automatically set by carrying out a successful Learn, using the **Learn Data Matrix** command. This is recommended if you are not familiar or do not know beforehand how to configure the **General** parameters.

In runtime, you should typically enable **General** properties to ensure a faster execution time. A disabled property signals to the tool to search for all possibilities covered by the parameter. For example, the Data Matrix Reader will attempt to look for both normal and mirror occurrences of a symbology if **Image Style** is disabled.

However, you should disable properties that may not apply to all the matrixes to be read. For example, if matrix height and width may change significantly during the inspection, disable **Height** and **Width** check boxes.

- **Data Matrixes to Decode:** Sets the number of symbols the tool must find in the image.
- **Height:** Sets the height, in pixels, of the data matrix.

- **Width:** Sets the width, in pixels, of the data matrix.
- **Number of Columns:** Sets the number of columns in the data matrix.
- **Number of Rows:** Sets the number of rows in the data matrix.
- **Polarity:** Sets the polarity of the data matrix. In a **Dark on Light** symbol, the background is lighter than the data matrix. A **Light on Dark** symbol has a dark background.
- **Format ID:** This parameter is not currently implemented.
- **ECC Level:** This is the Error Correction Level used in the matrix.
- **Orientation:** Enabling and providing an orientation value can help speed up the detection process. Enabling this parameter *will not* prevent the tool from finding a matrix *unless* its actual orientation is opposite, or nearly opposite, to set **Orientation** value (for example 0 vs. 180 or 45 vs. 225). For a known orientation range, select an **Orientation** value, then if required, set the allowable variation from this angle with the **Angle Variation** parameter, in the Advanced panel.
- **Image Style:** Normal is the default. Mirror can be set if data matrixes to be read appear as a mirror image.

Preprocessing

The Data Matrix Reader includes a set of preprocessing functions that allow you to enhance or modify the input image, before it is learned or decoded. Only one image processing operation can be applied.

By default, no preprocessing **Algorithm** is applied (**None**) is carried out. The **Size** text box sets the parameter for the selected preprocessing algorithm where applicable. Available preprocessing options are:

- **Invert:** Reverses the polarity of the image. For example, changing a light on dark data matrix to a dark on light data matrix. The **Size** parameter is ignored.
- **Erode:** This option enables you to increase dark cell size (of dark on light data matrix). The **Size** parameter sets the value of the

neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.

- **Dilate**: This option enables you to increase light cell size (of light on dark data matrix). The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Open**: This option enables you to remove minor light defects of dark cells. The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Close**: This option enables you to remove minor dark defects of light cells. The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Gaussian**: The Gaussian operation acts like a low pass filter and helps remove noise and has overall blurring effect on the image. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Sharpen**: This has the effect of sharpening the image, especially edges. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Sharpen Low**: This has the effect of sharpening and smoothing the image at the same time. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Median**: The Median operation reduces impulsive image noise without degrading edges or smudging intensity gradients. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Histogram Stretching**: This operation increases the contrast in the image. The **Size** parameter is ignored.

Learn Data Matrix

Learn Data Matrix launches a Learn function that auto-detects and sets the optimum parameters required to find and read the current data-matrix.

- When the Learn is successful (**Learn Succeeded** message), the tool sets the parameters to the values required to read data matrix in the current image.
- When the **Learn** fails (**Learn Failed** message appears), the **Assisted Learn** dialog opens. Use this dialog to fine-tune parameters until the Learn is successful. See the following ***Assisted Learn Interface*** section for more details.
- Each new successful Learn resets the detection parameters in the Configuration and Advanced panels.
- A Learn can only be carried out for a single data matrix.

Assisted Learn Interface

When the Learn process fails, the Assisted Learn interface allows you to manually fine-tune and set parameters until you obtain a successful Learn.

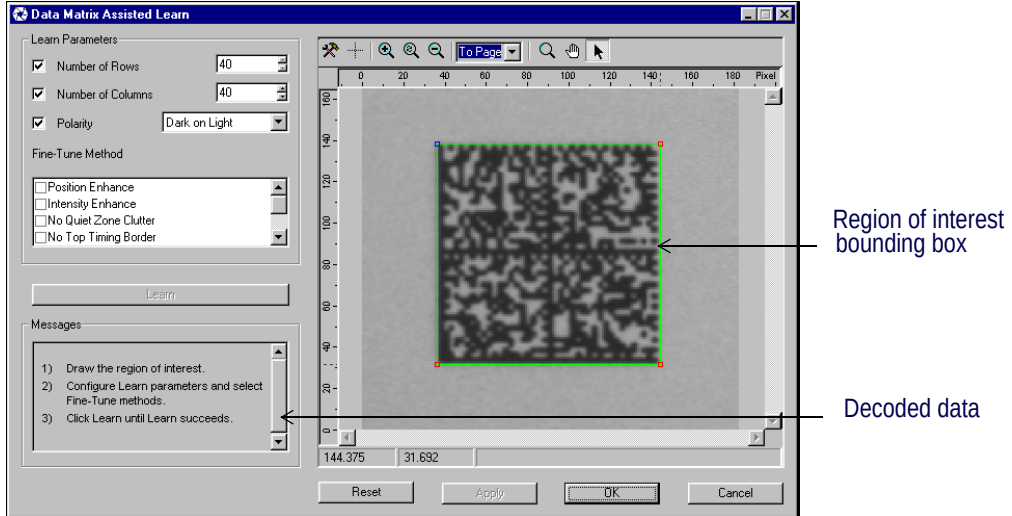


Figure 431 Data Matrix Reader — Assisted Learn Interface

Carrying out an Assisted Learn

- Drag a bounding box around the data matrix that is displayed in the image. If necessary, readjust the sides of the bounding box so that they are snapped tightly to the edge of the matrix, as shown in [Figure 431](#).
- Use the corner handles to adjust the size of the bounding box so that tightly encloses the matrix, i.e. closely snap the rectangle sides to the matrix edges, as show in [Figure 431](#).
- Set known Learn Parameters, for example the number of columns, and Fine-Tune methods that apply.
- Click **Learn**: The Messages box will report if the Learn has succeeded or failed. When the Learn is successful, the Messages box also displays the decoded data.
- If the Learn fails, you will need to make further changes before reattempting a Learn.

Reattempting a Learn

If the Learn fails, reverify and set known properties for the data matrix. If you manually set and enable Learn properties, the Learn process will not auto-detect their real values. Here are a few pointers to help you carry out a successful Learn:

Learn Region of Interest

Draw the region of interest so that it tightly encloses the matrix.

Tool Rectangle

In the Location panel, make sure that the sides of the Rectangle region of interest the matrix edges by at *least* 25 pixels.

Preprocessing

A preprocessing function (Configurations panel) may help you rectify or improve an image quality problem.

Fine-Tune Method

Some **Fine-Tune Methods** may need to be enabled prior to a Learn, for example when the matrix is missing a timing border. For more details, see ***Fine-Tune Methods*** in the ***Advanced Panel*** section.

Advanced Panel

Use this panel to configure, adjust or fine-tune parameters that improve tool performance.

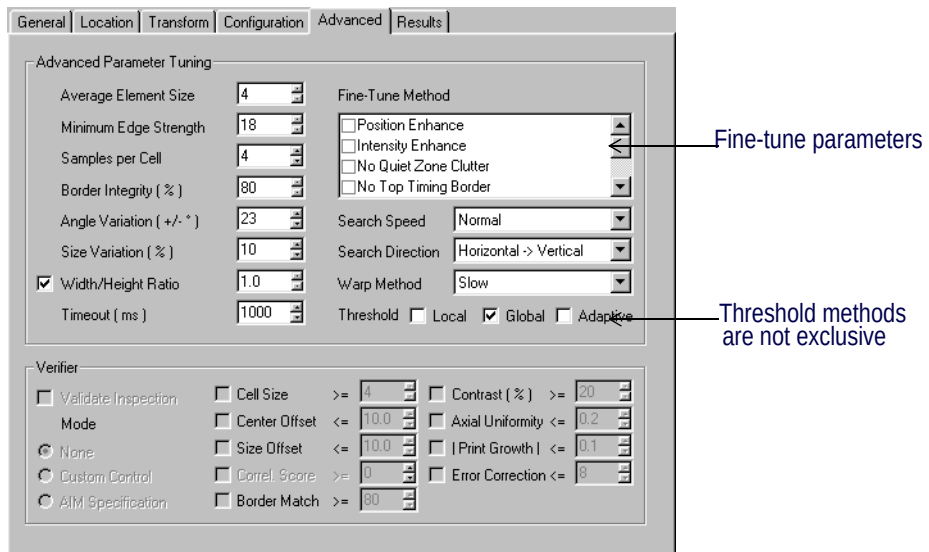


Figure 432 Data Matrix Reader — Advanced Panel

Average Element Size

This parameter defines the average size, in pixels, of the cells in a data matrix. The valid range is from 2 to 20 pixels.

- Normally, the field of view for the data matrix should be set such that after a successful Learn, the cell size is reported at a value between 5 and 10.
- If the learned **Average Element Size** drops below 4 pixels the reading time will increase and the robustness of the algorithm may degrade.
- The learned value reflects the actual average cell size of the data matrix and is the recommended setting for most applications.
- For high quality images it may be possible to increase the element size, without affecting read rates, in order to accelerate the read time.

Minimum Edge Strength

This is the required minimum edge strength, in greylevel values. The value can be set between 5 and 100.

- Use lower values for low-contrast images.
- Higher values can be used for high contrast images with a relatively low-contrast and low-clutter backgrounds.
- For extremely low contrast images, the edge strength value can be set between 5 and 18.
- The edge strength can be raised to higher values when the matrix is of relatively high contrast and the background clutter is of relatively low contrast. With higher edge strength, the Data Matrix Reader will ignore most of the low contrast objects and get to the higher contrast matrix area quicker.
- A value of 18 allows the program to read the data matrix with contrast as low as 10%. If the label's contrast is unlikely to change, for example, when the lighting source is stable, the level can be raised to 36 to produce the optimum run performance.

Samples Per Cell

Samples per Cell sets the number of samples taken by the tool in each matrix cell to determine whether the cell is light or dark. If you have enabled the **Local** and/or the **Adaptive Threshold** methods, set **Samples Per Cell** to 5 or greater.

Border Integrity

This is a percentage value. The recommended (default) value is 80%, which means a 20% border mismatch or damage is tolerated.

Angle Variation

Angle variation sets the allowable variation from the angle set by the **Orientation** property.

- If **Orientation** is disabled then Angle Variation is ignored.
- By default, the angle variation factor is set to +/-23, which in runtime will allow the tool to locate and decode matrixes that are rotated up 23 degrees from the configured orientation.
- Minimizing the angle variation may help reduce the execution time.

Size Variation

The size, representing the height and width of a Data Matrix, is set after a successful Learn. When the data matrixes in subsequent inspections are of similar size, the learned size applies to these matrixes.

- The default value is 10%, which means that the size of matrixes can vary up to 10% from the nominal values set by a successful Learn, regardless of the cell size used in the setup.
- When the variation exceeds 10%, both **Height** and **Width** should be deselected (unchecked).
- Minimizing the size variation may help reduce the execution time.

Width / Height Ratio

This tells the program to only search for a Data Matrix symbol with the specific aspect ratio when either Height or Width is unchecked. The value ranges from 0.3 to 5.0.

- This parameter is ignored when both **Height** and **Width** parameters are specified.
- When neither **Height**, **Width**, nor **Ratio** are selected, the system will look for data matrix of any size and aspect ratio.

Timeout

Timeout sets a limit to the processing time. When the timeout is attained, the decoding stops. The results are output for all data matrixes decoded up to the timeout.

Fine-Tune Method

Fine-tuning serves two main purposes. Parameter fine-tuning after a successful Learn can accelerate the normal execution of the tool and/or make the process more robust. In the case when the Learn is not successful for a certain symbol, parameter fine-tuning may provide the Data Matrix Reader more information about the symbol, thus improving symbol decoding in the Run mode.

- **Position Enhance:** Will attempt to reduce the number of error correction bits used for a high density data matrix with low resolution (e.g., 2 ~ 3 pixels per cell). This option can be turned on or off. When turned on, an extra enhancement step is executed. The default is Off.
- **Intensity Enhance:** Designed to overcome dramatic intensity variation over the matrix border area. For example, certain poorly marked data matrix may have some cells that are almost invisible compared to the rest of the cells. Without enabling the option, the system may issue a status code indicating a certain edge cannot be found. Enabling the option will help read this type of data matrix more consistently. The default is Off.
- **No Quiet Zone Clutter:** Enabling this option will speed up the reading process when data matrix has sufficient quiet zone. With sufficient quiet zone, a data matrix with irregular, curved, or distorted borders can be read more efficiently with the option checked. The border problem typically is associated with low quality inkjet or dot peen marks or when the data matrix is imaged at an angle.
- **No Top Timing Border:** The Top or Right timing borders refers to the timing borders on top or right of the data matrix when the data matrix has no rotation. Sometimes, a data matrix may be printed without the top timing border or the right timing border. To read this type of defective data matrix, first perform Learn data matrix. When it fails, follow the instructions in the Status Bar
- **No Right Timing Border:** Refer to above when dealing with defective data matrix without the right timing border.

- **Allow Steep Angle:** In some data matrix reading applications, it is not possible to set up the camera such that the focal plane is in parallel with the surface of the data matrix label. When the focal plane and the label surface.
- **Allow Severe Damage:** Enabling the option will increase the robustness of the software in reading data matrix with severe border damages. To read such matrixes, first perform a successful Learn on a less damaged label. then enable **Allow Severe Damage** for executing the tool in runtime.

Search Speed

The default for this setting is **Normal**, which is the most robust search speed.

- For a high contrast well formed data matrix, selecting **Overdrive** or **Turbo** mode may reduce the total locate/decode time

Search Direction

The default is **Horizontal ->Vertical**, which causes the Data Matrix to first search horizontally for a symbology. When no symbology is found or decoded, the algorithm then searches vertically for the Data Matrix. **Vertical ->Horizontal** causes the opposite search process.

- **Horizontal** and **Vertical** options restrict the Data Matrix to searching only in the specified direction.

Warp Method

The default **Fast** setting typically provides satisfactory results with much higher speed. Use **Slow** for very poor quality images.

Threshold

The three **Threshold** methods **Global**, **Local**, and **Adaptive** are not exclusive; you can enable more than one method.

- The **Global** method is usually faster than both the **Local** and **Adaptive** methods and is generally sufficient for reading high quality marks.

- Enable the **Local** method when symbologies have uneven spacing between dots is uneven, such as dot-peen marks or inkjet printer marks.
- Enable **Adaptive** when the background of the matrix is uneven due to poor marking or lighting.
- For the **Local** and **Adaptive** methods to work efficiently, **Average Element Size** and **Samples Per Cell** should both be set to a value higher than 5 and the **Warp Method** should be set to **Slow**.

Results Panel

The Results panel shows the results of the Data Matrix Reader process, including a visual display of the sampled Image. Read only text boxes provide dimensional and positional informations on this Image.

The screenshot shows the 'Results' tab of the Data Matrix Reader software. It contains a table of results, a section for elapsed time and coordinate system, and a messages section displaying decoded data and corner coordinates. Annotations with arrows point to specific elements:

- Grid of results**: Points to the table of results.
- Enable this check box to display corner coordinates**: Points to the 'Show Corner Coordinates' checkbox.
- Decoded data**: Points to the first line of the decoded data in the messages section.
- Corner coordinates expressed with respect to selected Coordinate System**: Points to the corner coordinates in the messages section.
- Status code**: Points to the status code '0000' in the messages section.

	1	2	3	4	5
Status Code	0000				
Angle	0.00				
Contrast	42				
Number of Error Bits	168				
Damage Percentage	12.96				
Border Match Percentage	97.00				
Threshold	78				
Locate / Decode Time (ms)	25				

Elapsed Time: 69.060 ms ☒ Show Corner Coordinates

Coordinate System: Tool

Messages:

```

0000 RVSI Acuity CMatrix
5 Shawmut Road
Canton, MA 02021
Tel: 781-821-0830 Fax: 781-828-8942
www.rvsi.com
(-4.00,4.00) (4.00,4.00) (4.00,-4.00) (-4.00,-4.00)

```

Figure 433 Data Matrix Reader — Results Panel

Grid of Results

The grid of results displays the following for each data matrix instance read by the tool.

- **Status Code:** The following table describes the possible status codes.

Table 46 Status Codes in Results Messages

Numeric Code	Description
0000	Success
1000	No Execute Performed
3048	Symbology Decode Failed
4601	No Edge Candidate Found
4604	First Edge Not Found or Too Small
4605	Second Edge Not Found
4611	Third Edge Not Found
4612	Fourth Edge Not Found
4620	Symbology Corner Out of Image
4621	Size Test Failed
4622	Unable to Find Row/Column
4633	Border Match Failed
4630	Inspection Timed Out
4301	Trigger Overrun
4303	String Mismatch
4304	Angle Match Failed
4305	Excessive Damage
4310	Process Overrun
6001	OCR Conformance Failure
6002	OCR General Error
9999	Dongle Missing

- **Angle:** The angle of rotation of a data matrix is the counterclockwise angle between the data matrix's X-Axis and the X-Axis of the Rectangle, i.e. the region of interest. See examples in [Figure 434](#).

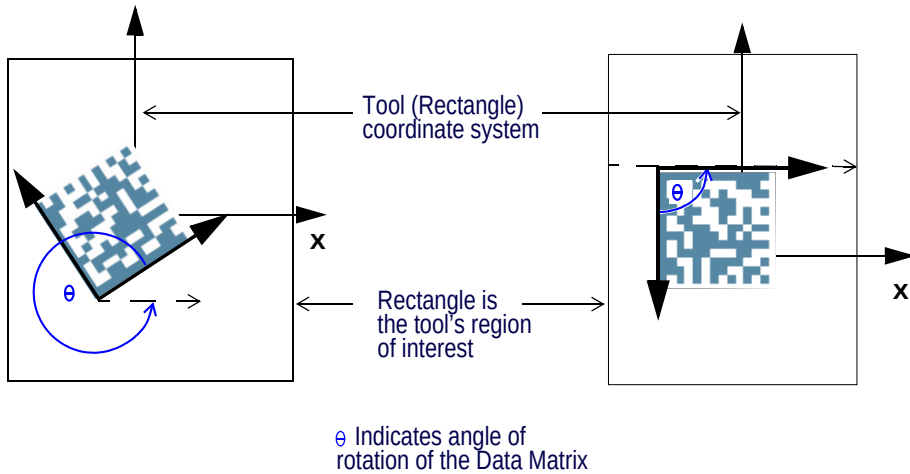


Figure 434 Data Matrix Angle of Orientation

- **Contrast:** The percentage of contrast in data matrix instance. A value of 100% indicates a contrast of 255.
- **Number of Error Bits:** Number of error bits found in each decoded Data Matrix.
- **Damage Percentage:** The percentage of damage in a data matrix instance. Damage percentage is calculated as the total number of error bits divided by the total number of bits in the matrix.
- **Border Match Percentage:** Measures how good the border of each Data Matrix is, when compared to ideal.
- **Threshold:** Threshold used to detect the cells in a given data matrix instance.
- **Locate/Decode Time:** This is the time required to read the given symbology instance, expressed in milliseconds.

Elapsed Time

Elapsed Time shows the total execution time of the Data Matrix Reader tool.

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image**, **World**, **Object**, and **Tool**.

Show Corner Coordinates

When **Show Corner Coordinates** is enabled, the **Message** box displays the X-Y coordinates for the four corners of each decoded symbology instance. These coordinates are expressed with respect to the selected **Coordinate System**: **Tool**, **Image**, **Object** or **World**.

- When the World, the Object or the Tool coordinate system is selected, corner coordinates are expressed in the user-selected length units.
- Coordinates are expressed in pixels when the Image coordinate system is selected.
- In the exceptional case where the tool is not configured to work with calibrated units, coordinates in the Tool coordinate system are expressed in pixels but values cannot be returned in the World or in the Object coordinate system.

Messages

The **Message** box provides information for each decoded symbology instance: the Status code, the decoded alphanumeric data, and, when **Show Corner Coordinates** is enabled, the corner coordinates of each decoded matrix.

OCR Fixed Font

The OCR Fixed Font tool recognizes and reads fixed-font character strings.

OCR Fixed Font Chapter Outline

[Understanding OCR Fixed Font Concepts](#)

[Region of Interest](#)

[Learn Process](#)

[Configuration Properties](#)

[Custom Fielding](#)

[Preprocessing and Image Quality](#)

[Using the OCR Fixed Font](#)

[General Panel](#)

[Location Panel](#)

[Transform Panel](#)

[Configuration Panel](#)

[Advanced Panel](#)

[Results Panel](#)

Understanding OCR Fixed Font Concepts

This section provides details on the basic concepts of the fixed font OCR process. Basically, the OCR Fixed Font tool can read up to four lines of character strings having the same font. Three standard fixed fonts are supported: SEMI, OCR-A, and OCR-B.

Region of Interest

The tool's Rectangle is the region of interest within which the process finds and reads character strings. The positioning of the tool Rectangle is presented in detail in the Symbology Tools chapter, under Tool Positioning.

- The Rectangle must be positioned so that its axes are orthogonal to the character string. Figure 435 shows examples of correct positioning (A, B and C) and incorrect positioning (D). Depending on the configuration of the tool, it can be possible to decode

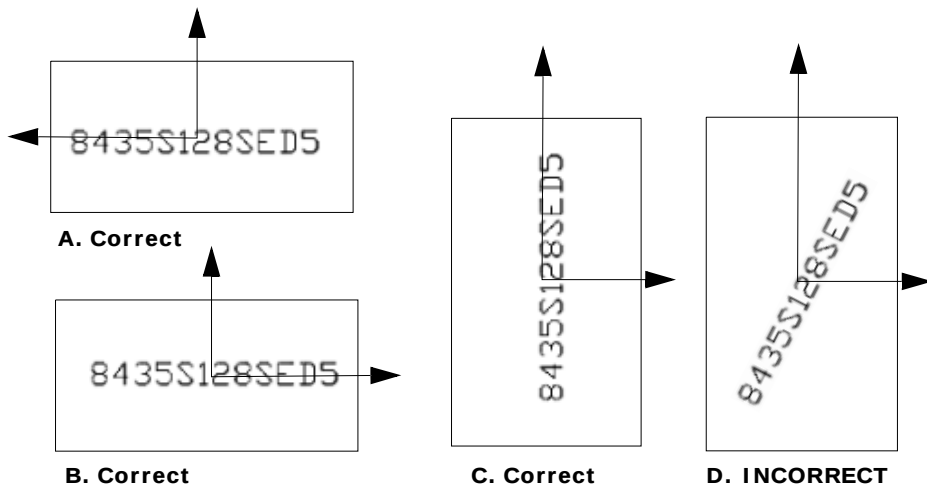


Figure 435 Correct Positioning of OCR Fixed Font Tool

- The Rectangle size must be defined so that the sides of the Rectangle exceed the text string by at least 25 pixels on all sides. If the sides of

the Rectangle are positioned at less than 25 pixels from the text string, both the learning process and character reading may fail.

- Restricting the size of the Rectangle will generally decrease the execution time of the tool.

Learn Process

The Learn process provides a method for optimizing and configuring parameters used in the character reading process. The learn process is not mandatory; the OCR Fixed Font tool can be configured directly by the user when the required parameters and properties for character reading are known.

Before carrying out a Learn, you must first configure the training parameters that will be applied by the Learn process and define both the training string and the primary character that will be learned.

- A successful Learn (**Learn Succeeded**) optimizes the following properties: Character Height, Character Width, Character Spacing and Character Stroke Width.
- A successful Learn auto-configures the following properties: Fine Tune Method, Font Type, Ocr Method, Text Orientation, Text Polarity, Characters on Row, and String Length Variation.
- An unsuccessful Learn (**Learn Failed**) indicates that image quality is inadequate, that parameters need to be changed or both.
- You can only carry out a Learn on a single training string (single row).

Training Parameters

Training parameters, including the **Primary Character** must be configured before carrying out a Learn. A successful Learn sets the value of the **Font Type**, **Ocr Method**, **Text Orientation**, and **Text Polarity** parameters for subsequent executions of the tool. If no Learn is carried out, these parameters will need to be set manually.

Training String

The training string is a string of alphanumeric characters upon which the Learn is carried out. This string should be typical of the characters strings the tool will be required to read at runtime.

The training string is defined by a bounding box and by its alphanumeric values.

Primary Character

The primary character is a character selected within the training sting. This character is used by the Learn process and should chosen such that the size and the stroke width of the character reflects the average size of the characters in the string.

The primary character is defined by a bounding box and by its alphanumeric value.

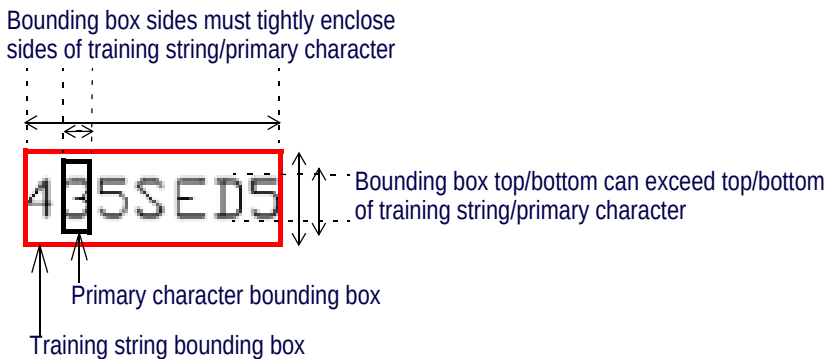


Figure 436 Training String and Primary Character Bounding Boxes

Configuration Properties

The OCR Fixed Font program can read up to four lines of character strings having the same font. The maximum number of characters per row is 30. Three standard fixed fonts are supported: SEMI, OCR-A, and OCR-B.

OCR Methods

One of three OCR methods can be applied for locating and decoding the characters. The default is the Degraded method, which supports only a single line of text, as is typically the case in wafer ID applications. The Standard and Dot Matrix modes can process up to four lines of text.

Degraded Method

This method is the most robust of all three methods and is typically recommended for most applications in which there is only a single line of text to read.

The **Degraded** method is tolerant to noise and damaged characters. It also allows minor rotation from the horizontal or vertical orientation. The speed of the OCR reading is slower than with the other two methods.

If the length of the string may vary during an application that uses this method, you can configure the String Length Variation to accept a variation in string length of up to +/- 10%.

Standard Method

This method works on clean characters with no significant background noise. The method allows the text string to rotate up to 15 degrees from the horizontal or vertical orientation. The speed of the method is relatively fast. The number of characters to be read during the run is allowed to vary from the number used in Learn.

If the number of characters in the string may vary during an application that uses this method, you can enable the Number of Character Vary option (Fine Tune Methods).

Dot Matrix Method

This method is optimized for reading Dot Matrix characters. It has the similar performance as the Standard method except that it is more tolerant of background noise.

If the number of characters in the string may vary during an application that uses this method, you can enable the Number of Character Vary option (Fine Tune Methods).

Text Polarity

Text Polarity whether the characters are **Dark on Light** or **Light on Dark**. The **Degraded** OCR method supports the recognition of either polarity, by enabling the **Unknown** mode. Because the **Unknown** polarity mode reduces the robustness in locating characters, this mode should be avoided whenever possible. **Standard** and **Dot Matrix** methods do not support **Unknown** polarity.

Text Orientation

Text orientation describes the reading direction of character strings. [Figure 437](#) illustrates some of the possible text orientations.

Top to Bottom and **Bottom to Top** orientations apply to strings that are perpendicular to the bottom of the region of interest.

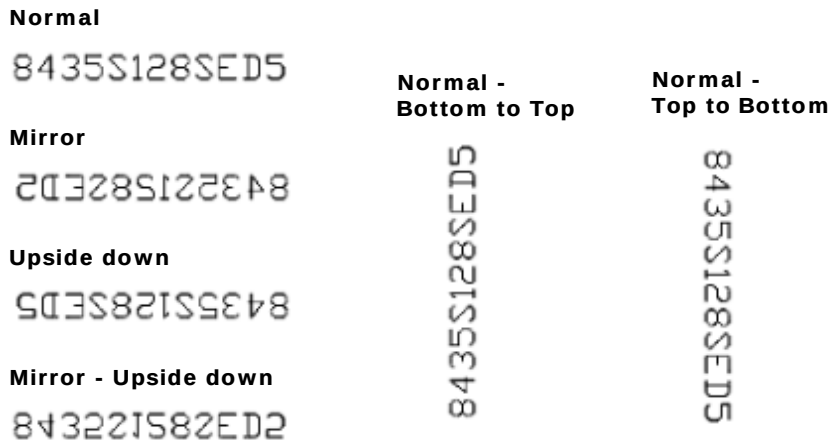


Figure 437 Examples of Text Orientation

Fine-Tune Method

The various **Fine-Tune Methods** provide options to handle special or specific decoding issues. Fine-tune methods are enabled by a successful Learn.

Detect Space

This option enables the tool to read of empty character spaces in the character string. This is automatically enabled by a Learn when the training string contains at least one empty space.

Number Of Characters Vary

This enables the tool to read strings in which the number of characters vary from the number of characters in the training string. This option is supported by **Standard** and **Dot Matrix** methods.

Disable Pre Filter

This option disables the tool's internal prefilter process to prevent the removal of too much image detail.

Character Size Space Vary

This option enables handling of nonuniform character size and spacing. It allows allows character size and spacing between characters to vary up to 10% from the training string values. This option is ignored when **Degraded** method is enabled.

Speed Enhance

Enabling this option applies an OCR algorithm that is faster but less robust.

Remove Noise

This option enables the reading of characters in noisy images. This option is automatically enabled when the **Degraded** method is enabled.

Semi M-Checksum

When enabled, the tool verifies that the read text string passes the SEMI Checksum test.

Custom Fielding

Custom fielding allows you to assign to a specific alphanumeric attributes to individual character positions in a string.

By default, all positions in a character string are defined as alphanumeric. However, a character position can be defined as only **Alpha** only **Numeric** or as a *fixed value*, or *range* of values for example: "G", "3", " " (empty space) or "123A" (range).

Purpose of Defining Custom Fielding Values

Attributing a custom fielding to one or more characters can help reduce the execution time.

Typically fielding values, other than the default **Alphanumeric**, are assigned to character positions whose alphanumeric attributes are known to remain constant in the strings that will be read by the application.

For example if the first three characters in the text strings to be read are always ABC, then 'A' can be attributed to the first position, "B" to the second and "C" for the third.

Custom Fielding Options

If the fielding for a character is set as **Alpha**, the process will always return an **Alpha** result for this position, even if the actual character in the decoded string is a *numeric* value.

Likewise, if the fielding for a character is set as **Numeric**, the process will always return an **Numeric** result for this position, even if the actual character in the decoded string is an *alpha* value.

The **Other** fielding option lets you assign either a specific value or a range of values to a character position.

When the **Other** option attributes a single character to a position, the tool does *not* decode or read the actual character in this position. Instead, the tool automatically returns for the single character value that is attributed to the position. Consequently, the returned **Character Confidence** is always 100 for any position that has been attributed a single specific value.

When a *range* of characters is attributed to a position, for example "123A", the tool will read and output the actual character in this position as one of the values in the range, in this case "1", "2,"3" or "A".

Note: If the strings to be read may contain "empty space" characters, you must enable **Detect Space** in **Fine-Tune Methods**. However if the empty spaces are always in the same position, you can use the **Other** fielding option to attribute " " (empty space) to each of these positions and disable **Detect Space**.

Preprocessing and Image Quality

Good image quality is significant factor in obtaining a successful Learn. However, the learned text string should be typical of the symbologies that will be processed by the application.

Since ideal conditions are not always possible, the OCR Fixed Font tool can apply one of several image processing functions before applying any other functions, including a Learn.

Using the OCR Fixed Font

Setup of the OCR Fixed Font is carried out on four tab-selected panels: General, Location, Transform, and Results. Steps for using and setting up the OCR Fixed Font will follow the left to right order of these panels.

For each panel you will find pointers that will guide you through the setup.

General Panel

Use this panel to name the instance of the OCR Fixed Font and enter the tool's input and output Images and Views. The General panel also configures the tool positioning mode, which affect the required inputs.

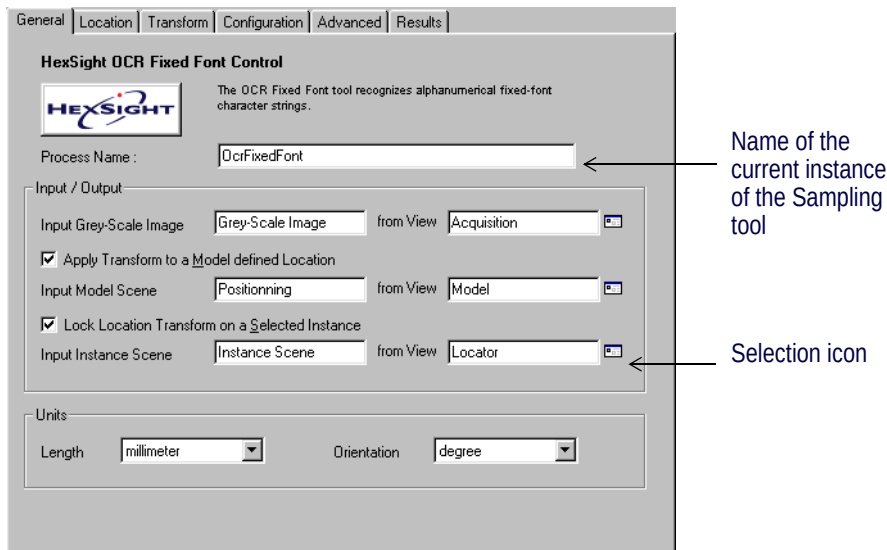


Figure 438 OCR Fixed Font — General Panel

Positioning Modes

The positioning mode defines how the tool's region of interest will be positioned on the input Image. Four positioning modes are available: Image-based uncalibrated, Image-based calibrated, Model-based manual and Model-

based automatic. See the **Inspection Tools** chapter of this guide for more details on tool positioning modes.

The positioning mode you choose to use will affect the input you need to provide to the tool as well as the settings of the following check boxes:

- **Apply Transform to a Model Defined Location**
- **Lock Location Transform on a Selected Instance**
- **Express Position Size and Sampling Step using Calibrated Units**

The table below provides an overview of the required inputs and settings for the four positioning modes.

Table 47 Requirements for Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Inputs	Image	Image	Image Model Scene	Image Model Scene Instance Scene
Apply Transform to a Model Defined Location	disabled	disabled	enabled	enabled
Lock Location Transform on a Selected Instance	n/a (automatically disabled)	n/a (automatically disabled)	disabled	enabled

Table 47 Requirements for Positioning Modes

	Image-Based Uncalibrated	Image-Based Calibrated	Manual Model- Based	Automatic Model-Based
Calibrated Units	disabled	enabled	n/a (automatically enabled)	n/a (automatically enabled)
Other	You must provide the position of the region of interest, in pixels.	You must provide the position of the region of interest, in calibrated units	You must provide the transform (scale factor, rotation and translation) to apply to the region of interest on an object instance	The region of interest is automatically applied to a selected object instance.

Process Name

Process Name sets the name of the current instance of the tool in your application. If you do not enter a name, the Process Manager assigns a default name: HSOCRFixedFont#, where # is replaced by its rank number in the list of instantiated processes.

Input Image

Input Image is the Image in the runtime database on which the process will be applied. This image is usually created in the runtime database by the **Acquisition Device**, although any Image created by any HexSight tool can be used.

- Type the name of the Image in the **Input Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Output Image

Output Image is the Image that will be created in the runtime database by the tool. Since this is an output Image, it will overwrite the selected Image in the runtime database when the tool is executed.

- Type the name of the Image in the **Output Image** text box and the name of the View containing this Image in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Images currently available in the runtime database.

Apply Transform to a Model-Defined Location

Use the **Apply Transform to a Model-Defined Location** check box to enable Model-based positioning modes. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

When you enable Model-based positioning, you must also specify the input Model Scene that will be used to position the tool on the Model.

Input Model Scene

Input Model Scene is the Scene in the runtime database that defines the Model of the object on which the tool will be positioned. This parameter is used only when Model-based positioning is enabled, otherwise it is ignored. This Scene is provided by a Locator configured to output its Models in the runtime database.

- Type the name of the Model Scene in the **Input Model Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Lock Location Transform on a Selected Instance

Use the **Lock Location Transform on a Selected Instance** check box to enable an automatic tool transform when using Model-based positioning. This

parameter is used only when Model-based positioning is enabled, otherwise it is ignored.

If you enable the automatic tool transform, you must also specify the input Instance Scene from which a selected Instance will provide the tool transform.

Input Instance Scene

Input Instance Scene is the Scene in the runtime database that is used to provide transform parameters on instance of objects found by the HexSight Locator. This Scene is provided by a HexSight Locator configured to output its Instance Scene in the runtime database. See the **Locator** chapter of this guide for more details.

- Type the name of the Instance Scene in the **Input Instance Scene** text box and the name of the View containing this Scene in the associated **from View** text box.
- Alternately, you can click the selection Icon to obtain a menu of compatible Scenes currently available in the runtime database.

Units

Length Units

Use the **Length** combo box to select the calibrated unit used to express positions and dimensions. This parameter is used only when Image-based calibrated or Model-based positioning is enabled, otherwise it is ignored.

Orientation Units

Use the **Orientation** combo box to select the unit used to express angles. This parameter is used with all positioning modes.

Note: Click Apply before changing units. Otherwise changes you made to parameters since the last Apply will be lost.

Location Panel

Use this panel to position the tool's Rectangle, which defines the region of interest in the Image to which the process will be applied.

Location Display

In addition to the Image used to position the tool, the **Location Display** also shows the Rectangle that defines the tool region of interest.

When Image-based positioning is used, the tool's region of interest is positioned directly on the input Image. In this positioning mode, the **Location Display** shows the input Image.

When Model-based positioning is used, the tool's region of interest is positioned on the Model Image and this image appears in the **Location Display**.

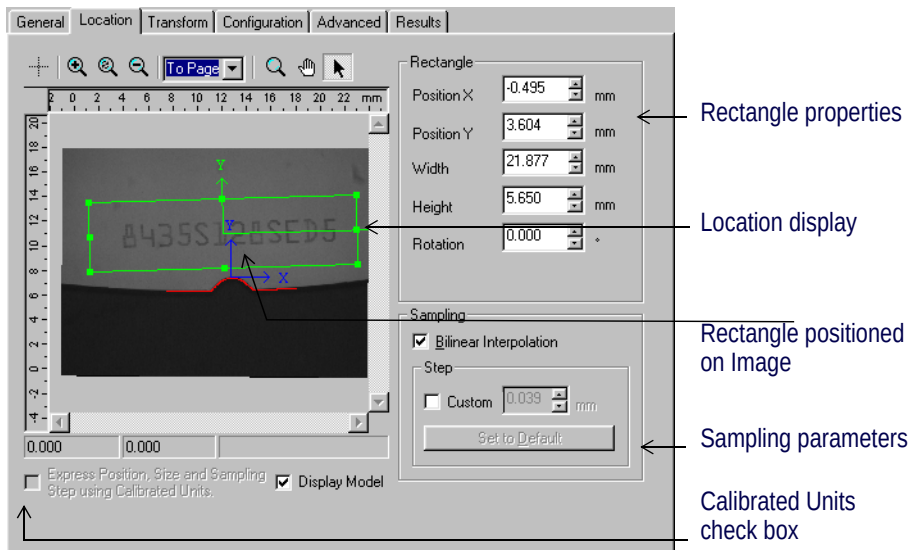


Figure 439 OCR Fixed Font — Location Panel

Express Position, Size and Sampling Step using Calibrated Units

Use the check box labeled **Express Position, Size and Sampling Step using Calibrated Units** to select whether calibrated units or pixels are used with Image-based positioning. This parameter is used only when Image-based positioning is enabled, otherwise it is ignored.

With Model-based positioning, positions and dimensions are always expressed using calibrated units.

Display Model

Use the **Display Model** check box to show the contours of the Model overlaid on its Image. A blue X-Y axes marker indicates the Object coordinate system that is used to position the tool.

This parameter is ignored for when the tool is Imaged-based.

Rectangle Parameters

The tool's region of interest is bounded by a Rectangle defined by the parameters **Position X**, **Position Y**, **Width**, **Height** and **Rotation**.

- **Position X**, **Position Y**, **Width**, and **Height** are expressed in pixels when using Image-based uncalibrated positioning. They are expressed in the selected calibrated unit otherwise. **Rotation** is always expressed in the selected orientation unit.
- With Image-based positioning, the Rectangle is positioned with respect to the Image coordinate system or the World coordinate system, depending on the setting of the **Express Position, Size and Sampling Step using Calibrated Units** check box.
- With Model-based positioning, the Rectangle is positioned with respect to the Object coordinate system.

Placing and Sizing the Rectangle

- Type the Rectangle parameters directly in the text boxes. You can use the associated spin up/down buttons to fine-tune the values.

- Move the Rectangle by dragging its border or the Tool coordinate system marker at the intersection of the X-Y axes.
- Size the Rectangle by dragging any of its eight resizing handles.
- Rotate the Rectangle by dragging the tip of the X-axis Tool coordinate system.

Bilinear Interpolation

Use the **Bilinear Interpolation** check box to enable bilinear interpolated sampling of the region of interest. Bilinear sampling is slower than uninterpolated sampling but should always be enabled when subpixel accuracy is required.

Sampling Step Settings

Sampling refers to the process of sampling the region of interest in the input Image and creating a corresponding Image. By default, before the tool is executed, the input Image is analyzed to compute an optimal sampling step, that is, the horizontal and vertical distances between each pixel in the sampled Image.

- **Set to Default** sets the custom sampling step to the optimal value. This default sampling step is recommended for most applications.
- To override the default value, enable the **Custom** check box and enter a value in the text box.

Transform Panel

Use this panel to set the transform that will be applied to the tool before processing the input Image. The Transform panel is active only when using Model-based positioning.

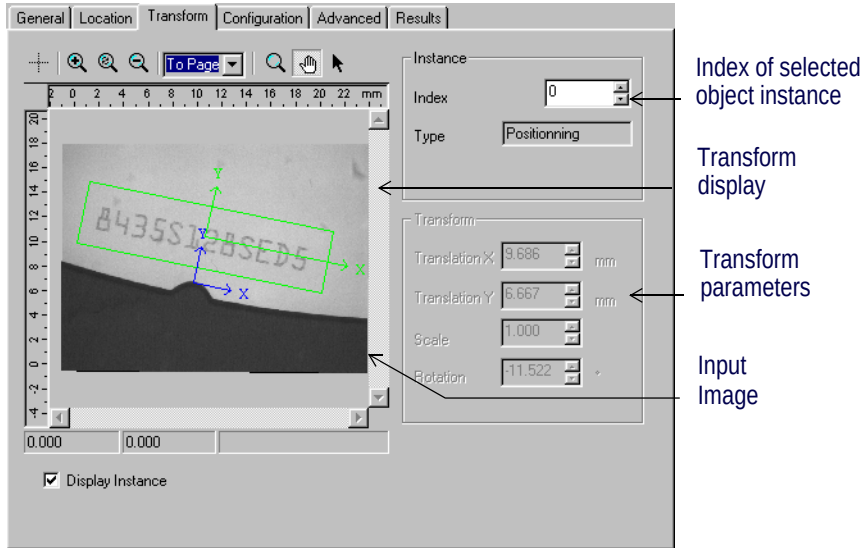


Figure 440 OCR Fixed Font — Transform Panel

Transform Display

In addition to the input Image, the **Transform Display** also shows the Rectangle region of interest, transposed according to the transform parameters.

Display Instance

Use the **Display Instance** check box to show the contours of the selected instance overlaid on the input Image. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

If the image contains more than one object instance, use the Instance **Index** to toggle between instances.

Instance Index

Use **Instance Index** to select an object instance in the input Instance Scene from which the tool transform is retrieved. This parameter is used only when Model-based automatic positioning is enabled, otherwise it is ignored.

Instance Index is zero based, that is, if 3 objects are available in the Instance Scene, it ranges from 0 to 2. When a new instance is selected, the transform parameters are shown in the read only **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes. The name of the Model corresponding to this instance is also shown in the read only **Instance Type** text box.

- Type the index of the instance to be processed in the **Index** text box or use the spin up/down buttons to browse the available instances in the input Instance Scene.
- When a new instance is selected, the tool's region of interest is transposed but the process is not applied to this new region of interest until the tool is executed.
- If the name of the Model corresponding to the selected instance does not match the Model for which the tool is configured, the execution of the tool is still possible. However, results might not be as expected since the instance processed does not match the type of object for which the tool is configured.
- The Transform parameters show the rotation, scale and translation for the selected instance.

Transform Parameters

Transform parameters consist of scale factor, rotation and X-Y translation. When Model-based manual positioning is enabled the **Translation X**, **Translation Y**, **Scale** and **Rotation** text boxes can be used to enter manually the transform used to transpose the tool's region of interest. When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based automatic positioning is enabled, these text boxes are read only and display the transform parameters of the selected instance in the Instance Scene.

- When Model-based manual positioning is enabled, you can manually modify the transform parameters.

Configuration Panel

The OCR Fixed Font tool can read up to four lines of character strings having the same font. The following standard fonts are supported: SEMI, OCR-A, and OCR-B. Each line can have up to 30 characters. When **Rows to Read** in the General tab is set to a positive value (up to 4), the number of characters in each available row in the **Number of Characters** group will be enabled for editing.

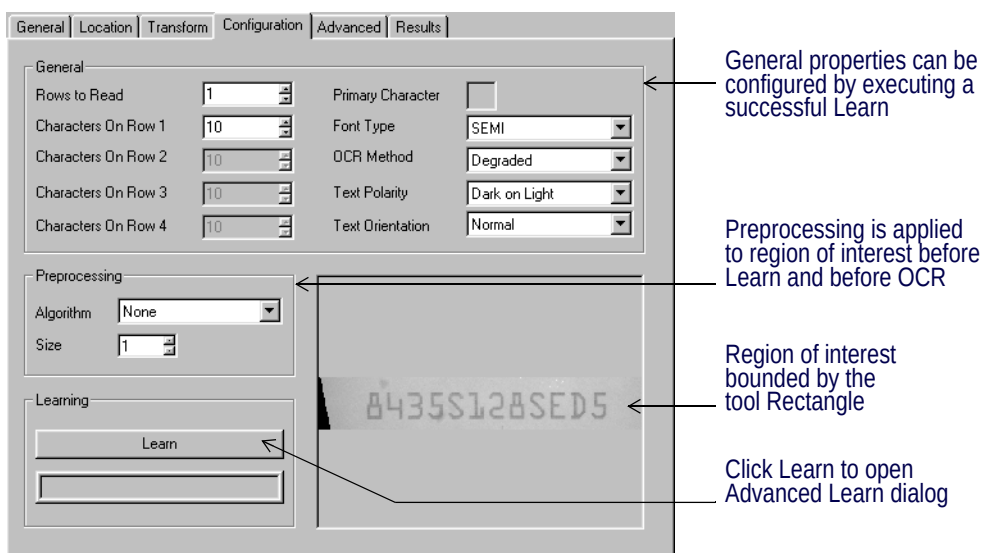


Figure 441 OCR Fixed Font — Configuration Panel

General Properties

General properties can be set manually when you know the values corresponding to the strings of text to read. However, a successful learn will set, or reset the general properties to the learned values.

Rows to Read

This sets the number of rows that need to be read for a single instance.

- The maximum number of row is 4.
- The **Rows to Read** value will enable the editing of the corresponding **Characters on Row** text boxes.

Number of Characters On Row

Use the **Characters On Row** text boxes, to set the number of characters to be read in each row.

- The minimum number of characters per row is 2.
- The maximum number of characters per row is 10.

Primary Character

This is the Primary Character that was selected in the training string. If no Learn has been carried out this box is empty.

Font Type

- Use the combo box to select the font type: **SEMI**, **OCR-A** or **OCR-B**. This property is auto-configured after a successful Learn.

OCR Method

Use the combo box to select the method to be for locating and decoding the characters. The default is **Degraded**.

- **Standard**: This method works on clean characters with no significant background noise. The method allows the text string to rotate up to 15 degrees from the horizontal or vertical orientation. The speed of the method is relatively fast. The number of characters to be read during the run is allowed to vary from the number used in Learn.
- **Dot Matrix**: This method is the Standard method optimized for Dot Matrix characters. It has the similar performance as the Standard method except that it is more tolerant of background noise.
- **Degraded**: This method is the most robust of all three methods and should be used whenever possible. It is capable of reading characters with damages and noise. It allows minor rotation from the horizontal or

vertical orientation. The speed of the OCR reading is slower than the above two methods.

Text Polarity

Use the combo box to select the polarity of the strings to be read:

- **Dark on Light:** Reads light text on dark background.
- **Light on Dark:** Reads dark text on light background.
- **Unknown:** The text polarity can be either **Dark on Light** or **Light on Dark**.

Text Orientation

Use the combo box to select the orientation of the strings to be read:

- **Normal:** Normal text reading order, i.e. left to right across the rows.
- **Top to Bottom:** The text to is read from top to bottom, by columns.
- **Upside Down:** The text is inverted, and is read as normal direction, but upside down.
- **Bottom to Top:** The text to is read from bottom to top, by columns.

Preprocessing

For poor symbology images, a properly chosen preprocessing step may enhance the image quality and reduce the noise to improve the reading robustness. Preprocessing parameters are applied to the image before the reading operation.

By default, no preprocessing **Algorithm** is applied (**None**) is carried out. The **Size** text box sets the parameter for the selected preprocessing algorithm where applicable. Available preprocessing options are:

- **Invert:** Reverses the polarity of the image. For example, changing a light characters on a dark background to dark characters on a light on a light background.

- **Erode**: This option enables you to increase dark areas (dark characters on light background). The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Dilate**: This option enables you to increase light areas (light characters on dark background). The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Open**: This option enables you to remove minor light defects of dark areas. The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Close**: This option enables you to remove minor dark defects of light areas. The **Size** parameter sets the value of the neighborhood applied by the operation. For example, 3 sets a 3X3 neighborhood.
- **Gaussian**: The Gaussian operation acts like a low pass filter and helps remove noise and has overall blurring effect on the image. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Sharpen**: This has the effect of sharpening the image, especially edges. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Sharpen Low**: This has the effect of sharpening and smoothing the image at the same time. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Median**: The Median operation reduces impulsive image noise without degrading edges or smudging intensity gradients. The **Size** parameter sets the value of the kernel applied by the operation. For example, 3 sets a 3X3 kernel.
- **Histogram Stretching**: This operation increases the contrast in the image. The **Size** parameter is ignored.

Learning

To conduct a Learn process, click the Learn command button. This opens the Assisted Learn dialog in which you will define the training string and primary character upon which the Learn will be performed.

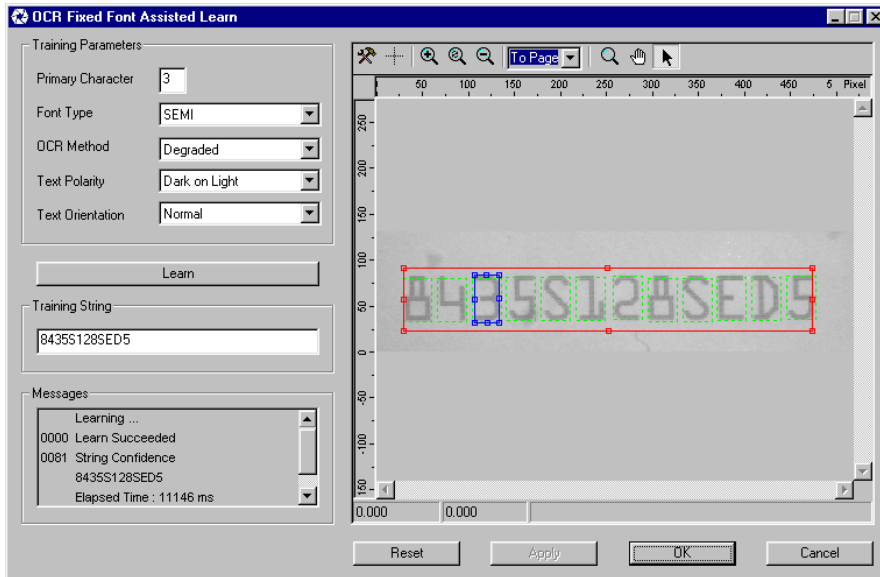


Figure 442 OCR Fixed Font — Assisted Learn Dialog

Procedure for Executing a Learn

In the assisted Learn Dialog:

1. Configure the **Training Parameters** based on the known characteristics of the provided training string.
2. In the display, draw the Training Box and the Primary Character box
3. Type in the actual characters of the Training Sting
4. Click learn to Execute the Learn.
5. The message box provides information on the success or failure of the Learn process.

Training Parameters

You must manually configure the training parameters before carrying out a Learn. When the Learn is successful, the values of the parameters will be configured for the corresponding runtime parameters (See Configuration Panel parameters).

- **Primary Character:** The primary character is a character chosen within the training string. Chose a primary character that reflects the average size and stroke width of the characters in the string.
- **Font Type:** Specifies the type of font used by the tool to decode the characters in the string to read.
- **OCR Method:** Choose the appropriate OCR Method. See the previous section for more information on this parameter
- **Font Type:** Select the font type corresponding to that in the training string.
- **Text Polarity:** Select the polarity corresponding to that in the training string.
- **Text Orientation:** Select the orientation corresponding to that in the training string. Note that the orientation depends on the orientation of the tool's region of Interest (See Location Panel).

Training String

You must enter the actual values of the current training sting in this text box, respecting the case. You must also enter any "empty space" characters that are in the string.

Drawing the Training String And Primary String Boxes

You must draw bounding boxes in the display to define the training string and the primary character.

1. Drag a Training Box (red) to enclose the training string. The left and right sides of the training box *must* tightly enclose the first and last characters of the string; i.e. *do not leave empty space* between the

- bounding box sides and the end characters of the training sting. You can readjust the size of the training box after you have drawn the Primary Character training box (step 3).
2. Drag the Primary Character training box (blue), over a character in the string. This must be the same character as the one entered in the Primary Character text box. The sides of this box must tightly enclose the left and right sides of the selected character.
 3. Click Learn.
 4. If the Learn fails, the Message box provides information on possible causes of the failure. The status code provides additional information on the success or failure of the operation. See in the **Results Panel** section for the complete list of possible status codes.

Advanced Panel

Use this panel to configure additional parameters that will be applied by the tool.

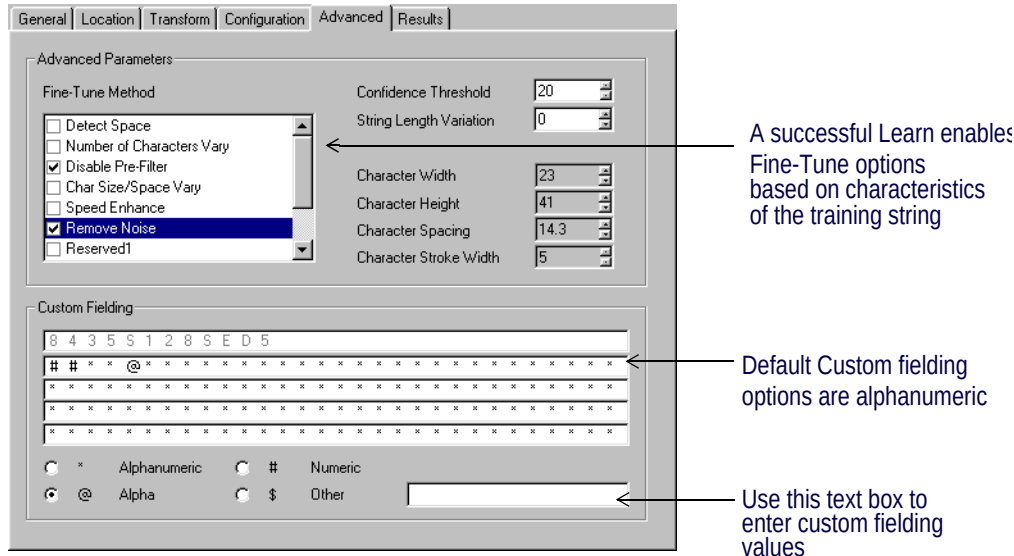


Figure 443 OCR Fixed Font — Advanced Panel

Advanced Parameters

Fine Tune Method

You can enable more than one **Fine-Tune Method** option. A successful Learn optimizes and sets Fine-Tune options based on the training string image. You may want to disable methods set by the Learn and enable others if conditions during runtime vary from the learned training string.

- **Detect Space:** This option must be enabled for the tool to recognize and properly read empty character spaces in the character string. The option is automatically enabled by a Learn when the training string contains at least one empty space.
- **Number Of Characters Vary:** Enables the tool to read strings in which the number of characters varies from the number of characters in the training string or from the values set for **Numbers of Characters On Row**. This option is supported by **Standard** and **Dot Matrix** OCR methods; it is ignored when the **Degraded** OCR method is used.
- **Disable Pre-Filter:** Disables the tool's internal prefilter process to prevent removal of too much image detail.
- **Char Size / Space Vary:** Enables a variation of up to 10%, from the training string values, in character size and spacing between characters. The option is ignored when **Degraded** OCR method is used.
- **Speed Enhance:** This option enables the tool to use an OCR algorithm that is faster but less robust.
- **Remove Noise:** Enable this for reading characters in noisy images. This option is automatically enabled when the **Degraded** method is enabled.
- **SEMI M-Checksum:** For SEMI fonts, this option enables the tool to carry out a checksum to ensure that all characters are correctly decoded.

Confidence Threshold

The tool calculates a confidence value between 0 to 100 for each character in a string.

- A high confidence threshold can ensure that only characters with a high quality of match are read.
- Conversely, a low threshold value is more robust to characters that do not perfectly match the ideal.

String Length Variation

This parameter sets the allowable variation in string length of up to +/- 10% from the training string length. This parameter only applies to **Degraded** OCR Method.

Character Width

This value is configured by a successful learn but can be changed if runtime instance differ from the learned values.

Character Height

This value is configured by a successful learn but can be changed if runtime instance differ from the learned values.

Character Spacing

This value is configured by a successful learn but can be changed if runtime instance differ from the learned values.

Character Stroke Width

This value is configured by a successful learn but can be changed if runtime instance differ from the learned values.

Setting Custom Fielding Options

Custom Fielding options are enabled only after a successful Learn. Assigning custom fielding values to character positions can reduce execution time.

- By default, each character position is designated as **Alphanumeric** and is represented by *****.
- To assign an different option to a character position, click its corresponding **"*"** symbol, then enable the radio button corresponding to the required option: **Alpha**, **Numeric** or **Other**. These will respectively be represented by **@**, **#** or **\$**.
- Use the **Other** option to attribute either a single value or range of values to a character position. Enter the value or range in the text box. For example, if a character is always either 1, 2, 3, or A, enter **123A** into the text box. This option can be used for separators such as **"-"**, **","** or **" "** (empty space).
- If you attribute only single character to a position in the string (**Other** option), the result for the character in this position will always be output as the *assigned character*, with a **Confidence Value** of **100**.

Results Panel

The Results panel shows the results of the OCR Fixed Font process, including a visual display of the sampled Image. Read only text boxes provide dimensional and positional informations on this Image.

	1	2	3
Status Code	0000		
Barcode Type	UPC / EAN		
Angle	-90.00		
Locate / Decode Time (ms)	8		

Elapsed Time: 180.087 ms ☒ Show Corner Coordinates

Coordinate System: Tool

Messages

0000 648234996494
(4.48,16.42) (4.32,-12.34) (-4.79,-12.18) (-4.64,16.42)

Figure 444 OCR Fixed Font — Results Panel

Found Strings

The **Row 1** to **Row 4** text boxes displays the characters read for each row of text decoded by the tool.

Elapsed Time

Elapsed Time shows the total execution time of the OCR Fixed Font tool.

Status Code

The Status codes provide additional information on the success or failure of the OCR process. The possible status codes are listed in the following table.

Table 48 Status Codes in Results Messages

Numeric Code	Description
0000	Success
4601	No Edge Candidate Found
4604	First Edge Not Found or Too Small

Table 48 Status Codes in Results Messages

Numeric Code	Description
4605	Second Edge Not Found
4611	Third Edge Not Found
4612	Fourth Edge Not Found
4620	Symbology Corner Out of Image
4621	Size Test Failed
4622	Unable to Find Row/Column
4633	Border Match Failed
3048	Symbology Decode Failed
4630	Inspection Timed Out
4301	Trigger Overrun
4303	String Mismatch
4304	Angle Match Failed
4305	Excessive Damage
4310	Process Overrun
6001	OCR Conformance Failure
6002	OCR General Error
9999	Dongle Missing

Coordinate System

Use the Coordinate System combo box to select the coordinate system in which the results are output. The available coordinate systems are: **Image**, **World**, **Object**, and **Tool**.

Messages

The **Message** box provides information on the strings and characters read by the tool:

- The **Elapsed Time** which is the time that was required to read the found text string(s).
- The actual text string(s) read by the tool.

- The **String Confidence** value, which correspond to the smallest **Character Confidence** value in the string(s)
- For each character in the string(s): the **Character Confidence** value, the text value of the character, the index of its string (row) and the 4 corners coordinates for the character.
- Corner coordinates for each character in character strings read by the tool. These coordinates are expressed in length units set in the General panel when the selected Coordinate System is **World**, **Object** or **Tool**. Coordinates are expressed in pixels when **Image** is selected. In the exceptional case where the tool is not configured to work with calibrated units, values returned in the Tool coordinate system are expressed in pixels, and values cannot be returned in the World nor the Object coordinate systems.

Class Library

The Class Library contains ActiveX controls that provide an external access to an application's runtime database (HSDatabase, HSView, HSIImage, HSScene, HSArc, HSLine and HSPoint), to error objects (HSErrorObject) and to calibrations (HSCalibration)

The objects of the Class Library are intended to be used programmatically. See the **Reference Manual** for more details on the Class Library.

Class Library Chapter Outline

[Relationship Between Database Objects](#)

[Visual Basic Example](#)

[Visual C++ Example](#)

Relationship Between Database Objects

Seven Class Library objects constitute elements of a HexSight database. This relationship is illustrated in Figure 445.

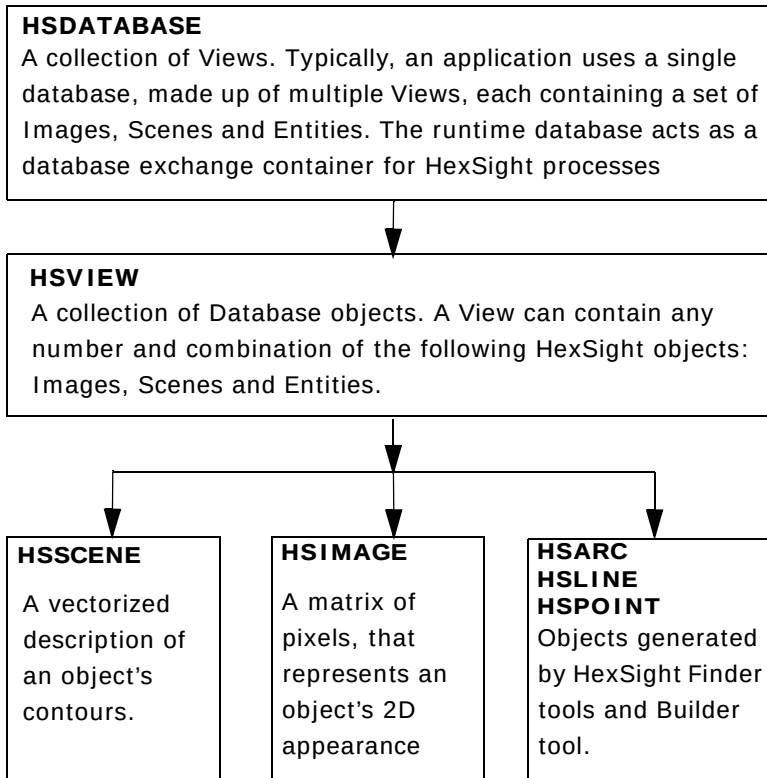


Figure 445 Relationship Between Database-Related Library Controls

Visual Basic Example

This Visual Basic example shows how to create a new View in the Database, add a new image in this View, copy the acquired Image in the new image and then finally, modify the new Image, pixel by pixel.

```

Dim lManager As HSProcessManager
Dim lGSImage As HSImage
Dim lNewImage As HSImage
Dim lNewView As HSView
Dim lDatabase As HSDatabase

Set lManager = HSApplication1.ProcessManager

'Image acquisition is performed
lManager.Execute "Acquisition"

'lDatabase is used to access the
'Database of the HSApplication 'Control
Set lDatabase = HSApplication1.Database
'lGSImage is assigned to the HSImage "Grey-Scale Image"
that
'is contained in the View "Acquisition"
Set lGSImage = lDatabase.Image("Acquisition", _
                                "Grey-Scale Image")

'A new View is added to the active Database
Set lNewView = lDatabase.AddView("New View")
'A new image is added to the View "New View"
Set lNewImage = lDatabase.AddImage( _
    "New View", "New Image", _
    HSDatabase.hsImageType.hsImage32bppRgb)
'The content of lGSImage is copied in lNewImage
lNewImage.Copy lGSImage.RawImagePointer, _
    lGSImage.Width, lGSImage.Height

```

```
'A process is applied to lNewImage
Dim x As Long
Dim y As Long
For y = 0 To lNewImage.Height
    For x = 0 To lNewImage.Width
        'All pixel values are decreased by 50
        lNewImage.Pixel(x, y) = lNewImage.Pixel(x, y) - 50
    Next x
Next y
```

Visual C++ Example

The C++ example shows how to get access to an acquired Image and how to directly process its pixel array.

```

HSProcessManager lManager =
    mApplicationControl.GetProcessManager();
// Image acquisition is performed
lManager.Execute(ColeVariant( "Acquisition" ),
                ColeVariant("Acquisition" ) );

HSDatabase lDBase = mApplicationControl.GetDatabase();
HSImage lGSImage = lDBase.GetImage("Acquisition",
                                   "Grey-Scale Image");
// The image is accessed through a pointer to its pixel
array
char *Ptr = (char *) (lGSImage.GetRawImagePointer());
int Width, Height;
Width = lGSImage.GetWidth();
Height = lGSImage.GetHeight();

// A process is applied to the pixel array int x,y;

for (y=0; y<Height; y++)
{
    for(x=0; x<Width; x++)
    {
        int pos = Width*y + x ;
        // All pixel values are decreased by 50
        *(Ptr+pos) = *(Ptr+pos)-50;
    }
}

```

This example shows that any user-defined processing algorithm can be easily applied to any Image in the runtime database.

Index

Symbols

- .bmp 395
- .cal 458
- .cff file format 18
- .cfg 262
- .dxf 350
- .hac 272
- .hdb 395
- .hig 395
- .hln 272
- .hpt 272
- .hsc 272
- .hvw 272

Numerics

- 1D Linear 798
- 2D Linear 798
- 2D Logarithmic 798

A

- Absolute Clipping Mode 796
- Absolute rotation 345
- Absolute scale 345
- Acquisition Device 381
 - C# tutorial 186, 187
 - Calibration sub-panel 457
 - Configuration panel 390
 - Contour detection 389
 - General panel 390
 - Image calibration 385
 - Image source 382
 - Input sub-panel 394
 - output 389, 478
 - Output sub-panel 478
 - Results panel 479
 - View 478
 - Visual Basic tutorial 52
 - Visual C++ tutorial 122, 123
- Acquisition tab
 - DT-3152 407
 - DT-3157 413

- PX610A 416
- PXC200 423, 452, 453
- Adaptive
 - Contrast Threshold 308
 - High Sensitivity 308
 - Low Sensitivity 308
- Adaptive High Sensitivity 732, 751, 775
- Adaptive Low Sensitivity 732, 751, 775
- Adaptive method 948
- Adaptive Threshold method 924, 948
- Add Process 262
- Adding
 - features to Model 358, 364
 - frame grabbers 397
 - new Models 351
 - reference point to Model 364
 - Reference points to Model 362
 - tools to an application 262, 267
 - virtual frame grabber 398
- Addition, arithmetic operation 796
- Advanced Panel
 - Barcode Reader 906
 - Data Matrix 942
 - OCR Fixed Font 978
- Advanced Parameters 339, 920
 - Barcode Reader 908
 - OCR FixedFont 979
- Advanced setup mode 332, 339
- AGC gain control 426
- Algorithm, preprocessing 905, 938
- Allow Severe Damage 923, 947
- Allow Steep Angle 923, 947
- Alphanumeric Fielding option 960, 981
- Analyze Model 364
- AND logical operation 799
- Angle
 - Arc Finder result 778
 - Line Finder result 755
- Angle Variation 921, 938, 944
- Application Control 259
 - accessing 259
 - available processes 264
 - commands 262
 - Explorer 268
 - general properties 273
 - Process Manager 261
 - Using 261

Applications

- adding a HexSight Control 50, 116, 184
- building 267
- constructing Main Form interface 48
- constructing program interface 112
- creating C# project 182
- creating VB Form 48
- creating VC++ project 112
- Multi-model 282, 286, 294, 295, 304, 307, 324, 365
- saving the configuration 261
- wafer ID 957

Apply Transform to a Model-Defined Location 896, 930, 965

- Arc Finder 763
- Line Finder 741
- Point Finder 722

Arc 786, 792

Arc Caliper 563

- Configuration panel 586
- General panel 576
- Location panel 580
- results 570
- Results panel 593
- tool position 564
- Transform panel 584

Arc Closest To Guideline 773

Arc Closest To Inside 773

Arc Closest To Outside 773

Arc Edge Locator 597

- Configuration panel 620
- General panel 609
- Location panel 613
- Results panel 625
- tool position 598
- Transform panel 617

Arc Finder 266, 759

- General panel 760
- Location panel 765
- Results panel 776
- Transform panel 769

Arc Finder results

- displaying 777
- Fit Quality 779
- Match Quality 779
- messages 779
- selecting coordinate system 777

Arc Must Be Totally Enclosed in Search Area 774

Arc, enclosed in Search Area 774

Area

- constraints 635, 661
- result 668
- Area of interest 484
- Arithmetic operations 795
 - Addition 796
 - clipping 796, 812
 - clipping modes 796
 - constant 811
 - Darkest 797
 - Division 797
 - Lightest 797
 - Multiplication 797
 - parameters 811
 - scale 812
 - Subtraction 797
- Assignment operations 797
 - constant 812
 - Copy 798
 - height 812
 - Initialization 798
 - Inversion 798
 - parameters 812
 - width 812
- Assisted Learn
 - Data Matrix 941
 - OCR Fixed Font tool 976
- Assisted Learn Interface
 - Data Matrix 940
- Automatic
 - Coarseness Selection 680, 694
 - Levels selection, Locator 285
 - Levels selection, Model 356
- Automatic Calibration 389, 474
 - example 476
 - procedure 474
 - when to use 474
- Automatic Levels 356
- Automation properties 273
- Average Contrast
 - Arc Finder result 778
 - Line Finder result 755
 - Point Finder result 734
- Average Element Size 920, 943, 948
- Average filter 800
- Axes marker, Model 363

B

- B 700, 714
- B color value 700, 714
- Barcode Properties 886, 907
- Barcode Reader 885
- Barcode Type 903, 910
- Barcode Types 889
- Barcode types supported 889
- Barcodes to Decode 903
- Barcode-Specific Properties 889, 908
- Based on Outline Level Only 373
- Basic setup mode 332, 339
- Best Arc 773
- Best Line 750
- Bilinear Interpolation 899, 969
 - Data Matrix 933
- Bilinear interpolation 584
 - Arc Edge Locator 617
 - Blob Analyzer 630
 - Caliper 525
 - Edge Locator 552
 - Image Histogram Tool 829
 - Image Sharpness Tool 847
 - Inspection tools 505
 - Pattern Locator 687
 - Sampling Tool 654, 708, 856, 867
- Black
 - tail 820
 - threshold 819
- Black Level, DT-3155 412
- Blank Length, PX610A 418
- Blob
 - Chain Code 636, 660
 - constraints 661
 - Display 635, 659
 - marker 636
 - processing parameters 661
 - results 636
- Blob Analyzer 629, 645
 - adding C# code 221
 - adding Visual Basic code 87
 - adding Visual C++ code 158
 - bilinear interpolation 630
 - Blob results 636
 - C# tutorial 216
 - Configuration panel 657

- configuring 657
- Coordinate system selector 667
- General panel 645
- Location panel 651
- Results panel 666
- Transform panel 655
- Visual Basic tutorial 82
- Visual C++ tutorial 153
- Blob Analyzer results
 - area 668
 - blob count 667
 - description 668
 - elongation 668
 - enabling result categories 667
 - Extrinsic Box 670
 - Extrinsic Inertia 669
 - inertia x-axis 669
 - inertia y-axis 669
 - Intrinsic bounding box 669
 - Intrinsic Inertia 668
 - Perimeter 668
 - position 668
 - principal axes rotation 669
 - sorting 667
- Blob display modes
 - Blob Image 659
 - chain code 660
 - pixel weight image 660
- Blob Image 636
 - clear 648
 - create 648
- Blob properties 636
 - Chain code 642
 - convex perimeter 637
 - Extrinsic moments of inertia 639, 640
 - General properties 637
 - greylevel 643
 - intrinsic moments of inertia 638
 - Perimeter properties 637
 - raw perimeter 637
 - roundness 637
- bmp file format 272
- bmp files
 - importing 395
- Border Integrity 921, 944
- Bottom to Top instance ordering 311
- Bottom to Top text orientation 958
- Bottom to Top, text orientation 974

- Bounding Area, Model 358
- Bright display mode 358, 367
- Brightness
 - PXC200 425
- Broken Border 917
- Build Model 282, 357
- Builder 266
 - Results panel 789, 876, 882
- Builder Mode
 - Table of descriptions 784
- Builder results
 - displaying 790, 876
 - Extrapolated 790
 - Intersected 790
 - Output 0, Output 1 790
 - selecting coordinate system 789, 877, 883
 - Table of outputs and results 792
- Builder Tool 781, 873, 879
 - General Panel 782, 874, 880
 - Parameters panel 783, 875, 881
- Building a Model 355
- BW Detect, PXC200 426

C

- C#
 - adding code to Main Form 184
 - tutorials 181
- cal file format 458
- Calibrated
 - Height, Sampling Tool 872
 - Width, Sampling Tool 871
- Calibrated coordinate system 385, 386
- Calibrated Units
 - Builder 787
- Calibrated units
 - Arc Caliper 581
 - Arc Edge Locator 614
 - Arc Finder 766
 - Barcode Reader 898
 - Blob Analyzer 651
 - Caliper 523
 - Data Matrix Reader 932
 - Edge Locator 550
 - Image Histogram Tool 828
 - Image Sharpness Tool 846
 - Inspection tools 506
 - Line Finder 744

- OCR Fixed Font 968
- Pattern Locator 686
- Point Finder 342, 725
- Sampling Tool 706, 866
- Calibration
 - Automatic 389
 - C# tutorial 241
 - Coordinate System 857
 - Distortion LUT 388, 466
 - Distortion Model 389, 467
 - error 471
 - files 458
 - From Another Configuration 459
 - loading an existing calibration 459
 - Perspective 387, 463
 - portability 385
 - Programmatic 389
 - saving a calibration 459
 - selecting the right method 459
 - setting dot detection 469
 - sub-panel 457
 - Visual Basic tutorial 107
 - Visual C++ tutorial 178
 - X-Y Scale 387, 460
- Calibration methods 386
- Calibration units 386
- Caliper 509
 - adding C# code 213
 - adding Visual Basic code 78
 - adding Visual C++ code 150
 - Configuration panel 528
 - Edge detection 511
 - Edge scoring 513
 - General panel 518
 - Location panel 522
 - results 516
 - Results panel 534
 - Transform panels 525
- Camera Gain, DepictMaster 4x1 445
- Camera Input
 - PX610A 416
 - PXC200 423
- Camera Offset, DepictMaster 4x1 445
- Camera Port
 - DepictMaster 4x1 445
 - PC-DIG 404
 - PCVision 405
- Camera position, calibration

- Perspective 388
- XY scale 387
- Candidate Points 838, 850
 - Automatic setting 838
 - Standard Deviation 838
- Carrying out an Assisted Learn 941
- CCD cameras, recommended 9
- Center 363
 - Arc fit mode 773
- Center of mass, Blob Analyzer 636
- Center Point
 - Arc Finder result 778
 - illustration 779
- Center Point of an Ar 792
- Center Point of an Arc 785
- Center, Model coordinate system 363
- cfg file format 262
- Chain Code
 - Length 642
- Chain Code properties
 - DeltaX, Delta Y 642
- Changes to the List of Enabled Models 304
- Changes to the Range of Rotation 305
- Changes to the Scale Factor Range 305
- Channel
 - Meteor-II/Standard 431
 - Orion 444
- Channel, Meteor-II/Multi-Channel 434
- Char Size / Space Vary 979
- Character Confidence 961
- Character Height 980
- Character Size Space Vary 959
- Character Spacing 980
- Character Stroke Width 980
- Character Width 980
- Characters On Row 973
- Checksum Enabled 909
- Choosing Features for the Model 361
- Chroma Comb Filter, PXC200 427
- Chrominance Filter, DT-3155 412
- Circle 786, 792
- Circular Model 291
- Class Library 985
 - Visual Basic example 987
 - Visual C++ example 989

- Class Wizard, incompatibility 117
- Clear Blob Image 648
- Clear Quality 377
- Clipping Modes 796
- Clock tab, DT-3152 408
- Clock, DT-3157 413
- Close 806
- Close, preprocessing 939, 975
- Closest Arc-Arc 785, 792
- Closest Line-Arc 785, 792
- Closest Line-Line 785, 792
- Closest Point-Arc 785, 792
- Closest Point-Line 785, 792
- Coarseness
 - Detail Level 348
 - Outline Level 347
- Code Height and Length 886
- Color
 - HSL values 700, 714
 - RGB values 700, 714
 - Tolerances 700, 715
- Command buttons, common HexSight 25
- Compute Center 773
- Compute Center and Radius 773
- Compute Radius 773
- Confidence Threshold 980
- Confidence value
 - Effect of custom fielding 981
- Configuration
 - Acquisition Device 390
 - definition 16
 - saving an application 261
- Configuration file
 - export 262
 - import 262
 - load 262
 - save 262
- Configuration Files
 - Meteor-II/Digital 438
 - Meteor-II/Multi-Channel 433
 - Meteor-II/Standard 430
 - Orion 442
 - PC-DIG properties 403
 - PCVision 405
- Configuration Panel 902
 - Acquisition Device 390

- Data Matrix 936
- Inspection tools 508
- OCR FixedFont 972
- Pattern Locator 693, 711
- Configuration panel
 - Arc Caliper 586
 - Arc Edge Locator 620
 - Blob Analyzer 657
 - Caliper 528
 - Edge Locator 555
- Configuration Properties 956
- Conformity Tolerance 315, 323, 375, 752, 775
 - default 374, 752, 775
 - valid range 374, 752, 775
- Connectivity 731
- Constant, as Operand 795
- Contour Detection 306
 - based on Models 347
 - Contrast Threshold 308, 348
 - Detail Level 307, 348
 - Outline Level 307, 347
 - parameters 307
 - Parameters Based on 347
 - settings 347
- Contour Panel
 - Locator 346
- Contour Scene 279
 - Detail 338
 - Outline 338
- Contrast Polarity 313, 321, 372, 373
 - Don't Care 313, 321, 373
 - Normal 313, 321, 373
 - Normal and Reverse 313, 321
 - Reverse 313, 373
- Contrast Threshold 286, 308, 348, 356, 732, 751
 - Arc Finder 774
 - Line Finder 751
 - Point Finder 732
- Contrast, PXC200 424
- Control
 - adding, in C# 184
 - adding, in Visual Basic 50
 - adding, in Visual C++ 116
 - referencing 24
- Controller board, Opteon DepictMaster 444
- Conversion Operations 798
- Conversion operations

- High 812
- Low 812
- parameters 812
- To Intensity Dynamic Range 798
- To Intensity Static Range 798
- Conversions, unit and angle 257
- Convex Perimeter 637, 668
- Coordinate System
 - Barcode Reader 912
 - Data Matrix results 951
 - Locator results 378
 - OCR FixedFont results 983
- Coordinate system
 - Arc Caliper results 595
 - Arc Finder results 777
 - Builder results 789, 877, 883
 - Edge Locator results 561
 - Line Finder results 754
 - Point Finder results 734
- Coordinate System, Model 362
- Coordinate systems 251
 - Arc Edge Locator 627
 - Blob Analyzer 667
 - Image 254
 - Object 255
 - Pattern Locator 696
 - Sampling Tool 870
 - Tool 256, 484
 - World 253
- Copy, assignment operation 798
- Core Function, PXC200 425
- Coreco Imaging
 - multiple frame grabbers 406
- Coreco Imaging frame grabber
 - IC-ASYNCR 399
- Coreco Imaging frame grabbers
 - how to install 12
 - IFC software 13
 - ITEX software 12
- Corner interpolation 731
- Create
 - Blob Image 648
 - Custom Kernel 813
 - Pattern from Input Image 692
- Cropping models 281, 320, 364
- Current Instances 368
- Custom Fielding 960

- purpose 960
- Custom fielding
 - How to set 980
- Custom Fielding Options 960
- Custom Kernel
 - Anchor 814
 - dialog 813
 - Dimensions 814
- Custom Shading Area 364

D

- Damaged matrixes, decoding 923
- Dark
 - pixels, eliminating from Histogram 819
 - segmentation, Blob Analyzer 634, 665
 - Threshold Image, Processing Tool 807
- Dark On Light polarity 918
- Dark on Light polarity 938
- Dark on Light text polarity 958, 974
- Darkest, Aritmetic operation 797
- Data Matrix Reader 915
- Data Matrixes to Decode 937
- Data Storage Area 917
- Data Translation frame grabber
 - DT-3152 406
 - DT-3155 411
 - DT-3157 412
 - how to install 12
- Database
 - building an emulation database 396
 - display objects 271
 - loading Model files 350
 - manage objects 270
 - Models 281
 - saving objects 272
 - selecting specific images in 396
- Database Selection mode 382
- Database, definition 16
- DCAM Device 452
- Debounce
 - Both, PXC200 427
 - Long,PXC200 427
 - PX610A 417
- Decode Left to Right 909
- Default configuration, Acquisition Device 391
- Default Conformity Tolerance 374

- Defective data matrixes, decoding 923
- Degraded Method 957
- Degraded OCR method 957, 973
- Deleting
 - feature from Model 60, 130, 194
 - image from database 395
 - Model features 364
 - model from database 352
 - reference points 362
- Demo Mode
 - how to use 34
 - limitations 34
 - using the tutorials in 35
- Demonstration Mode 34
- Demos
 - Arc Finder 31
 - Arc Finder and Edge Finder 32
 - Automated Model Teaching 31
 - Board Location 29
 - Bracket Inspection 30
 - Builder 31
 - Calibration 30
 - Caliper 32
 - Contour Detection 30
 - Contour Drawing 31
 - Custom Model Editor 31
 - Disambiguation of similar objects 32
 - Finder Tools 31
 - Image Processing 31
 - Line Finder 31
 - Locator 31
 - Model Optimization 31
 - Model-Based Board Location 30
 - Part Sorting 32
 - Pattern Locator 32
 - Point Finder 31
 - Project Examples
 - OEM Generic Locator 32
 - OEM Locator .NET 33
 - OEM Locator DLL 33
 - provided with HexSight 28
 - Pulley location and inspection 32
 - Regulator inspection 32
 - Shading Consistency 31
- DepictMaster 4x1 444
 - Multi-camera acquisitions 446
 - Slave mode 446
- DepictMaster 4x1 properties

- Camera Gain 445
- Camera Offset 445
- Camera Port 445
- Edge 446
- Exposure 445
- Period 446
- Strobe 446
- Timeout 446
- Trigger 445
- Detail Contour Scene 338
- Detail Level 348, 356
 - configuring 348, 356
- Detect Space 959, 961, 979
- Detected Frame Grabber 397
- Detection Parameters 887
- Digital I/O tab, DT-3157 415
- Dilate, Morphological operation 806
- Dilate, preprocessing 939, 975
- Dim display mode, Normal display mode 358, 367
- DirectShow device 448
 - Installing 13, 14
 - system requirements 448
- DirectShow device properties
 - Input Source 448, 451, 452, 453, 454, 455
 - Output Source 448, 452, 455, 456
 - RGB inputs 448
 - Stream Format 449
 - Timeout 448, 453, 456
 - Video Parameters 449
- DirectX 8.0 448
- Disable Pre Filter 959
- Disable Pre-Filter 979
- Disabling Model Disambiguation 282
- Disambiguating features 291
- Disambiguation
 - definition 16
 - demo 32
 - disable/enable 282
 - inter-model 294
 - multi-model 294
- Discrete Cosine Transform (DCT) 798
- Discriminating features 289
- Display
 - C# tutorial 203
 - scroll bars 358, 368
 - toolbar 247
 - using HexSight displays 245

- Visual Basic tutorial 69
- Visual C++ tutorial 140
- Display control
 - adding C# code for 204
 - adding Visual Basic code for 69
 - adding Visual C++ code for 141
- Display Instance 656, 901, 970
 - Arc Caliper 585
 - Arc Edge Locator 618
 - Arc Finder 770
 - Caliper 526
 - Data Matrix 935
 - Edge Locator 553
 - Image Histogram Tool 831
 - Image Sharpness Tool 848
 - Line Finder 747
 - Locator 343
 - Pattern Locator 689
 - Point Finder 728
 - Sampling Tool 709, 868
- Display Model 693, 898, 968
 - Arc Caliper 582
 - Arc Edge Locator 615
 - Arc Finder 766
 - Caliper 524
 - Data Matrix 932
 - Edge Locator 550
 - Image Histogram Tool 828
 - Image Sharpness Tool 846
 - Line Finder 744
 - Locator 342
 - Pattern Locator 686
 - Point Finder 725
 - Sampling Tool 707, 866
- Display modes 338, 358, 367
 - Blob Analyzer 652
 - Locator 338
 - Matched Model 325, 379
 - Transformed Model 325, 379
- Display properties, Explorer 276
- Display screen
 - mode area 249
 - rulers of a 249
 - status bar 250
 - tool area 247
 - using 245
 - viewport 250
 - zoom area 248

Displayed Coarseness, Pattern Locator 697

Distortion Calibration

 procedure 467

 when to use 467

Distortion LUT Calibration 388, 466

 when to use 388, 466

Distortion Model Calibration 389, 467

Division, arithmetic operation 797

Don't Care contrast polarity 313, 373

Dot detection parameters 469

Dot Matrix Method 957

Dot Matrix OCR method 957, 973, 979

Dot targets 472

 description 472

 guidelines for creating 473

Drawing

 Primary Character Box 977

 Training String Box 977

DT-3152 properties

 Black Level 408

 Chrominance Filter 408

 Clamp 410

 Frequency 409

 Input Source 407

 Interlaced 408

 Lines 410

 Pixels 410

 Polarity 409

 Source 409

 Sync Source 407

 Trigger 408

 White Level 408

DT-3152 Setup 407

 Acquisition tab 407

 Clock tab 408

 Field tab 409

DT-3155 properties

 Black Level 412

 Chrominance Filter 412

 Input Source 411

 Sync Source 411

 Timeout 412

 Trigger 412

 White Level 412

DT-3157 properties

 Clock 413

 Exposure 413

- Lines 414
- Pixels 414
- Timeout 413
- Trigger 414
- DT-3157 Setup 412
 - Acquisition parameters 413
 - Digital I/O tab 415
 - Field tab 414
- Duplicate Process 263
- dxl files 350
- Dynamic thresholds 634, 664
- Dynamic Update 378

E

- ECC 000 - ECC140 characteristics 919
- ECC 200 characteristics 919
- ECC Level 918, 938
- Edge Count
 - Arc Edge Locator 627
 - Edge Locator 562
- Edge Detection
 - Arc Finder 774
 - Line Finder 750
 - Point Finder 732
- Edge detection 538
 - Arc Caliper 565, 589
 - Arc Edge Locator 599, 622
 - Caliper 511, 530
 - Edge Locator 556
- Edge Locator 537
 - C# code 238
 - C# tutorial 231
 - Configuration panel 555
 - description of results 562
 - Edge scoring 541
 - General panel 545
 - Location panel 549
 - results 544
 - Results panel 560
 - Transform panel 552
 - Visual Basic code 104
 - Visual Basic tutorial 97
 - Visual C++ code 175
 - Visual C++ tutorial 168
- Edge pair count
 - Arc Caliper 535, 595
- Edge score

- Arc Caliper 568, 594
- Arc Edge Locator 602, 626
- Caliper 513
- Edge Locator 541, 561
- Edge Strength
 - Data matrix 920
- Edges
 - in Rectangle display 529, 556, 588, 621
 - magnitude of 561
- Edit Process 263
- Editing Models 287, 358
- Elapsed Time 328, 379
 - Barcode Reader 912
 - Data Matrix results 950
 - OCR FixedFont 982
- Elapsed time
 - Arc Caliper 595
 - Arc Edge Locator 627
 - Arc Finder 777
 - Blob Analyzer 667
 - Builder 789, 877, 883
 - Caliper 536
 - Edge Locator 562
 - Image HistogramTool 836
 - Image Sharpness Tool 853
 - Line Finder 753
 - Locator 328, 379
 - Pattern Locator 696
 - Point Finder 733
 - Sampling Tool 871
- electronics applications 287, 315, 320
- Elements of a Data Matrix 916
- Elongation 638, 668
- Empty space character 961, 977
- Empty space characters 981
- Emulation database
 - building 396
 - input mode 382
 - selecting specific images in 396
- Enabled models 351
- Enabled models, learning 304
- End Angle 778
- End Point
 - Arc Finder result 778
 - Line Finder result 755
- End Point of a Line 785, 792
- End Point of an Arc 785, 792

- Enhance
 - Intensity 922, 946
 - Position 922, 946
- Entity
 - definition 16, 270
 - file formats 18
- Equalization, Image Processing tool 806
- Erode 806
- Erode, preprocessing 938, 975
- Event, PXC200 427
- Evidence order 372
- Exceptions to Learn process 304
- Excluding
 - artifacts from histogram 820
 - background pixels from histogram 819
- Execute Sequence in Application Control 263
- Exhaustive recognition effort 374
- Explorer
 - display images, scenes, entities 271
 - Display properties 276
 - general options 275
 - modifying the database 270
 - properties 275
 - saving database objects 272
- Explorer interface 268
 - tree view 268
 - view the runtime database 268
- Exporting a configuration 262
- Exposure
 - DepictMaster 4x1 445
 - DT-3157 413
- Express Position and Size using Calibrated Units
 - Arc Finder 766
 - Line Finder 744
 - Point Finder 342, 725
- Express Position, Size and Sampling Step using Calibrated Units 898, 968
 - Arc Caliper 581
 - Arc Edge Locator 614
 - Blob Analyzer 651
 - Caliper 523
 - Edge Locator 550
 - Image Histogram Tool 828
 - Image Sharpness Tool 846
 - Pattern Locator 686
 - Sampling Tool 706, 866
- External Trigger 439
 - Meteor-II/Multi-Channel 435

- Meteor-II/Standard 432
- Orion 444
- PC-DIG 404
- PCVision 406
- Extrapolate Arc 787
- Extrapolate Line 787
- Extrapolation
 - Builder results 790
 - effect on results 787
 - illustration of 788
- Extrapolations
 - displaying Results 790
- Extrinsic Bounding Box 640
 - dimensions 641
 - extents 642
 - extrinsic extents 642
 - rotation 641
- Extrinsic Box
 - position 669
 - results 670
- Extrinsic Extents 642
- Extrinsic Inertia 639
 - moments of inertia 639
 - principal axes 639
 - results 669
 - rotation 640

F

- Fast Fourier Transform (FFT) 798
- Feature extraction
 - conformity tolerance 315
 - positioning 319
- Feature recurrence 299, 301
- Feature selection 287
 - criteria 287
 - Model 287
- Feature Sub-Panel 360
- Features
 - adding manually 358
 - adding, in Model Editor 358
 - criteria for choosing 361
 - deleting from Model 359
 - Location 361
 - Required 361
- Field Length, PX610A 418
- Field tab
 - DT-3152 409

DT-3157 414
Fielding options 980
File Emulation mode 394
File formats
 bmp (Image),.bmp 272
 cal (Calibration) 458
 cfg (Configuration) 262
 hac (Arc entity) 272
 hln (Line entity) 272
 hpt (Point entity) 272
 hsc (Scene) 272
 hvw (View) 272
file formats in HexSight 18
Files, Model database 281
Filling blob holes 661
Filter half-width
 Arc Caliper 589
 Arc Edge Locator 622
 Caliper 513, 530
 Edge Locator 541, 557
Filtering operations 800
 Average 800
 Custom 813
 Gaussian 800
 High Pass 803
 Horizontal Prewitt 801
 Horizontal Sobel 802
 kernel 813
 Laplacian 803
 Median 805
 parameters 813
 scale 813
 Sharpen 804
 SharpenLow 805
 Vertical Prewit 801
 Vertical Sobel 802
Finder Pattern 917
Finder Tools
 Arc 759
 Builder 781, 873, 879
 Line 737
 Point 719
Fine Tune Method 922
 OCR FixedFont 979
Fine-Tune Method
 Data Matrix 942, 946
 OCR FixedFont 959

FireWire camera 448, 452
 Installing 13, 14
 system requirements 448, 452, 454

Fit Mode
 Arc Finder 773

Fit Quality 327
 Arc Finder 779
 Line Finder 755

Fixed Arc 786
 3 points 786
 results 792

Fixed Arc (3 points) 792

Fixed Circle 786
 2 points 786, 792
 3 points 786, 792
 results 792

Fixed Contrast Threshold 308

Fixed Line 785
 2 points 785, 792
 results 792

Fixed Point 784
 results 792

Fixed Value 732, 751, 775

Font Type 973, 977

Format ID 938

Frame grabber
 configuration 383
 DT-3152 406
 DT-3155 411
 DT-3157 412
 Generic 397
 how to install 12
 how to remove 399
 IC-ASYNC 399
 Meteor-II/Multi-Channel 432
 Meteor-II/Standard 429
 Orion 442
 PCVision 403, 404
 PX610A 415
 PXC200 422
 specified 397

Frame Grabber mode, Acquisition Device 382

Frame grabbers, supported in HexSight 8

G

G 700, 714
G color value 700, 714

Gain Control, PXC200 426
Gain, PX610A 419
Gamma Correction, PXC200 426
Gap
 PX610A 421
 PXC200 429
Gaussian filter 800
Gaussian, preprocessing 939, 975
General Panel 893, 927
 Barcode Reader 893
 Data Matrix Reader 927
 OCR Fixed Font 962
General panel
 Acquisition Device 390
 Arc Caliper 576
 Arc Edge Locator 609
 Arc Finder 760
 Blob Analyzer 645
 Builder 782, 874, 880
 Caliper 518
 Edge Locator 545
 Image Histogram Tool 823
 Image Processing Tool 808
 Image Sharpness Tool 841
 Inspection tools 501
 Line Finder 738
 Locator 332
 Pattern Locator 681
 Point Finder 719
 Sampling Tool 701, 860
General Properties 886, 918
 Barcode Reader 903
 Data Matrix 937
 OCR Fixed Font 972
General properties, Blob 637
Generic frame grabber 397
GigE Vision Cameras 449
Global Threshold method 924, 947
Gocator Sensor
 Data Type 455, 456
 Encoder Sync 457
 Gocator IP 455
 Image setting 456
 Parameters 454, 456
 Scan lines 457
 Timeout 456
Grab, IC-ASYN 401

Graphical interface elements 25

Greylevel properties

- maximum 643
- range 644

Greylevel properties, Blob Analyzer 643

- mean 643
- minimum 643
- range 644
- standard deviation 644

Greylevel results

- Image Histogram tool 835

Grey-Scale Image

- displaying in Rectangle 621

Grey-Scale Image output 389

Grid of dots 388

Grid of Results 376

- Barcode Reader 910
- Data Matrix 949

Guideline

- Arc Finder 768
- Arc Search mode 773
- illustration, Arc Finder 769
- illustration, Line Finder 746
- illustration, Point Finder 727
- Line Finder 745
- Point Finder 726

Guideline Offset 727, 746

H

H 700, 714

H, HSL color value 700, 714

hac file format 272

Hard drive, minimum requirements 8

Hard threshold 632, 662

Hardware requirements 8

hdb files

- importing 395

Height

- Data matrix 937

Height and Width 918

Help, online 27, 263

HexSigh

- file formats 18

HexSight

- Acquisition tools 264
- adding a component to the toolbox 23
- command buttons 25

- displays 245
- Example applications 28
- getting started 15
- graphical interface elements 25
- hardware requirements 8
- Image processing tools 265
- Inspection tools 265
- installing 11
- kit contents 8
- Locator tools 266
- online help 263
- referencing a control 24
- software requirements 9
- system requirements 8
- Technical support 27
- terminology 16
- tools 21
- uninstalling 11
- what is 16
- HexSight Control
 - adding in C# 184
 - adding in Visual Basic 50
 - adding in Visual C++ 116
- HexSight file formats 18
- Hidden Locator parameters 340
- hig file format 272
- hig files
 - exporting to 395
 - importing 395
- High Pass filter 803
- Histogram 798
 - Blob Analyzer 631, 658
 - Data 822, 836
 - display
 - Histogram Tool 834
 - display, Blob Analyzer 658
 - Image Histogram Tool 819
 - operation, Image Processing Tool 814
- Histogram operations 806
 - Dark Threshold 807
 - Equalization 806
 - Image Processing Tool 806
 - Light Threshold 807
 - Stretching 806
 - threshold 814
- Histogram Stretching, preprocessing 939, 975
- hln file format 272
- Hole Count 644

- Hole Filling 635, 661
- Horizontal -> Vertical 904
- Horizontal and Vertical Probe Spacing 904
- Horizontal Polarity 422
- Horizontal Prewitt filter 801
- Horizontal search direction 904
- Horizontal Sobel filter 802
- hpt file format 272
- hsc file format 272
- HSL color definition 700, 714
- HSL tolerance 700, 715
- HStart, PXC200 429
- Hue 700, 714
- hvw file format 272
- HWidth, PXC200 429

I

- IC-ASYNC properties
 - Grab 401
 - Output 401
 - Timeout 401
 - Trigger 401
- IC-ASYNC Setup 399
 - Configuration File 400
 - multi-camera application 401
 - Two-channel application example 401
- IEEE 1394
 - system requirements 448, 452
- IEEE 1394 camera
 - Installing 13, 14
- IEEE 1394 camera support 9
- IFC software 13
- Image
 - in runtime database 271
- Image calibration 381, 385
 - errors 471
- Image coordinate system 254, 385
- Image display options
 - Model Editor 358
 - Model Optimizer 367
- Image Distance instance ordering 312
- Image Distance order 372
- Image Histogram Tool 817
 - basic concepts 818
 - Configuration Panel 832
 - General Panel 823

- Histogram Data 836
- Image Subsampling 821, 833
- Location Panel 827
- Pixel data 836
- Positioning Mode 823
- purpose 818
- results 821, 834
- Results panel 834
- Statistics 835
- Tails 820, 833
- Thresholds 819, 832
- Transform Panel 830
- Image Histogram Tools
 - results displays 834
- Image Processing Tool 793
 - Arithmetic operations 795
 - Assignment operations 797
 - Filtering operations 800
 - General Panel 808
 - Histogram operations 806
 - Image types 794
 - Input Image 795
 - Logical operations 799
 - Morphological operations 805
 - Operations 795
 - Output Image 795
 - Processing Panel 810
 - Results Panel 815
- Image processing tools 265
- Image quality, Model 283
- Image segmentation 631, 662
 - dynamic threshold 634
 - hard threshold 632
 - soft threshold 633
 - thresholding 632
- Image Sharpness Tool 837
 - basic concepts 837
 - Configuration Panel 850
 - General Panel 841
 - Location Panel 845
 - Measurement Points 850
 - Positioning Modes 841
 - purpose 837
 - results 840, 851
 - Results panel 851
 - Sharpness Operator 838
 - Transform Panel 847
- Image Style 919, 937, 938

- Image Subsampling
 - Arc Finder 774
 - Line Finder 751
 - Point Finder 732
- Image, definition 17, 269
- Image-based calibrated positioning mode 490
- Image-based positioning mode
 - Arc Caliper results in 574
 - Arc Edge Locator results in 607
- Image-based uncalibrated positioning mode 489
- Imagenation frame grabbers
 - how to install 12
 - PXC200 422
 - PXC610A 415
- Images
 - grey-scale 389
 - saving, as Bitmap files 272
 - selecting in emulation database 396
- Import Pattern from Database 692
- Importing a configuration 262
- Improving Search Performance 319
- Inertia Maximum 668
- Inertia Minimum 668
- Inertia properties
 - Extrinsic 669
 - Intrinsic 668
- Inertia properties, Extrinsic 668
- Initialization 798
- Input 278, 335, 825, 843
 - Inspection tools 501
 - Locator 335
- Input Entities 786, 876, 882
 - Builder Mode 784
 - displaying Results 790
- Input for Locator tool 278
- Input Grey-Scale Image 895, 929, 964
- Input Image
 - Arc Caliper 578
 - Arc Edge Locator 611
 - Arc Finder 762
 - Blob Analyzer 647
 - Builder 783, 875, 881
 - Caliper 520
 - Edge Locator 547
 - Image Processing Tool 809
 - Line Finder 740
 - Pattern Locator 683

- Point Finder 722
- Sampling Tool 703, 862
- Input Instance Scene
 - Arc Caliper 580
 - Arc Edge Locator 613
 - Arc Finder 764
 - Barcode Reader 896
 - Blob Analyzer 650
 - Caliper 522
 - DataMatrix Reader 930
 - Edge Locator 336, 549
 - Image Histogram Tool 826
 - Image Sharpness Tool 844
 - Line Finder 742
 - OCR Fixed Font 966
 - Pattern Locator 684
 - Point Finder 723
 - Sampling Tool 705, 864
- Input Model Scene 896, 930, 965
 - Arc Caliper 579
 - Arc Edge Locator 612
 - Arc Finder 763
 - Blob Analyzer 649
 - Caliper 521
 - Edge Locator 336, 548
 - Image Histogram Tool 826
 - Image Sharpness Tool 843
 - Line Finder 741
 - Pattern Locator 684
 - Point Finder 722
 - Sampling Tool 704, 863
- Input modes
 - database selection 382
 - file emulation 382, 394
 - frame grabber 382, 397
 - generic frame grabber 397
 - selection 394
- Input Source
 - DirectShow device 448, 451, 452, 453, 454
 - DT-3155 411
 - Meteor-II/Digital 439
 - Meteor-II/Multi-Channel 434
- Inputs tab
 - PXC200 427
- Inside segmentation 635, 665
- Inspection tools 265, 481
 - basic concepts 484
 - basic setup of a tool 500

- General panel interface 501
- Location panel interface 503
- shared characteristics 484
- Transform panel interface 506
- units 502
- Installing a frame grabber
 - Coreco Imaging 12
 - Data Translation 12
 - Imagination 12
 - Matrox 13
 - Opteon 13
- Installing HexSight 7, 11
 - before installing 10
- Instance Analyzed, Optimizer 353
- Instance Identification and Model Type 325
- Instance Index 901, 935, 970
 - Arc Caliper 585
 - Arc Edge Locator 619
 - Arc Finder 770
 - Blob Analyzer 656
 - Caliper 527
 - Edge Locator 553
 - Image Histogram Tool 831
 - Image Sharpness Tool 848
 - Line Finder 747
 - Locator 343
 - Pattern Locator 690
 - Point Finder 728
 - Sampling Tool 709, 868
- Instance Ordering 311, 322, 372
- Instance Scene 279, 338
- Instance, definition 17
- Instances
 - identification 325
 - number of 310
- Instances Analyzed 367
- Instances to Find 321, 371, 679, 693
- Instantiate, definition 17
- Instantiated Processes 261
- Intellicam
 - Meteor-II/Digital 438
 - Meteor-II/Multi-Channel 433
 - Meteor-II/Standard 430
 - Orion 442
- Intensity Enhance 922, 946
- Inter-model disambiguation 282, 294
- Interpolate Position 731

Interpolated sampling 856
Intersected
 Builder results 790
Intersection
 effect on results 791
 illustration of effect on Builder results 791
Intersection interpolation 731
Intra-model disambiguation 282
Intrinsic Bounding Box 638
Intrinsic bounding box properties
 extents 639
 rotation 639
Intrinsic box results
 extents 670
 rotation 670
Intrinsic Extents 639
Intrinsic inertia properties
 elongation 638
 Moments of inertia 638
Intrinsic Inertia Results 668
Intrinsic moments of inertia 638
Inversion 798
Invert, preprocessing 938, 974
Itex software 13

K

Kernel size
 Image Processing Tool 813
 Image Sharpness Tool 839
Keyboard Combinations, Model Editor display 359

L

L 715
L, HSL color value 715
Laplacian filter 803
Last Coordinate System 871
Learn and Unlearn Commands 306
Learn Barcode 906
Learn command 306, 354
Learn Data Matrix 937, 940
Learn Exceptions 304
Learn Failed 940
Learn Process 890, 924, 955
Learn process 304, 319
Learn process, effect on performance 319
Learn Region of Interest 942

- Learn time 379
- Learn, OCR FixedFont 976
- Learn, procedure for OCR 976
- Learning 976
- Left to Right instance ordering 311
- Left to Right ordering 372
- Left-handed coordinate system 383
- Length units
 - Arc Caliper 580, 765
 - Arc Edge Locator 613
 - Barcode Reader 897
 - Builder 783
 - Caliper 522
 - Data Matrix Reader 931
 - Edge Locator 339, 549
 - Image Histogram Tool 827
 - Image Sharpness Tool 844
 - Line Finder 743
 - OCR Fixed FontTool 966
 - Pattern Locator 685
 - Point Finder 724
 - Sampling Tool 705, 864
- Levels, Outline and Detail 286, 307
- Light
 - pixels, eliminating from Histogram 819
 - Segmentation, Blobs 634, 665
 - Threshold, Image Processing 807
- Light On Dark polarity 918
- Light on Dark polarity 938
- Light on Dark text polarity 958, 974
- Lightest, Arithmetic operation 797
- Line 785, 792
- Line Closest To Guideline 750
- Line Finder 266, 737
 - Configuration panel 748
 - General panel 738
 - Location panel 743
 - Results panel 753
 - Transform panel 746
- Line Finder results
 - displaying 754
 - Fit Quality 755
 - Match Quality 755
 - messages 756
 - selecting coordinate system 754
- Line With Maximum Negative X Offset 750
- Line With Maximum Positive X Offset 750

- Link to Detail 299, 366
- Loading a configuration 262
- Local threshold 948
- Local Threshold method 924
- Locate All Symmetric Instances 372
- Locate command 364
- Locate Symmetric Instances 314
- Location 361
- Location Display
 - Barcode Reader 897
 - OCR FixedFont 967
- Location display
 - Arc Caliper 580
 - Arc Edge Locator 613
 - Arc Finder 765
 - Blob Analyzer 651
 - Caliper 522
 - Edge Locator 549
 - Image Histogram tool 827
 - Image Sharpness Tool 845
 - Inspection tools 503
 - Line Finder 743
 - Model-based 504
 - of an Image-based tool 504
 - Pattern Locator 685
 - Point Finder 340, 724
 - Sampling Tool 706, 865
- Location feature 361
- Location Panel 897
 - Data Matrix Reader 931
 - OCR FixedFont 967
- Location panel
 - Arc Caliper 580
 - Arc Edge Locator 613
 - Arc Finder 765
 - Blob Analyzer 651
 - Caliper 522
 - Edge Locator 549
 - Image Histogram Tool 827
 - Image Sharpness Tool 845
 - Inspection tools 503
 - Line Finder 743
 - Pattern Locator 685
 - Point Finder 724
 - Sampling Tool 705, 865
- Location property, features 289, 292, 294

Location results, Locator tool 326

Locator 266, 277

- adding C# code 200

- adding Visual C++ code 136

- C# tutorial 182

- Contour detection 306

- Contour panel 346

- Execution time 328

- General panel 332

- General properties 277

- Input 278

- Learn process 304

- Model panel 349

- Models 279

- Models database 350

- Output 278

- Region of interest 303

- results 324

- Results panel 376

- Results Viewer 324

- Search area 303

- Search constraints 309

- Search panel 370

- Transform panel 342

- Visual Basic tutorial 48

- Visual C++ tutorial 112

Locator results 324, 325, 376

- Elapsed time 379

- Fit Quality 377

- Match Quality 377

- messages 379

- Model 377

- Model type 377

- Position 377

- Rotation 377

- Scale 377

- selecting coordinate system 378

- symmetric instances 377

- Symmetry 377

- X-Y position 377

Locator search

- Based on outline level 314

- contrast polarity 313

- number of instances 310

- rotation 310

- scale factor 309

- timeout 311

Lock Location Transform on a Selected Instance 896, 930, 965

- Arc Caliper 579
- Arc Edge Locator 612
- Arc Finder 763
- Blob Analyzer 650
- Caliper 521
- Edge Locator 548
- Image Histogram Tool 826
- Image Sharpness Tool 844
- Line Finder 741
- Pattern Locator 684
- Point Finder 723
- Sampling Tool 704, 810, 864
- Logical operations 799
 - And 799
 - constant 813
 - Nand 799
 - Nor 799
 - Or 799
 - parameters 813
 - Xor 799
- Lookup table 388, 466
- Low-Pass Filter, PXC200 425
- Luma Comb Filter, PXC200 426
- Luma/Chrome tab 425
- Luminance 715

M

- Magnitude
 - Arc Caliper 594
 - Caliper results 535
- Magnitude constraint
 - Arc Caliper 568, 592
 - Arc Edge Locator 603, 624
 - Caliper 513, 533
 - Edge Locator 541, 559
- Magnitude curve
 - Arc Caliper 567
 - Arc Edge Locator 601
 - Caliper 512
 - Edge Locator 529, 540, 556, 588, 622
- Magnitude score
 - Arc Caliper 594
 - Arc Edge Locator 627
 - Caliper results 535
 - Edge Locator 561
- Magnitude threshold
 - Arc Caliper 567, 589
 - Arc Edge Locator 622

Caliper 512, 530
Edge Locator 540, 556
Main Form
 adding basic code 118, 184
 adding basic code to 50
 creating in C# 182
 creating in Visual Basic 48
 creating in Visual C++ 112
Manual Model editing 287
Match 695
 Count 696
 Threshold 680, 694
Match Quality 327
 Arc Finder 779
 Line Finder 755
Matched
 contours 315, 316
 Model 325, 338, 379
Matched Model mode 325
Matrix Orientation 921
Matrox frame grabbers
 how to install 13
 Meteor-II/Digital 437
 Meteor-II/Multi-Channel 432
 Meteor-II/Standard 429
 MIL-Lite software 13
 multiple frame grabbers 439
 Orion 442
Maximum Angle Deviation
 Arc Finder 773
 Line Finder 750
Maximum Bar Width 907
Maximum Code Height 907
Maximum Code Length 907
Maximum Grey Level 643
Maximum Negative X Offset 731
Maximum Number of Bars 907
Maximum Positive X Offset 731
Mean Grey Level 643
Measurement Points 850
Measurement Points Count 852
Median filter 805
Median, preprocessing 939, 975
Memory, required 8
Messages 365, 379
 Arc Finder 779

- Data Matrix results 951
- Line Finder 756
- Model creation 297, 365
- Model Editor 297, 365
- OCR FixedFont Results 983
- Point Finder 734
- Search analysis 329
- Search process 379
- Meteor-II/Digital properties
 - Input Source 439
- Meteor-II/Digital Setup 438
 - Configuration files 438
 - Multi-camera application 440
 - three-channel application example 440
- Meteor-II/Multi-Channel properties
 - Channel 434
 - External Trigger 435
 - Input Source 434
 - Output Source 435
 - Sync Source 435
 - Timeout 435
- Meteor-II/Multi-Channel Setup
 - multi-camera application 436
- Meteor-II/Multi-Channel setup 432
 - Configuration files 433
 - Intellicam 433
 - three-channel application example 436
- Meteor-II/Standard properties
 - Channel 431
 - External Trigger 432
 - Timeout 432
- Meteor-II/Standard setup
 - Configuration files 430
 - multiple frame grabbers 431
- Meter, Image Sharpness 852
- Midway Point 785, 792
- MIL/MIL-Lite software
 - Meteor-II/Digital 438
 - Meteor-II/Multi-Channel 433
 - Meteor-II/Standard 430
 - Orion 442
- Minimum
 - Feature Recurrence 300, 301, 367
- Minimum Arc Percentage 776
- Minimum Bar Width 907
- Minimum Clear Percentage 318, 375
- Minimum Code Height 907

Minimum Code Length 907
Minimum Edge Strength 888, 908, 912, 920, 944
Minimum Feature Recurrence 300, 301, 367
Minimum Line Percentage 752
Minimum Model Percentage 316, 323, 375
Minimum Number of Bars 907
Minimum Percentage of Required Features 318, 375
Minimum Quiet Zone 886, 888, 908
Mirror Image Style 919
Mode
 Builder tool 784, 875, 881, 882
Mode, Builder geometric entities 784, 875, 881, 882
Model 278
 adding features to 358
 adding reference points 362
 automatic levels 356
 building 282
 choosing features for 361
 contrast threshold 356
 copying 352
 creating 354
 creating and editing 282
 Database in Locator 350
 definition 279
 Detail Level 286
 editing 282, 352
 feature selection 287
 importing 350
 management 279
 optimiz 352
 Outline Level 286
 reference points 296
 removing features 359
 type 325
 View 337
Model Based Rotation 345
Model Based Scale 345
Model Considerations Affecting Performance 320
Model Contour Detection 285
Model contour detection
 contrast threshold 356
 detail level 356
 outline level 356
 tracking inertia 357
Model coordinate system 296
 Center 363
 rotation 363

- X-Y position 363
- Model Creation 355
- Model Creation and Edition 282
- Model database
 - importing a Model into 350
 - load 350
 - new 350
 - save 351
 - shared 281
- Model Disambiguation 282, 352, 353
- Model Editor 354
 - adding features in 358
 - adding reference points in 362
 - Bounding Area 358
 - C# tutorial 191
 - Coordinate System 362
 - display 368
 - display modes 358
 - display options 357
 - Features sub-panel 360
 - Messages 364
 - messages 365
 - removing features from 359
 - Visual Basic tutorial 57, 127
- Model Editor commands 364
- Model Editor Messages 297
- Model features
 - deleting 364
 - discriminating 289
 - editing 287
 - stability 288
 - unstable 289
- Model file 281
 - format 281
 - loading 350
- Model Image Considerations 283
- Model optimization
 - enabling 353
 - executing 353
 - feature recurrence 299
 - how it works 299
 - noisy objects 298
 - parameters 299
 - purpose 298, 352
 - variable shapes 298
- Model Optimizer 365
 - display modes 367
 - display options 367

- enabling 353
- Model Panel 349
- Model View 279, 337
- Model(s) Mode threshold 309
- Model(s) Value and Mode threshold 309
- Model(s) Value threshold 309
- Model-based Automatic positioning 496
- Model-based Locator 302
- Model-Based Locator, purpose 302
- Model-Based Locator, requirements 302
- Model-based Manual positioning 491
- Model-based tool
 - Location display 504
 - Transform display 506
- Model-Based Tools 302
- Models 279, 351
 - new 351
 - teaching 282
- Models Command Buttons 351
- Models Database 281, 351
- Models Database Sharing 281
- Modes
 - Dark segmentation 665
 - emulation 382
 - Inside segmentation 665
 - Light segmentation 665
 - operating, Process manager 274
 - Outside segmentation 666
 - positioning, in Inspection tools 501
 - positioning, Inspection tools 501
- Moments of inertia 638, 639
- Morph to Average Shape 300, 366
- Morphing 300
- Morphological operations 805
 - Close 806
 - Dilate 806
 - Erode 806
 - neighborhood 814
 - Open 806
 - parameters 814
- Move Reference Points with Axes 364
- Multi-camera application
 - Meteor-II/Digital 440
 - Meteor-II/Multi-Channel 436
- Multi-channel acquisition 478
- Multi-model applications 282, 286, 294, 295, 304, 307, 324, 365

Multiple

- Acquisition Device Configurations 391

- camera configurations 392

- Coreco Imaging frame grabbers 399

- Locators 281

- Models in applications 282, 286, 294, 295, 304, 307, 324, 365

- Multiple acquisitions 390

- Multiple Barcode Reader Tools 891

- Multiple Barcode Reader tools 891

- Multiple barcodes

- detecting with multiple tools 891

- detecting with single tool 892

- Multiple cameras

- configuring 390

- enabling 390

- Multiple configurations, Acquisition Device 390

- Multiple Data Matrix Reader Tools 926

- Multiple frame grabbers

- configuring 390

- enabling 390

- Multiplication, arithmetic operation 797

- Multi-Resolution Search 678

- Multi-scale applications 320, 323

N

- NAND logical operation 799

- No Quiet Zone Clutter 922

- No Right Timing Border 946

- No Top Timing Border 946

- NOR logical operation 799

- Normal & Reverse contrast polarity 373

- Normal Adaptive Sensitivity 732, 751, 774

- Normal and Reverse contrast polarity 313

- Normal Clipping Mode 796

- Normal contrast polarity 313, 373

- Normal Image style 919

- Normal Search speed 923, 947

- Normal text orientation 974

- Notch Filter, PXC200 426

- Number and Width of Bars 887

- Number Of Characters Vary 959, 979

- Number of Columns 938

- Number of Instances 310

- Number of instances to find 310

- Number of Rows

- Data Matrix 938

Number of Rows and Columns 918
Numbers of Characters On Row 979

O

Object coordinate system 255, 296
Object coordinate system, Model 362
OCR Fixed Font 953
OCR Method 973, 977
OCR Methods 957
Offset
 Arc Finder guideline 768
 Negative X, Search Mode 731
 Positive X, Search Mode 731
Offset, PX610A 420
Online help 27
Open 806
Open, preprocessing 939, 975
Operand Image 795, 809
Operands, types of 794
Operating modes, Process manager 274
Operations, Image Processing 811
Opteon frame grabbers
 DepictMaster 4x1 controller board 444, 449
 Installing 13
Optimization
 object of variable shape 298
 parameters 299
 process, how it works 299
 removal of noise from model 298
 variable-shaped objects 299
Optimize 353
OR logical operation 799
Orientation 919
 Data matrix 938
Orientation units
 Arc Caliper 580, 765
 Arc Edge Locator 613
 Barcode Reader 897
 Blob Analyzer 650
 Builder 783
 Caliper 522
 Data Matrix Reader 931
 Edge Locator 339, 549
 Image Histogram Tool 827
 Image Sharpness Tool 844
 Line Finder 743
 OCR Fixed FontTool 966

- Pattern Locator 685
- Point Finder 724
- Sampling Tool 705, 864
- Orion
 - Configuration files 442
 - Intellicam 442
 - MIL/ MIL-Lite software 442
 - Setup 411, 442
- Orion properties
 - Channel 444
 - External Trigger 444
 - Timeout 444
- Other Fielding option 960
- Other fielding option 981
- Outline and Detail Level Values 286
- Outline and Detail Levels 307
- Outline Contour Scene 338
- Outline Level 347, 356
 - coarseness 347
- Outline vs. Detail Level 307
- Output 278, 337
- Output Entities 788
 - displaying Results 790
- Output Entity
 - Arc Finder 764
 - Line Finder 742
 - Point Finder 723
- Output Grey-Scale Image 965
- Output Image
 - Image Processing Tool 809, 871
 - Sampling Tool 704, 863
- Output Objects in Instance Scene as 338
- Output of Locator tool 278
- Output Source
 - Meteor-II/Multi-Channel 435
- Output sub-panel 478
- Output Symmetric Instances 372
- Output, definition 17
- Output, IC-ASYNC 401
- Outputs
 - Acquisition Device 389
 - Locator 337
- Outputs tab
 - PX610A 420
 - PXC200 428
- Outside segmentation 635, 666

Overdrive Search speed 923
Override Model Image for Pattern 685, 692
Override Output Image 810

P

Pair browser
 Arc Caliper 590
 Caliper 531
Pair count
 Arc Caliper 595
 Caliper 535
Parallel Line 786, 792
Parameters
 Builder 787
 Builder Mode 784
Parameters Based on 347
Parameters panel
 Builder 783, 875, 881
Pattern
 Area 677
 definition 674
 detection 678
 Image 676
 Rectangle 675, 691
 rotation 678
 Search Rectangle 678
Pattern Display 692, 697
Pattern Locator 673
 C# tutorial 224
 Configuration Panel 693, 711
 General Panel 681
 Location Panel 685
 Pattern Area 677
 Pattern definition 674
 Pattern Detection 678
 Pattern Image 676
 Pattern Panel 691
 Pattern Rectangle 675, 691
 Positioning Modes 682
 results 694
 Results Panel 694
 Transform Panel 688
 Visual Basic tutorial 90
 Visual C++ tutorial 161
PC-DIG properties
 Camera Port 404
 External Trigger 404
 Timeout 404

- PC-DIG Setup 403
 - Configuration files 403
- PCVision properties
 - Camera Port 405
 - External Trigger 406
 - Timeout 406
- PCVision Setup 404
 - Configuration File 405
- Peak Filter, PXC200 426
- Percentage of Points to Analyze 300
- Perimeter results 668
 - convex perimeter 668
 - raw perimeter 668
 - roundness 668
- Period 429
 - DepictMaster 4x1 446
 - PXC200 429
- Perpendicular Line 792
- Perspective Calibration 387, 462, 463
 - corner detection mode 464
 - manual mode 465
 - procedure 463
 - when to use 463
- Pick-and-place applications 311
- pipelined acquisition 393
- Pixel
 - Data 822
 - Height 871
 - Width 871
- Pixel Count
 - Histogram 822
 - Image 822
 - Mode 821
- Pixel Weight Image 636, 660
- Placing and Sizing the Rectangle 341, 899, 968
 - Data Matrix 933
- Point
 - displaying Results 734
- Point Closest To Guideline 731
- Point Finder 266, 719
 - Configuration panel 729
 - General panel 719
 - Location panel 724
 - Results panel 733
 - Transform panel 727
- Point Finder results
 - displaying 734

messages 734
selecting coordinate system 734
Point X, Point Y
Point Finder result 734
Polarity 918
Arc Caliper 570, 590
Arc Edge Locator 604, 623
ArcFinder 772
Barcode Reader 909
Caliper 515, 531
Data matrix 938
DT-3157 frame grabber 413
Edge Locator 543, 557
Line Finder 749
Locator 321, 373
OCR FixedFont 977
PX610A frame grabber 421
Strobe 421
Text 958
Polarity along the X axis 730, 749
Polarity from Inside to Outside 772
Polarity, text 974
Position
Arc Caliper 564
Arc Edge Locator 598
Caliper 532
Edge Locator 538
Rectangle 484
Tool region of interest 484
Position constraint
Arc Caliper 569
Arc Edge Locator 603, 624
Caliper 514
Edge Locator 542, 558
Position Enhance 922, 946
Position score
Arc Caliper 594
Arc Edge Locator results 626
Caliper results 535
Edge Locator results 561
Positioning 319
Positioning accuracy
Arc Finder 776
Line Finder 752
Point Finder 733
Positioning Coarseness 679, 694
Positioning Modes 893, 927, 962

- Positioning modes 488
 - Arc Caliper 576
 - Arc Edge Locator 609
 - Arc Finder 761
 - Blob Analyzer 645
 - Caliper 518
 - Edge Locator 548
 - Image Histogram tool 823
 - Image Sharpness tool 841
 - Image-based calibrated 490
 - Image-based uncalibrated 489
 - Inspection tools 501
 - Line Finder 739
 - Model-based Automatic 496
 - Model-based Manual 491
 - Pattern Locator 682
 - Point Finder 720
 - Sampling Tool 701, 860
- Positioning modes summary 497
- Positioning Speed 733, 752
- Positioning speed 319, 324, 376
- Pre-installation of HexSight 10
- Preprocessing 942, 974
 - Barcode Reader 904
 - Data Matrix 938
 - OCR FixedFont 974
- Preprocessing and Image Quality 891, 925, 961
- Presence/absence applications 314, 319
- Primary Character 956, 973, 977
- Primary String Box 977
- Principal Axes 639
 - Rotation 669
- Probe Spacing 887
- Process manager 261
 - operating modes 274
 - properties 274
- Process Name 895, 929
 - Barcode Reader 895
 - OCR Fixed Font 964
- Process name
 - Arc Caliper 578
 - Arc Edge Locator 611
 - Arc Finder 762
 - Blob Analyzer 647
 - Builder 782, 874, 880, 881
 - Caliper 520
 - Edge Locator 547

- Image HistogramTool 825
- Image Processing Tool 808
- Image Sharpness Tool 842
- Line Finder 740
- Locator 334
- Point Finder 721
- Sampling Tool 703, 862
- Processes, instantiated 261
- Processing panel,Image Processing 810
- Processor, required 8
- Programmatic Calibration
 - code example 474
- Programmatic calibration example 476
- Project Examples
 - Arc Edge Locator 32
 - Arc Finder 31
 - Automated Model Teaching 31
 - Board Location 29
 - Bracket Inspection 30
 - Builder 31
 - Calibration Demo 30
 - Contour Detection Demo 30
 - Contour Drawing 31
 - Custom Model Editor 31
 - Disambiguation of similar objects 32
 - Finder Tools 31
 - Image Processing 31
 - Line Finder 31
 - Locator 31
 - Model Optimization 31
 - Model-Based Board Location 30
 - Pad Inspection with Caliper 32
 - Part Sort 32
 - Pattern Locator 32
 - Point Finder 31
 - Project Examples
 - OEM Generic Locator DLL 32
 - OEM Locator .NET 33
 - OEM Locator DLL 33
 - Pulley Location 32
 - Regulator Inspection 32
 - Shading Consistency 31
- Project examples 28
- Projection curve
 - Arc Caliper 567
 - Arc Edge Locator 601
 - Caliper 512
 - displaying 529, 556, 588, 621

- Edge Locator 539
- Projection Image
 - Arc Caliper 566, 588
 - Arc Edge Locator 600, 621
 - Caliper 511, 529, 621
 - Edge Locator 539
 - in Rectangle display 556
- Projection path
 - Arc Caliper 565
 - Arc Edge Locator 599
 - Caliper 511
 - Edge Locator 539
- Properties
 - automation 273
 - Explorer 275
 - Explorer display 276
 - general Application Control 273
 - Location 326
 - Process manager 274
- PX610A properties
 - Blank Length 418
 - Camera Input 416
 - Debounce 417
 - DX 419
 - DY 419
 - Field Length 418
 - Gain 419
 - Gap 421
 - Offset 420
 - Scale X 419
 - Scale Y 419
 - Single Field 418
 - Strobe Lines 421
 - Strobe Period 421
 - Strobe Polarity 421
 - Sync Lines Enabled 421
 - Sync Mode 417
 - Timeout 416
 - Trigger Delay 417
 - Trigger Type 416
 - X0 419
 - Y0 419
- PX610A Setup 416
 - Acquisition tab 416
 - Outputs tab 420
 - typical configurations 420
 - Video tab 417
- PXC200

Inputs tab 427

PXC200 properties

- Brightness 425
- BW Detect 426
- Camera Input 423
- Chroma Comb Filter 427
- Contrast 424
- Core Function 425
- Debounce Both 427
- Debounce Long 427
- Event 427
- Gain Control 426
- Gamma Correction 426
- Gap 429
- HEn 429
- Holdoff Enable 429
- Holdoff Start 429
- Holdoff Width 429
- Low-Pass Filter 425
- Luma Comb Filter 426
- Notch Filter 426
- Peak Filter 426
- Period 429
- Start Line 428
- Strobe 428
- Timeout 423
- Trigger 429
- Trigger State 423
- Trigger use 423
- Video Level 425
- Video Type 424

PXC200 Setup 422

- Acquisition tab 423
- Luma/Chroma tab 425
- Outputs tab 428
- Video tab 424

Q

Quality instance ordering 312

Quality order 372

Quick recognition effort 374

Quiet Zone 917

Quiet Zone Clutter 922, 946

R

R 700, 714

R color value 700, 714

R,G,B images, DirectShow device 448

Radius

- Arc Finder result 778

Radius results

- Arc Caliper 572, 594

- Arc Edge Locator 607, 626

Ratio

- matrix height/width 922

Ratio matrix width/height 945

Raw Perimeter 637, 668

Reading Multiple Bar Codes 891

Reading Multiple Data Matrixes 925

Reattempting a Learn 942

Recognition 373

- conformity tolerance 374, 752

- effort 315

- parameters 315, 373

- positioning 376, 733, 752

- speed 374

Recognition effort 315, 323

- slider 374

Recognition parameters 315

- Arc Finder 775

- effect on performance 323

- Line Finder 751

Rectangle 685

- characteristics 484

- height 485

- Locator 341

- placing, within an Image 505

- position 484

- properties 341

- rotation 485

- selection 630

- sizing 652, 687, 707, 829, 846, 866, 968

- width 485

Rectangle display

- Edge Locator 539

Rectangle Parameters 551, 652, 687, 707, 829, 846, 866, 899, 968

- Data Matrix 933

Recurring features 299

Reference Points 296

- in Model 362

- move with axes 364

Reference Points Sub-Panel 362

Region Interest 886

Region of Interest 886, 916, 954

- Region of interest 484
 - Image Histogram 829
 - Image Sharpening tool 846
 - Locator 341
 - Pattern Locator 678, 685, 687
 - placing 505
 - Sampling Tool 707, 866
 - size 942
- Region of interest characteristics 484
- Region of interest, Image Histogram Tool 819
- Region of interest, Sampling Tool 856
- Relative rotation 345
- Relative scale 345
- Remove Noise 959, 979
- Remove Process 263
- Removing a Frame Grabber 399
- Removing features from the Model 359
- Required feature 361
- Required property 291, 295
- Requirements
 - hardware 8
- Requirements, software 9
- Reset All, Model Optimizer
 - resetting 352, 353
- Reset History, Image Sharpness 852
- Results
 - Arc Caliper 570, 593
 - Arc Edge Locator 605, 625
 - Arc Finder 778
 - Blob Analyzer 636, 666
 - Builder 790, 876
 - Caliper 516, 534
 - Edge Locator 544, 560
 - Image Histogram Tool 821
 - Image SharpnessTool 851
 - Line Finder 754
 - Locator 324, 376
 - Pattern Locator 694
 - Point Finder 734
 - Sampling Tool 857
- Results display
 - Arc Caliper 595
 - Arc Edge Locator 620
 - Arc Finder 777
 - Blob Analyzer 666
 - Builder 790, 876
 - Caliper 536

- Edge Locator 561
- Image Histogram Tool 834
- Image Sharpness Tool 853
- Line Finder 754
- Pattern Locator 697
- Point Finder 734
- Sampling Tool 716, 870
- Results Panel
 - Barcode Reader 909
 - Data Matrix 948
 - OCR FixedFont 981
- Results panel
 - Acquisition Device 479
 - Arc Caliper 593
 - Arc Edge Locator 625
 - Arc Finder 776
 - Builder 789, 876, 882
 - Caliper 534
 - Edge Locator 560
 - Image HistogramTool 834
 - Image Processing Tool 815
 - Image Sharpness Tool 851
 - Inspection tools 508
 - Line Finder 753
 - Locator 376
 - Pattern Locator 694
 - Point Finder 733
 - Sampling Tool 715, 870
- Results Viewer 324, 378
 - display modes 379
 - using 378
- Reverse contrast polarity 313, 373
- RGB color definition 700, 714
- Right to Left instance ordering 311
- Right to Left ordering 372
- Right-handed coordinate system 383
- ROI 484
 - characteristics 484
 - placing, within an Image 505
 - position 484
 - rotation 485
 - size 485
- Rotation 310, 321, 363, 371
 - Arc Caliper 572
 - Arc Edge Locator 626
 - Edge Locator 561
 - Inspection tools 485
 - Locator 310, 371

- pattern 678
- Pattern Locator 594, 695
- Sector, arc-tools 488
- Rotation Arc
 - Caliper 594
- Rotation, Absolute 345
- Rotation, Model-Based 345
- Rotation,Relative 345
- Roundness 637, 668
- Rows to Read 972
- Rulers, in displays 249
- Running in Demo Mode 34
- Runtime database
 - display images, scenes, entities 271
 - managing objects 270
 - saving objects 272

S

- S 715
- S_Video, PXC200 426
- S, HSL color value 715
- Sample applications 28
- Sample pattern, Pattern Locator 674
- Sampled Image properties 821, 840, 858
- Samples Per Cell 921, 944, 948
- Sampling 498
 - Bilinear interpolation 498
 - interpolated 856
 - Sampling Step 499
- Sampling Step Settings 900, 969
- Sampling Tool 699, 855
 - basics 856
 - General panel 701, 860
 - Location panel 705, 865
 - results 857
 - Results panel 715, 870
 - Transform panel 708, 867
- Sampling-step 499
- Sampling-step settings
 - Arc Caliper 584
 - Arc Edge Locator 617
 - Blob Analyzer 655
 - Caliper 525
 - Edge Locator 552
 - Image Sharpness Tool 847
 - Inspection tools 506
 - Pattern Locator 688

- Sampling Tool 708, 867
- Sampling-step settings Image Histogram Tool 829
- Saturation 715
- Saving
 - application configuration 262
 - arc entities 272
 - line entities 272
 - point entities 272
 - Scenes 272
 - Views 272
- Scale
 - Image Sharpness 852
- Scale factor 309, 371
 - constraints affecting performance 320
- Scale X and Scale Y,PX610A 419
- Scale, Absolute 345
- Scale, Model-Based 345
- Scale,Relative 345
- Scene
 - definition 18, 270
 - in runtime database 271
- Score threshold
 - Arc Caliper 570, 593
 - Arc Edge Locator 604, 625
 - Caliper 515, 534
 - Edge Locator 543, 560
- Scoring
 - Arc Caliper 591
 - Arc Edge Locator 623, 626
 - Caliper 513, 532
 - Edge Locator 558, 561
- Scroll bars 358, 368
- Search
 - Based on Outline Only 322
- Search Area 303, 306, 346
 - Arc Finder 767
 - illustration, Arc Finder 767
 - Line Finder 745
 - placing and sizing 341
 - Point Finder 726
 - positioning the Arc Finder 768
 - positioning the Line Finder 745
 - positioning the Point Finder 727
- Search area
 - Locator 341
- Search Coarseness 679, 694
- Search Constraints

- Arc Finder 772
- Line Finder 749
- Point Finder 730
- Search constraints 309, 370
 - based on outline level only 373
 - contrast polarity 372, 373
 - Locator 309, 370
 - number of instances 371
 - polarity 373
 - rotation 371
 - scale 371
 - symmetric instances 372
 - timeout 372
- Search Constraints Affecting Performance 320
- Search Direction 887, 904, 923
- Search Direction, Data Matrix 947
- Search Display, Pattern Locator 697
- Search panel 370
- Search Process 302
- Search Result Messages 329
- Search Speed 923
- Search speed 281, 309, 320
- Search Speed, Data Matrix 947
- Search time 379
- Sector
 - characteristics 487
 - configuration parameters 582, 615
 - display 587, 620
 - how to place 582, 615
 - opening 488
 - placing Arc Edge Locator 615
 - position 487
 - radius 488
 - rotation 488
 - sizing Arc Caliper 582
 - sizing Arc Edge Locator 615
 - thickness 488
- Sector display
 - Arc Edge Locator 627
- Selection of efficient patterns 677
- SEMI M-Checksum 979
- Semi M-Checksum 960
- Semiconductor applications 287, 315, 320
- Sensitivity
 - contrast thresholds 308
 - locator contrast threshold 286
 - Model contrast threshold 356

Sequential acquisition, multiple cameras 392

Setting

 Custom Fielding 980

Setting the Search Area 341

Shading Consistency

 demo 31

Shading Consistency instance ordering 312

Shading Consistency result 372

Sharpen filter 804

Sharpen Low, preprocessing 939, 975

Sharpen, preprocessing 975

SharpenLow filter 805

Sharpness 853

 blurred images 839

 blurriness 838

 Factor 840

 high frequency textures 839

 History 840

 kernel 851

 Operator 838, 851

 Peak 840

 sharp images 839

Show

 Coordinate System 379

 Real-Time Display 399

 Results Viewer 378

 Search Area 379

Show Corner Coordinates

 Barcode Reader 912

 Data Matrix 951

Show options

 Model Editor display 357

 Model Optimizer display 367

Single Barcode Reader Tool 892

Single Barcode Reader tool 892

Single Data Matrix Reader Tool 926

Single Field, PX610A 418

Size

 Arc Caliper 594

 Caliper 535

Size Variation 921, 945

Skew 486

Slow Warp method 924, 948

Soft threshold 663

Software requirements 9

Solid Border 917

- Sony Intelligent Cameras 450
- Sort By 667
- Sort Results 667
 - Arc Edge Locator 625
 - Blob 667
 - Edge Locator 560
- Speed Enhance 959, 979
- Speed/accuracy tradeoff 319
- Stable features 288
 - effect on performance 308
- Standard Deviation Grey Level 644
- Standard Deviation Threshold 838, 850
- Standard Method 957
- Standard OCR method 957, 973, 979
- Start Angle
 - Arc Finder result 778
- Start Line, PXC200 428
- Start Point
 - Line Finder result 755
- Start Point of a Line 785, 792
- Start Point of an Arc 785, 792
- Start/Stop Enabled 909
- Statistics
 - Image Histogram Tool 821, 835
- Status bar 250
- Status Code
 - Barcode Reader 910
 - OCR FixedFont results 982
- Stream Format, Direct Show device 449
- Stretching 806
- String Length Variation 980
- Strobe
 - DepictMaster 4x1 446
 - PXC200 428
- Strobe Lines mode, PX610A 421
- Strobe periods, PX610A 421
- Strong features, used for optimization 299
- Subsampling
 - Arc Finder 774
 - Line Finder 751
 - Point Finder 732
- Subsampling, Histogram image 821
- Subtraction, arithmetic operation 797
- Symmetric instances 314, 322, 372
- Symmetries 291
- Symmetry

- effect on performance 322
- in Models 291
- patterns 677
- results 327

Symmetry of 327

Sync Mode, PX610A 417

Sync Source

- DT-3155 411
- Meteor-II/Multi-Channel 435

System requirements 8

T

Tails, Image Histogram Tool 820

Target position, calibration

- Perspective 388
- XY scale 387

Target, creating 473

Teaching a Model 282, 355

Technical support 27

- e-mail 27
- e-mail queries 27
- telephone access 27

Test applications

- C# 202
- Visual Basic 68
- Visual C++ 139

Text Orientation 958, 974, 977

- illustration 958

Text Polarity 958, 974, 977

Three-channel application 436, 440

Threshold method, Data Matrix 947

Threshold modes, Data Matrix 924

Thresholds 662

- Black 819, 832
- Blob Analyzer 632
- displaying 529, 556, 588, 621
- dynamic 634, 664
- hard 632, 662
- Image Histogram Tool 819, 832
- magnitude
- Caliper 512
- soft 633, 663
- Standard Deviation 838, 850
- White 819, 832

Time, Locator results 328

Time-critical applications 311, 314, 322

Timeout 311, 322, 372, 408

- Barcode Reader 904
- Data Matrix 945
- DepictMaster 4x1 446
- DirectShow device 448, 453
- DT-3155 412
- DT-3157 413
- IC-ASYNC 401
- Meteor-II/Multi-Channel 435
- Meteor-II/Standard 432
- Orion 444
- PC-DIG 404
- PCVision 406
- PX610A 416
- PXC200 423
- Timing Border 946
- Timing Borders 923
- Tolerance, color 700, 715
- Tool coordinate system 256
 - Inspection tools 484
- Tool orientation 485
- Tool positioning modes 488
- Tool Rectangle
 - Data Matrix 942
- Tool region of interest 484
 - rotation 485
- Tool rotation 485
- Toolbar, HexSight displays 247
- Tools
 - Acquisition Device 264, 381
 - Arc Caliper 563
 - Arc Edge Locator 597
 - Blob Analyzer 645
 - Caliper 509
 - Edge Locator 537
 - Image Histogram 817
 - Image Processing 793
 - Image processing 265
 - Image Sharpness 837
 - Inspection 265, 481
 - Locator 266, 277
 - Pattern Locator 673
 - Sampling 699, 855
- Tools, Display window 247
- Top to Bottom instance ordering 311
- Top to Bottom ordering 372
- Top to Bottom text orientation 958, 974
- Topological results 644

- Tracking Inertia 286, 357
- Training Parameters 955, 977
- Training parameters, OCR 976
- Training String 956, 977
- Training String Box 977
- Transform
 - Discrete Cosine 812
 - Fast Fourier 812
- Transform Display
 - Barcode Reader 901
 - Data Matrix 935
 - OCR Fixed Font 970
- Transform display
 - Arc Caliper 584
 - Arc Edge Locator 618
 - Arc Finder 770
 - Blob Analyzer 656
 - Caliper 526
 - Edge Locator 553
 - Image Histogram tool 830
 - Image Sharpness Tool 848
 - Line Finder 747
 - Locator 343
 - Pattern Locator 689
 - Point Finder 728
 - Sampling Tool 709, 868
- Transform mode
 - Arc Caliper 579
 - Arc Edge Locator 612
 - Blob Analyzer 649
 - Caliper 521
 - Edge Locator 548
 - Image HistogramTool 826
 - Image Sharpness Tool 843
 - Pattern Locator 684
 - Sampling Tool 704, 863
- Transform Operations 798
- Transform operations
 - parameters 812
- Transform output
 - 1D Linear 812
 - 2D Linear 812
 - 2D Logarithmic 812
 - Histogram 812
- Transform Panel 900
 - Barcode Reader 900
 - Data Matrix 934
 - OCR FixedFont 969

Transform panel
 Arc Caliper 584
 Arc Edge Locator 617
 Arc Finder 769
 Blob Analyzer 655
 Caliper 525
 Edge Locator 552
 Image Histogram Tool 830
 Image Sharpness Tool 847
 Inspection tools 506
 Line Finder 746
 Locator 342
 Pattern Locator 688
 Point Finder 727
 Sampling Tool 708, 867
Transform Parameters 902
 Data Matrix 936
 OCR FixedFont 971
Transform parameters
 Arc Caliper 586
 Arc Edge Locator 619
 Arc Finder 771
 Blob Analyzer 657
 Caliper 527
 Edge Locator 554
 Image Histogram Tool 832
 Image Sharpness Tool 849
 Line Finder 748
 Locator 344
 Pattern Locator 690
 Point Finder 729
 Sampling Tool 710, 869
Transformed Model 339, 379
Transformed Model Mode 325
Transformed Model mode 325
Trigger
 DT-3152 408
 DT-3155 412
 DT-3157 414
 IC-ASYNC 401
 PXC200 429
Trigger Delay, PXC10A 417
Trigger Mode
 DepictMaster 4x1 445
 External A 445
 Master 445
 Slave 445
Trigger State, PXC200 423

- Trigger Type, PX610A 416
- Trigger use, PXC200 423
- Troubleshooting 330
- Turbo Search speed 923, 947
- Tutorial
 - C# 181
 - Visual Basic 47
 - Visual C++ 111
 - Tutorial, C#
 - Acquisition Device 186
 - Blob Analyzer 216
 - Calibration 241
 - Caliper 207
 - Display 203
 - Edge Locator 231
 - Locator 182
 - Pattern Locator 224
 - viewing Scenes and Images 189
 - Tutorial, Visual Basic
 - Acquisition Device 52
 - Application Control Interface 52
 - Blob Analyzer 82
 - Calibration 107
 - Caliper 73
 - Display 69
 - Edge Locator 97
 - Locator 48
 - Pattern Locator 90
 - testing the application 68
 - viewing Scenes and Images 54
 - Tutorial, Visual C#
 - testing the application 202
 - Tutorial, Visual C++
 - Acquisition Device 122
 - Blob Analyzer 153
 - Calibration 178
 - Caliper 144
 - Display 140
 - Edge Locator 168
 - Locator 112
 - Pattern Locator 161
 - testing the application 139
 - viewing Scenes and Images 124

U

- Understanding
 - Barcode Reader Concepts 886
 - Data Matrix Reader Concepts 916

Understanding Barcode Reader Concepts 886
Understanding Data Matrix Reader Concepts 916
Understanding OCR Fixed Font Concepts 954
Uninstalling HexSight 11
Unit and angle conversions 257
Units 339
 Arc Caliper 580, 765
 Arc Edge Locator 613
 Barcode Reader 897
 Blob Analyzer 650
 Builder 783
 Caliper 522
 Data Matrix Reader 931
 Edge Locator 549
 Image Histogram Tool 827
 Image Sharpness Tool 844
 Inspection tools 502
 Line Finder 743
 Locator 339
 OCR Fixed Font 966
 Pattern Locator 685
 Point Finder 724
 Sampling Tool 705, 864
Unknown text polarity 958
Unlearn command 306, 354
Unstable features, Model 289
Upside Down text orientation 974
USB Camera support 9
USB camera
 Installing 13, 14
Use Default Conformity Tolerance 374
Use External Database 351
Using the Barcode Reader 893
Using the Data Matrix Reader 927
Using the OCR Fixed Font 962

V

Valid Conformity Tolerance Range 374, 752, 775
Variation
 Data matrix angle 938
 matrix angle 944
 Matrix size 945
Vector Point
 illustration 756
 Line Finder result 755
Vector Point of a Line 792
Vertical -> Horizontal 904

- Vertical Polarity 422
- Vertical Prewitt filter 801
- Vertical search direction 904
- Vertical Sobel filter 802
- Video Level, PXC200 425
- Video Parameters, DirectShow device 449
- Video tab
 - PX610A 417
 - PXC200 424
- Video Type, PXC200 424
- View runtime database 270
- View, definition 18, 269
- Viewer
 - Image Histogram 834
 - Locator results 324, 378
 - Sharpness history 851
 - Sharpness history reset 852
- Viewport 250
- Visual Basic
 - adding Application Control 259
 - adding code to Main Form 50
 - Class Library example 987
 - tutorials 47, 111
- Visual C++
 - adding Application Control 260
 - adding code to Main Form 118
 - Class Library example 989
 - tutorials 111

W

- Wafer ID applications 957
- Warp Method 924, 947, 948
- White
 - tail 820
 - threshold 819
- White Level 412
 - DT-3152 408
 - DT-3155 412
- Width
 - Data matrix 938
- Width/Height Ratio 922, 945
- World coordinate system 253, 385
- World Distance instance ordering 312
- World Distance order 372
- World error 471
- World vs. Image 462, 466

Wrappers

- adding, HexSight 117
- Class Wizard 117

X

- X and Y Position, Model Editor 363
- XOR logical operation 799
- XY Scale Calibration 459
 - procedure 460
 - when to use 459

Z

- Zoom area 248





www.LMItechnologies.com

©2009 LMI Technologies Inc.
Errors and omissions subject to change without notice.
HexSight is a registered trademark of LMI Technologies Inc.
All other names are property of their respective owners.