

Visionscape Tools Reference Manual

v9.0.1, June 2018

Copyright ©2018
Omron Microscan Systems, Inc.
Tel: +1.425.226.5700 / 800.762.1149
Fax: +1.425.226.8250

All rights reserved. The information contained herein is proprietary and is provided solely for the purpose of allowing customers to operate and/or service Omron Microscan-manufactured equipment and is not to be released, reproduced, or used for any other purpose without written permission of Omron Microscan.

Throughout this manual, trademarked names might be used. We state herein that we are using the names to the benefit of the trademark owner, with no intention of infringement.

Disclaimer

The information and specifications described in this manual are subject to change without notice.

Latest Manual Version

For the latest version of this manual, see the Download Center on our web site at:
www.microscan.com.

Technical Support

For technical support, e-mail: helpdesk@microscan.com.

Warranty

For current warranty information, see: www.microscan.com/warranty.

Omron Microscan Systems, Inc.

United States Corporate Headquarters
+1.425.226.5700 / 800.762.1149

United States Northeast Technology Center
+1.603.598.8400 / 800.468.9503

European Headquarters
+31.172.423360

Asia Pacific Headquarters
+65.6846.1214

Contents

PREFACE

Welcome xiii

Purpose of This Manual xiii

Manual Conventions xiii

CHAPTER 1

Advanced Visionscape Concepts 1-1

Introduction 1-2

Acquisition Setup 1-3

 Single Camera 1-3

 Single Camera Pipeline Processing 1-3

 Multiple Cameras 1-10

Producing Reports 1-15

 Fail/Pass Results Setup 1-15

 Inspection Execution Timing 1-16

Non-Triggered Image Acquisition 1-17

 Triggering Acquisitions Internally 1-17

Advanced Triggering Techniques 1-18

 Mixing Acquisitions Within a Single Inspection 1-18

Viewing Timing Signals from Running Set of Inspections 1-21

System Resource Considerations for Pipeline Operation 1-22

 Image Buffer Allocation 1-22

 Overrun Conditions 1-23

Introduction to Trajectory Inspections 1-25

Buttons on the Properties Page 1-28

Resource Datums 1-29

Input Datums 1-30

Output Datums 1-31

Contents

References	1-32
Acquire Step	1-32
FlexArray Step	1-42
Formatted Output Step	1-44
If Step	1-54
Inspection Step	1-55
Job Step	1-65
Loop Step	1-66
Sequence Step	1-68
Snapshot Step	1-70
Trajectory Step	1-77
Trajectory Grid Setup Step	1-80
VarAssign Step	1-83
Vision System Step	1-86
Wait Step	1-91
With Step	1-92
Color Channel Selection for Vision Tool Steps	1-94
Color Channel Selection Examples	1-101
Green Interpolation Color Channel	1-103

CHAPTER 2

Calibration 2-1

Calibration Tree Hierarchy	2-4
Calibration Procedure	2-5
Using Robust Calibration	2-7
Using Previously Saved Calibration Data	2-13
Using Calibration in the Result Report	2-16
Multiple Camera Calibration	2-17
Calibration Target Design Criteria	2-19

CHAPTER 3

Masking 3-1

Introduction	3-1
Masking	3-2
Static Masks	3-3
Dynamic Masks	3-3
Generating Masks	3-5
Static Mask Tool	3-5
OCV Tool	3-5
Template Find	3-7
DynamMask Tool	3-7
Using Masks	3-9
Use Input Mask	3-9

- Custom Mask Templates 3-11
- Sample Applications 3-13
 - Simple Mark/Package Inspection 3-13
 - Static Masks From Different Buffers 3-15
- References 3-17
 - AutoThreshold 3-17
 - Compute Polarity 3-19
 - DynamMask Tool 3-21
 - StaticMask Tool 3-27

CHAPTER 4 Dynamic Location 4-1

- Point Locators 4-3
 - Center Point 4-3
 - Rotation 4-3
 - Reference Location 4-4
 - Location Adjustment 4-4
- Building a Job 4-5
 - Inserting a RectWarp 4-6
- Setting Up a Job 4-7
 - Internal vs. External Connections 4-7
 - Positioning of ROIs 4-8
 - Creating Reference Positions 4-10
- Running a Job 4-11
 - Calculation of Location Adjustment 4-11
 - Movement of ROIs 4-11
- Advanced Configurations 4-12
 - Input Point Selection 4-12
 - Adding and Removing Steps 4-14
 - Movement of Integrated Find Steps After Training 4-14
 - Movement of Controlled ROIs After Training 4-14
- Sample Applications 4-17
 - Using Corner Points for Location 4-17
 - Connecting to OCV Autofind 4-18
 - Dynamic Location of an AnnularUnwrap Warp 4-20
- Reference Frames 4-24
 - Reference Frames of Points 4-24
 - Reference Frames of the Locator 4-26
- Results 4-30
 - CenterPt 4-30
 - Offset Distance 4-30
- Edit Window 4-31
 - Status Bar 4-31
- Reference 4-33
 - NPt Locator Tool 4-33

CHAPTER 5 Dynamic Thresholding 5-1

- Average Gray Value 5-2
- Threshold Reference Value 5-3
- Threshold Adjustment 5-3
- Building a Job 5-3
 - Inserting Steps With Thresholds 5-4
- Setting Up a Job 5-5
 - Dynamic Threshold ROI 5-5
 - Controlled Step's ROIs 5-5
 - Normal Training of Reference Value 5-5
 - Manually Setting Reference Value 5-5
- Running a Job 5-6
 - Integrated GrayHistogram Tool 5-6
 - Calculation of the Threshold Adjustment 5-6
 - Modification of Threshold Values 5-7
 - Exception for Thresholds 0 and 255 5-7
 - Clamping 5-7
- Advanced Configurations 5-8
 - Reconnecting Input Reference 5-8
 - Changing Thresholds of Controlled Steps 5-8
 - Connecting Steps Using Step List 5-9
- Sample Applications 5-10
 - Adding Dynamic Thresholding to an Existing Step 5-10
 - Customizing Threshold Adjustment With an Expression 5-10
 - Controlling Execution When Thresholds are Clamped 5-11
- Results 5-13
 - Average Gray Value 5-13
 - Threshold Adjustment 5-13
- Edit Window 5-14
 - Status Bar 5-14
- Reference 5-15
 - DynamThresh Tool 5-15

CHAPTER 6 I/O 6-1

- Introduction 6-1
- General Purpose I/O 6-2
- Sensors And Strobes 6-3
- Virtual Software I/O 6-3
- Network Access 6-4

- I/O In Visionscape Steps and Tools 6-5
 - Digital Inputs Step 6-5
 - Digital Outputs Step 6-5
 - Output Valid Step 6-5
 - Inspection Step 6-5
 - Sequence Step 6-6
 - Acquire Step 6-6
 - Tolerance Meas and PointTolerance Meas Tools 6-7
- References 6-8
 - Digital Inputs Step 6-8
 - Digital Outputs Step 6-12
 - Output Valid 6-14
 - I/O Example 6-15

CHAPTER 7 **Image Analysis Tools 7-1**

- Connectivity Analysis 7-2
 - Blob Tool 7-2
 - Blob Filter 7-14
- Circle Analysis 7-26
 - Circle Find 7-26
- Edge Analysis 7-31
 - Edge Tool 7-31
 - Fast Edge Tool 7-36
- Flaw Analysis 7-46
 - Flaw Tool 7-46
- Histogram Analysis 7-54
 - GrayHistogram Tool 7-54
- IntelliFind Tool 7-58
 - Model Edge Contours Selection 7-60
 - Controls in Model Editor UI 7-63
 - Teaching and Building a Model 7-64
 - Editing/Adding or Removing Features from a Model 7-65
 - Feature Sub-panel 7-66
 - Setting the Model Reference Frame (HotSpot) 7-67
 - Checking Model Contours 7-67
- Correlation Analysis 7-84
 - Correlation 7-84
 - Template Find 7-86
- Vector Tool 7-93
 - Setting Up the Vector Shape 7-101
 - Shape Manipulation 7-101
 - Generating a Set of Vectors 7-103
 - Results 7-107
 - Using Results 7-108

Contents

- FTP Image Logging of Pass/Fail Images 7-109
 - FTP Output Step 7-109
 - Theory of Operation 7-109
 - Potential Use Case 7-109
 - Settings 7-110
 - MS-Linkable Settings 7-111
 - Training 7-111
 - I/O Summary 7-112
 - Server Setup 7-112
 - Configuring the FTP Output Step 7-118

CHAPTER 8 **Geometric Fitting and Measurement Tools 8-1**

- Measurement Types 8-2
 - Angle 8-2
 - Distance 8-3
 - Line 8-3
 - Point 8-5
 - Area 8-5
 - Integer and Double 8-5
 - Customized Measurement 8-5
- Geometric Fitting 8-6
 - LeastSquaresCircle Fit 8-6
 - LeastSquaresLine Fit 8-9
 - RobustCircle Fit 8-13
 - RobustLine Fit 8-25
- Measurement Tools 8-29
 - Dist2Pts Meas 8-29
 - IntersectLines Meas 8-33
 - BisectLines Meas 8-35
 - PtLineNormal_Meas 8-38
 - Pt to Line Distance 8-41
- Measurements Tolerancing 8-44
 - Tolerance Meas 8-44
 - PointTolerance Meas 8-52
- Expressions 8-59
 - Expression Datum Rules 8-65

CHAPTER 9 **Automatic Identification and Symbol Quality Verification 9-1**

- Symbology Tool 9-2
- OCRTrainableFont Tool 9-13
 - OCR Font Training 9-20
 - Training Tips 9-33
 - Tips for Marking OCR Fonts 9-34

IntelliText OCR 9-35
 IntelliText Overview 9-35
 IntelliText Binarization 9-35
 IntelliText OCR Tool Inputs 9-36
 Optimization Examples 9-41
 IntelliText OCR Tool Outputs 9-48
Symbol Quality Verification Tool 9-49

CHAPTER 10

OCV Tools 10-1

OCV Inspection 10-2
 Additional Filters 10-3
 Brief Descriptions 10-4
Custom Properties — Create/Modify OCVFonts (Library) 10-6
 Custom Settings 10-6
 Main Custom Properties Dialog 10-7
 Font Manager Dialog Box 10-9
 Training Fonts 10-11
 Training the OCVFontTool 10-13
 Remove Symbol Dialog 10-16
 Automatic Font Selection and Scaling Dialog 10-16
 Automatic Font Selection — The “AutoFont” Button 10-17
 Font Scaling — The “Scale” Button 10-21
OCVFont 10-23
 Creating FontSymbols 10-23
FontSymbol 10-30
AutoFind 10-39
 Troubleshooting 10-42
OCVFontTool 10-43
OCVRuntimeTool 10-53
OCVFontless Tool 10-64
OCVSymbolStep 10-79
OCV Tips 10-87
 OCVFont 10-87
 Layout Step 10-87
 DefaultSymbol 10-87
 OCVRuntime Tool 10-88
 Converting Jobs with Embedded OCVFonts 10-88
Troubleshooting 10-90
 Training Font Based Tools – Read Match% 10-90

CHAPTER 11

Custom Tools 11-1

Custom Step 11-1
 Versions of Script Files 11-5
 Script Files Not Found or “none” Script 11-5

Contents

Custom Vision Tool	11-7
Versions of Script Files	11-10
Script Files Not Found or “none” Script	11-11
Perl Scripts Included with Visionscape	11-12
Cylinder_UnWarp	11-12
Dynamic_Binarize	11-12
FailCode	11-15
FindRotated	11-17
Color Perl Scripts	11-22
ColorTrain and ColorID	11-24
ColorSpotCheck	11-28
ColorDifference	11-31
Color Image Display Options	11-34
Color Image Display in FrontRunner	11-34
Running Color Plane Selection	11-35
Color Image Display in AppRunner	11-39
Running Color Plane Selection	11-40
White Balance	11-44
White Balance Gain Values	11-44
White Balance Implementation	11-49

CHAPTER 12 Image Transform Tools 12-1

References	12-3
Annular Unwrap Warp	12-3
Cylinder Unwrap Warp	12-9
Projection Tool	12-13
Rect Warp	12-16

CHAPTER 13 Image Processing Tools 13-1

BinaryMorph Filter	13-3
FrameAverage Filter	13-8
GainOffset Filter	13-11
GrayMorphFilter	13-13
ImageArith	13-18
Operator Set to ADD, COPY, DIFF or SUB	13-20
Operator Not Set to ADD, COPY, DIFF, or SUB	13-24
Operator Set to BINARIZE or PASSRANGE	13-25
Operator Set to MASKSHIFT	13-26
Operator Set to Any Two-Operand Operation	13-27
MeanLP Filter	13-29
Sobel Filter	13-31
Binarized Color Thresholding Filter Tool	13-37
Zooming and Binning	13-43

APPENDIX A

Error Codes A-1

Abstract Vision Step Error Codes A-1
Arith Step Error Codes A-1
Auto/Dynamic Mask Step Error Codes A-2
Binary Morphology Agent Error Codes A-2
Binary Morphology Step Error Codes A-2
Blob Agent Error Codes A-2
Blob Datum Error Codes A-3
Blob Step Error Codes A-3
Buffer Datum Error Codes A-3
Buffer Manager Error Codes A-4
C2D Error Codes (Math Library 2D Calibration) A-4
Correlation Search Agent Error Codes A-6
Custom Step Error Codes A-6
DataMatrix/BarCode Step Error Codes A-7
Edge Step Error Codes A-9
File I/O Error Codes A-9
GainOffset Step Error Codes A-11
General Error Codes A-12
Gradient Scan Agent Error Codes A-12
Gray Morphology Agent Error Codes A-13
Hough Agent Error Codes A-13
Hough Agent Error Codes (Data Table) A-14
Hough Agent Error Codes (Line Fit) A-15
Hough Agent Error Codes (Matrix) A-15
Hough Agent Error Codes (Scan) A-16
Line Functions Error Codes A-16
Mask Agent Error Codes A-17
Mask Datum Error Codes A-17
Measurement Step Error Codes A-17
Mean Filter Step Error Code A-18
Npin Find Step Error Codes A-18
Point-Point Distance Step Error Codes A-18
Project Step Error Codes A-19
Shape Datum Error Codes A-19
Snapshot Error Codes A-20
Sobel Step Error Codes A-21
Vector Error Code A-21
FTP Output Step Error Codes A-21

APPENDIX B

Sample Calibration Targets B-1

Welcome

Purpose of This Manual

The manual helps you to program and customize your applications. It provides instructions to develop vision programs, called Jobs, using the FrontRunner application running on the host PC. It also provides in-depth information on the various tools used to create an inspection Job.

Manual Conventions

The following typographical conventions are used throughout this manual.

- Items emphasizing important information are **bolded**.
- Menu selections, menu items and entries in screen images are indicated as: Run (triggered), Modify..., etc.

CHAPTER 1

Advanced Visionscape Concepts

This chapter explains advanced Visionscape concepts.

Introduction

The Visionscape software allows one or more inspections to execute concurrently on a GigE Vision System or smart camera. An inspection can capture and process one or more images from one or more GigE cameras connected to the Visionscape system.

Pass/Fail results are produced at the end of the inspection and can be accessed in the ResultUploadDone event in VB for further analysis and reports. Pass/Fail results can also be signalled directly through Digital I/O or Virtual I/O.

Acquisition Setup

Single Camera

The Snapshot/Acquire pair of steps handles image acquisition.

The Acquire Step allocates a buffer, which is memory where the acquired image is stored, and starts image acquisition. Once the acquisition is finished, image processing tools execute on that image. At the end of the inspection, the buffer containing the image is released, allowing it to be reused during the next cycle.

Triggered Acquisition

If the Acquire Step is triggered, the Acquire and Snapshot Steps are executed concurrently, allowing the acquisition of an image to occur while the processing of the previous image is underway. This uses an additional image buffer per Snapshot to store the image while Snapshot processes the previous one. This is called pipeline execution.

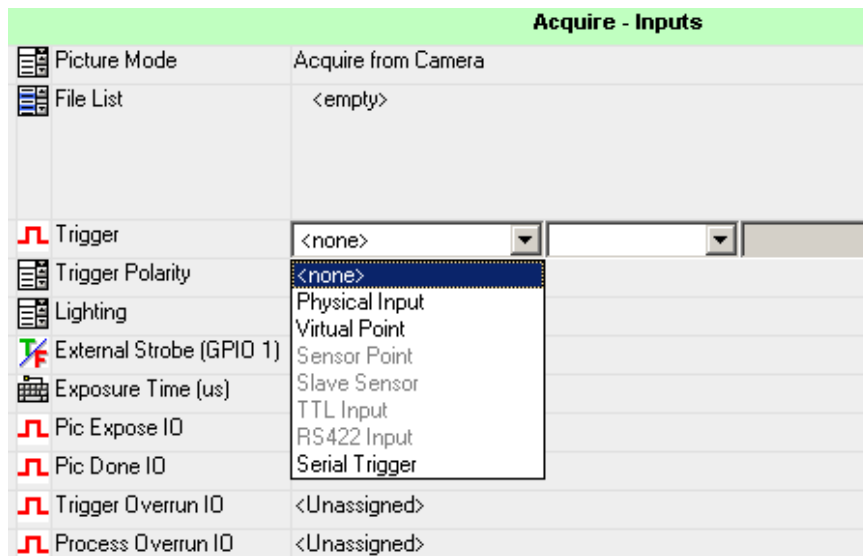
Note: If the triggering occurs in bursts and the image processing takes significant time, the pipeline can get several images deep while the image processing works on the current image.

In triggered mode, the inspection busy signal and data valid signals are available to implement handshaking with the control system providing the triggers.

Single Camera Pipeline Processing

Pipeline execution occurs when processing of the previous image is overlapped with acquisition of the current image. The inspection must be triggered in order to enable single-camera pipeline. From FrontRunner, select the Acquire Step in the Job Tree and enable the trigger by selecting the appropriate trigger source from Trigger, as shown in Figure 1–1.

FIGURE 1–1. Selecting the Appropriate Trigger Source



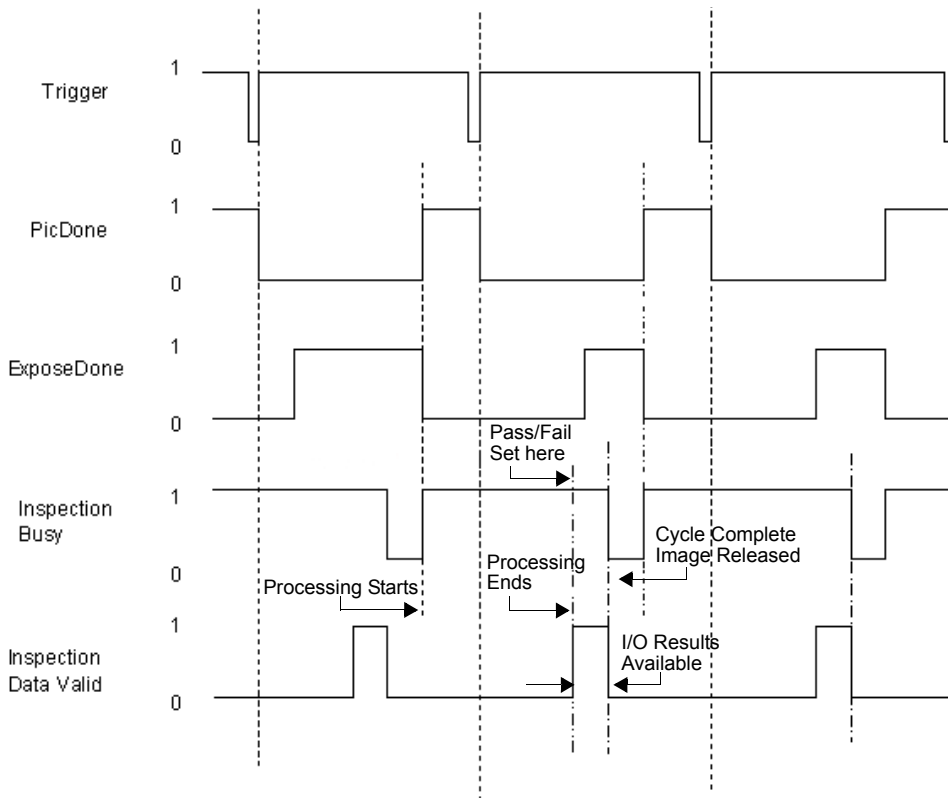
In addition, set **Trigger Polarity** (High->Low or Low->High) based on the type of trigger signal used.

The types of triggers are:

- **Physical Input** — There are General Purpose I/O points available on GigE cameras. These can be configured as inputs, in which case they are valid triggers, or outputs, in which case they cannot be used as triggers. The definition of the points as inputs or outputs is done in the VisionSystemStep.
- **Virtual Point** — There are 2048 virtual I/O triggers. They are controlled through the AvpIOClient object and can be used in the same manner as physical inputs and output. However, Virtual I/O points are both inputs and outputs simultaneously. They can trigger internally the system from another Inspection. For more information, see “Triggering Acquisitions Internally” on page 1-17.
- **Sensor Point** — There are 4 built-in, dedicated sensor lines available. They are input only points.
- **Slave Sensor** — This is a special point used for master slave configurations.
- **TTL Input** — Unavailable on GigE systems.

- RS422 Input — Unavailable on GigE systems.
- Serial Trigger — This option triggers the acquisition when a user-specified string is received on either a TCP or RS-232 serial port. Select which port to listen on with the second drop down box and enter the match string in the edit box. The acquisition is triggered when the last character of the match string is received.
- IO Board DIO01 — Selects one of the general purpose inputs from a PCIe IO board. There are 16 input points and 16 output points. These points are also available as Virtual Points and will also appear in that list starting point number 160. There is no difference in operation whether the point is selected from the Virtual Point list or the IO Board list. NOTE: When selecting an IO Board input point for triggering, only the first 8 points can be used for low-to-high trigger events and only the second 8 input points can be used for high-to-low trigger events. If trigger events from both edges are required, the signal should be connected to two input pins, see the GigE Camera Guide for more information.

The timing of the trigger, the busy signal, and other related signals are shown in Figure 1–2, which shows a single-camera inspection running in pipeline mode, with overlapped processing and acquisition.

FIGURE 1-2. Timing Diagram for Triggered Single Camera Inspection

The PicDone signal is asserted each time the image becomes available from the Digitizer. The image processing signal, Inspection Busy, occurs as soon as the image is digitized and is reset at the end of the cycle. At the end of the inspection, the Pass/Fail status is asserted and can be read while the data valid signal is active. The cycle starts again as soon as the data valid signal is turned Off, at which point the inspection releases the Snapshot's image. The next image acquisition occurs while the inspection is running on the previous one.

All signals can be programmed and attached to physical or virtual I/O outputs and are located as follows:

- **Trigger, PicDone, ExposeDone: Acquire Step** — The Expose signal is asserted whenever the camera sensor is ready to acquire a new image. This occurs before image readout from the camera is complete. The camera or part can be moved as soon as the Expose Signal has been asserted, allowing an application to overlap the changing of the scene with capture and processing of the previous scene.

- **Busy, Data Valid: Inspection Step** — The programmable data valid duration should be set according to the external system response time. When the Target Vision Accelerator is connected to a PLC, for example, the Data Valid duration should be programmed as the PLC scan rate in milliseconds. Both the high and low times of this signal are guaranteed to be at least the programmed pulse width.

Trigger Filter, De-bounce, Delay

The purpose of the **filter** is not to trigger or end on random noise. Filter removes leading edge noise from state change. The purpose of **de-bounce** is to not retrigger or End a read cycle from switch noise or contact ringing. De-bounce removes trailing edge noise that could pull one out of a state such as a read cycle and even result in the start a second read. The purpose of **delay** is to postpone a capture after a valid trigger has been received to counterbalance the time between a physical trigger and the arrival of the target object in the system's field of view.

Filter allows you to prevent a false start and or a false end. But if you only have noise during the change of state, you don't need a filter – you only need de-bounce. However, filter creates a variable picture snap timing based on variable noise durations.

De-bounce always starts on the first transition. The system will ignore any other trigger activity until the de-bounce timer expires. However, de-bounce may start a read on various kinds of spurious noise, because the system will not see any triggering activity again until a valid trigger is received and de-bounce time expires. The same applies to end-of-read.

Delay will read after a valid trigger has been received. You will not have to move the trigger sensor around to meet read trigger requirements instead of changing the trigger delay to meet those requirements. However, while using delay, you will not be able to read while the system is in a trigger delay condition until the delay time expires.

Filter (Camdef Default 100 μ s)

Leading Edge does not signify high or low input values. It leads into ACTIVE READ, or it leads into END OF READ. Additional edges restart the filter timer. Filter is the time it takes noise to stabilize. For example, a filter time of 2 ms is used if noise bursts are not spaced closer than 2 ms. The noise could be 5 ms long, but it will not satisfy the requirement for 2 ms of stability. This is why the filter timer must restart on any activity.

De-bounce (Camdef Default 1 ms)

Trailing Edge is used after ACTIVE READ starts or after END OF READ, after the change of state has already been published to the system, at which point a new change of state can be published again.

Strobing

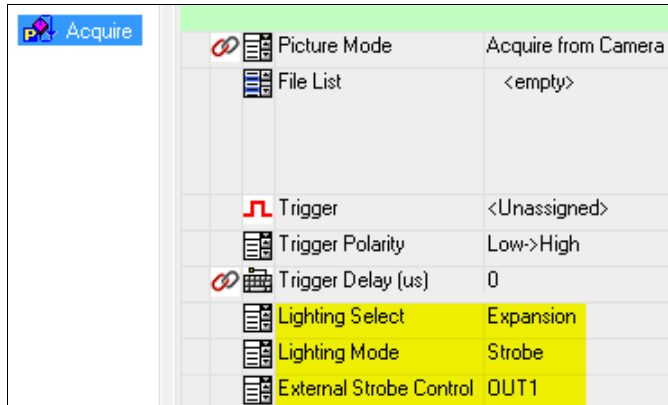
When a part is moving, it is necessary to program a strobe into the Acquire Step, to freeze motion at the time of the trigger and acquire a coherent image for processing.

The strobe signal can be selected as one or more of the four strobe outputs in the AcquireStep.

Power Strobe and External Strobe

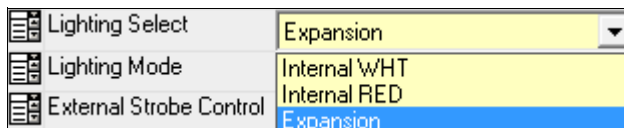
Power Strobe and External Strobe allow you to select different lighting configurations as part of the Acquire Step.

Note: Power Strobe does not function with rolling shutter.



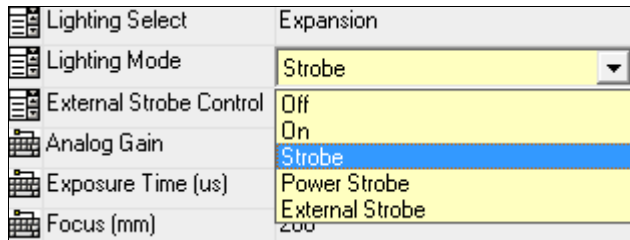
Lighting Select

Allows you to select between Internal WHT, Internal RED, or Expansion lighting. This setting is used in conjunction with the Lighting Mode selection.



Lighting Mode

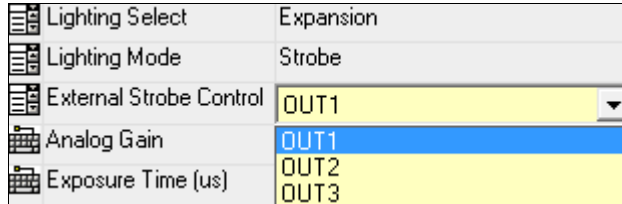
Allows you to select between Off, On, Strobe, Power Strobe, or External Strobe. Power Strobe mode is limited to a maximum 1 ms pulse width (as with Omron Microscan's NERLITE Smart Series Illuminators).



External Strobe Control

Allows you to select which output is used to control the External Strobe. The selected output is no longer available for use elsewhere in the job.

Note: There are no outputs available for MV-20.



Lighting Mode Table

Light Mode Table	Inner LEDs	Outer LEDs	Off	On	Strobe	Power Strobe Inner LEDs	Power Strobe Outer LEDs	External Strobe
Engine USB Power	Yes	No	Yes	Yes	Yes	No	N/A	Yes
Engine Ext Power	Yes	No	Yes	Yes	Yes	No	N/A	Yes*
MV-20	Yes	No	Yes	Yes	Yes	No	N/A	No
MV-20 Color	Yes	No	Yes	Yes	Yes	No	N/A	No
MV-30 USB Power	Yes	Yes	Yes	Yes	Yes	No	No	Yes**
MV-30 External Power	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
MV-30 Color	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
MV-40	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
MV-40 Color	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

* If using breakout cable for engine

** If using correct accessory breakout cable

Multiple Cameras

This section applies to the Visionscape GigE systems only, which support more than one camera. When using multiple cameras, you have a few things to consider when programming your Job. Primarily, you need to consider whether you want your image acquisition to be synchronous or asynchronous.

- **Synchronous Acquisition** — This means that the image acquisition from each camera will happen in a synchronous fashion. A Job is programmed for synchronous acquisition by inserting multiple Snapshot Steps into a single Inspection Step. In this model, all of the Snapshots must acquire their images before an Inspection cycle can complete. Typically, it means that you will be triggering all cameras at the same time, and that you wish to receive one set of inspection results from all cameras. Typically, multiple cameras that look at the same part are programmed to be synchronous.
- **Asynchronous Acquisition** — This means that the image acquisition from each camera will happen in an asynchronous fashion. A Job is programmed for asynchronous acquisition by inserting each Snapshot Step into a separate Inspection Step. In this model, the cameras act independently of each other, and you will receive a separate set of results from the inspection of each image. Typically, this means that you will be using different triggers for each camera, and that you do not wish for the results of each camera to be tied to the results of the other cameras. In other words, choose this option when you want to create an entirely independent inspection for each camera. Typically, multiple cameras that look at different parts, or at different inspection stations on a machine, are programmed to be asynchronous.

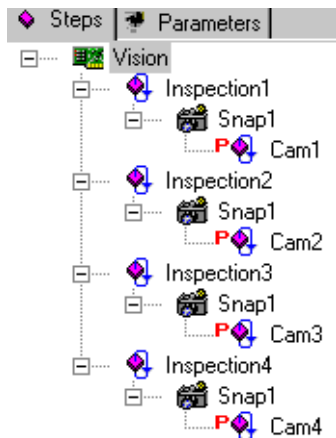
Detailed descriptions of these two acquisition models follow.

Multiple Camera Asynchronous Operation

You create an asynchronous image acquisition Job by inserting each of your Snapshot Steps into a separate Inspection Step. For example:

- Four Inspections, Four Snapshot/Acquire Pairs — Each Acquire Step's "Camera Number" datum is set to use a different camera channel, and each "Trigger" datum is set to use a different IO point. Figure 1–3 shows the Job for this model. Each inspection will compute its own independent set of results.

FIGURE 1–3. Four Asynchronous Inspections



Note: Results cannot be shared between inspections unless they are synchronized using I/O Input steps. Results for each inspection are delivered separately to the VB application in multiple ResultUploadDone events, one for each Inspection.

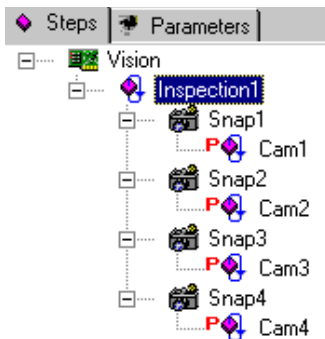
If you were to assign the same trigger to two or more of the acquire steps, you would cause those cameras to acquire simultaneously (synchronously), but their inspections will still run independently. In general, when multiple cameras will use the same trigger, you should insert your snapshot steps into the same inspection, since this will improve performance.

Multiple Camera Synchronous Operation

You create a synchronous image acquisition Job by inserting all of your Snapshot Steps into the same Inspection Step. For example:

- One Inspection, Four Snapshot/Acquire Pairs — Each Acquire Step's "Camera Number" datum is set to use a different camera channel, and each "Trigger" datum is set to use the SAME IO point. You'll use this model when processing from all cameras must be combined into a single set of results by the inspection. Figure 1–4 shows the Job for this model.

FIGURE 1–4. One Inspection, Four Snapshot/Acquire Pairs



The images are processed in sequential order, according to the way in which the Snapshots are ordered in the Job. In this mode, Snap1 is processed first followed by Snap2, Snap3, and Snap4. As soon as Snap 1 is acquired, Visionscape will begin processing it, even if acquisition is not complete on the other snapshots. All strobes are fixed and all cameras are inhibited at the same time. Therefore, the images are acquired at a single moment, which is trigger time.

Note: It is possible to construct a Job like the one shown in Figure 1–4 that uses different triggers for each of the acquire steps. There are some application scenarios where this type of configuration will be required, but in general, we recommend against this, as this can lead to the confusing situation where your inspection images appear to be out of sequence.

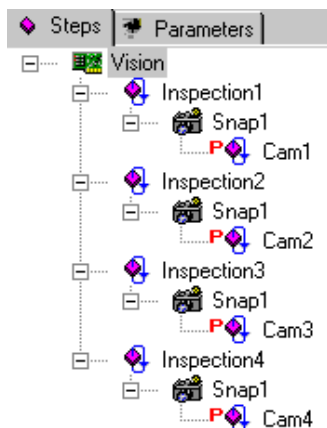
Consider the following scenario:

You have two snapshots inserted inside of a single Inspection step, and their triggers are assigned to Sensor input 1, and Sensor input 2, respectively.

Remember that an inspection cycle can not complete until ALL snapshots have been triggered. Now, imagine that noise on your trigger line should cause Sensor 1 to accidentally fire at some random time between parts. An image will be acquired immediately by snapshot 1 and it will be inspected, but the inspection will then move to Snapshot 2 and block, waiting for it to be triggered. An image will NOT be displayed in FrontRunner at this point, because images are not uploaded until the inspection cycle completes. When the next part comes in front of the cameras, and Sensor 1 triggers again, the image will be acquired and buffered, but not inspected, as the inspection is currently blocked waiting for Snapshot 2 to finish. When Sensor 2 triggers, Snapshot 2 will acquire an image and inspect it, and the inspection cycle will complete. The Inspection step will then loop around and start the next cycle, but instead of blocking and waiting for the next trigger on Snapshot 1, it will see that there is already a buffered image, and inspect it immediately and then move to Snapshot 2 again, waiting for it to be triggered. As you can see, your inspection is now out of sequence, and will remain that way until you stop and restart your inspections. So avoid using multiple triggers whenever you have multiple snapshots in one inspection.

- Four Inspections, Four Snapshot/Acquire Pairs, same trigger used for all snapshots — Because the snapshots all use the same trigger in this scenario, the image acquisition will be synchronous in the job shown in Figure 1–5. A single trigger starts processing of every inspection. Because each Inspection step runs in its own thread, however, they will all run concurrently and, therefore, the order in which each will finish is somewhat random. In other words, you should not assume in this scenario that Inspection 1 will finish first, followed by Inspection 2, then 3 and 4. The image acquisition is synchronized in this scenario, but the Inspections are not.

FIGURE 1-5. Four Inspections, Four Snapshot/Acquire Pairs



In this model, each inspection computes independently its own set of results. The strobes are fired and the cameras are inhibited individually. This introduces a slight time difference from the firing and inhibiting of the first camera to the last. Typically, you would only program your Job this way if you needed a separate set of counts and results for each camera.

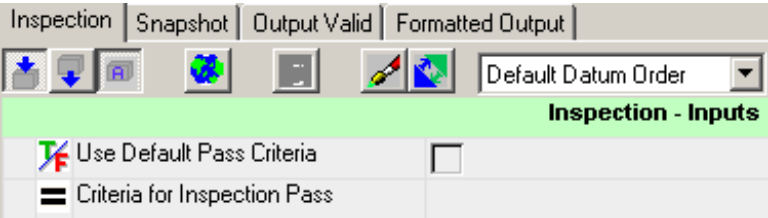
Note: Results cannot be shared between inspections unless they are synchronized using I/O Input steps. Results for each inspection are sent as separate events to the VB application, and the order of receipt of these events will be random.

Producing Reports

Fail/Pass Results Setup

By default, the Inspection Step is programmed to set the Pass/Fail status for the current cycle by ANDing all tool statuses together. In the case where a different criteria is necessary, a free-form expression can be substituted as the Inspection Step Pass/Fail cycle condition. Disable (uncheck) **Use Default Pass Criteria**, as shown in Figure 1–6.

FIGURE 1–6. Use Default Pass Criteria Disabled



Results are calculated during the inspection and sent to VB at the end of a cycle, and then displayed in a spreadsheet view in FrontRunner. Results are available for every active inspection.

The following results are reported:

- Execution report including Cycle and Processing time statistics.
- Individual tools results such as points, lines, and Pass/Fail status.
- Statistic results calculated over many cycles by the Tolerance Meas and PointTolerance Meas steps.

An example results report is shown in Figure 1–7.

FIGURE 1–7. Statistics and Report Vision:Inspection Window

Hawkeye1000E							
Insp1	Inspect: 595		Pass: 297	Fail: 298			
Cycle	68	Cyc Worst	72	Process	47	Draw	0
PPM	881	PPM Worst	1200	Idle	15	DMA	0
Buffers	9 of 20 used (45%)			Overruns	None		

Select the results to be uploaded in the Inspection properties page, using **Select Results to Upload**.

Inspection Execution Timing

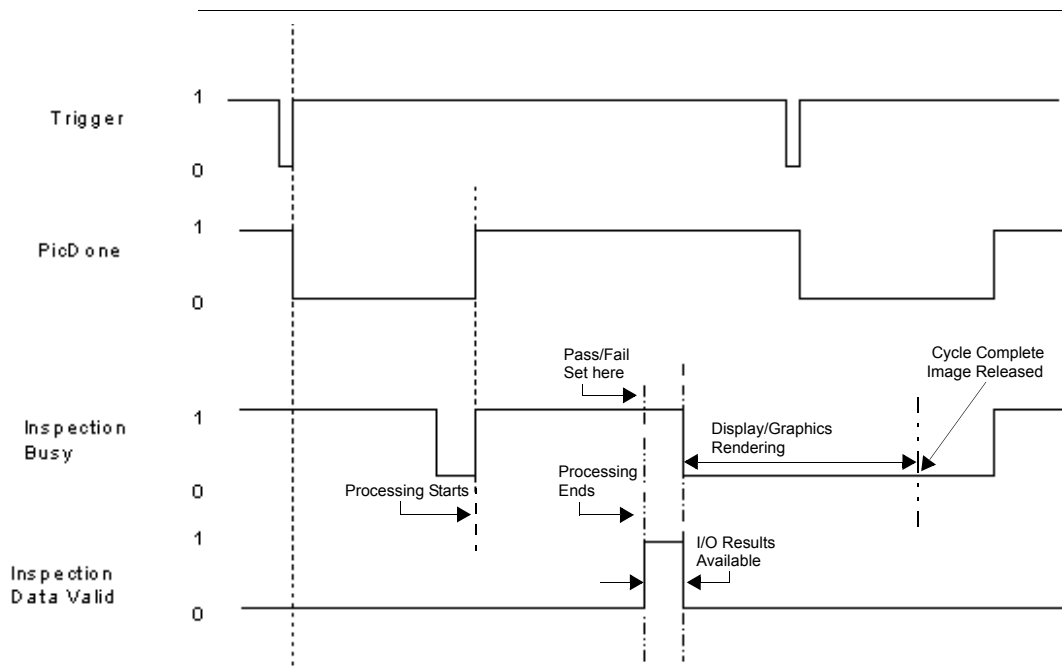
Runtime execution of an inspection includes the time required to display a particular image with graphics on the host PC monitor.

Results upload and image display decrease the throughput of an inspection, as they must be done as part of the inspection cycle. For a typical mix of results, for example, for a set of 15 or so results, expect an additional 3 to 5 milliseconds on the inspection overall cycle time. This can be measured by running the inspection without any results selected for upload and noting the cycle time. Then, stop the inspection, select results for upload and run the inspection. Once the inspection is running, turn on the Report by clicking **Reports** from FrontRunner and note the cycle time. The difference is the time spent collecting and sending results at the end of the cycle.

Images and graphics sent to VB are part of the execution of each cycle for as long as there is a display connected to one of the Inspection Snapshots.

An inspection cycle with image display and graphics enabled is shown in the timing diagram in Figure 1–8.

FIGURE 1–8. Inspection Timing Diagram with Display and Graphics



Non-Triggered Image Acquisition

By default, FrontRunner creates a non-triggered single camera inspection. For each GigE Camera or smart camera, this is well suited for bench application testing on saved images. In this mode, an inspection runs continuously at the fastest rate possible with the limit being two thirds (2/3) of the frame rate of an interlaced camera.

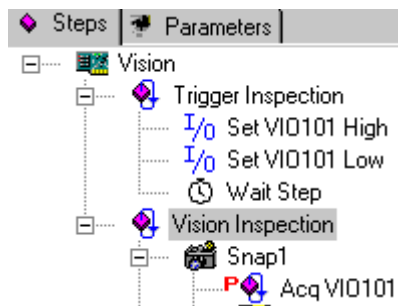
Note: An equivalent triggered inspection maintains a faster rate because image acquisition and image processing are pipelined when camera images, not stored images, are used.

When running from stored images, the inspection cycles continuously all the images that have been programmed into the Acquire Step.

Triggering Acquisitions Internally

To test the maximum possible rate without an external trigger, a trigger inspection can be created. Its job is to generate triggers at regular intervals to another inspection. The Job achieves this by selecting a virtual I/O signal to be generated from one Inspection and then using the same signal as the trigger in another Inspection, as shown in Figure 1–9.

FIGURE 1–9. Trigger Inspection for Pipeline Continuous Mode



Two Digital I/O Steps create a High->Low transition on the Virtual I/O Point 101. The Acquire step in Vision Inspection is programmed to accept Virtual I/O Point 101 for the trigger with High->Low polarity. The Wait Step wait time is selected such that the inspection can run at maximum speed without overrunning.

Note: This can also be achieved from the host with VB code attached to a Timer object. Then, the VB code can assert a Virtual I/O Point at periodic intervals, causing the inspection running on the GigE Camera or smart camera to receive a trigger.

Advanced Triggering Techniques

Both synchronous and asynchronous runtime modes can be mixed freely when the Inspection Step contains Snapshot/Acquire Step pairs using the same triggers, different triggers, or no triggers at all. A non-triggered Acquire executes serially with the other tools in the Job. The inspection is suspended while the camera I/O card captures the image.

Mixing Acquisitions Within a Single Inspection

A typical synchronous mode is to construct an Inspection that contains two Snapshots connected to different cameras.

This scenario does not run as optimally as the following scenario. In the second scenario, each of the two inspections contains its own Snapshot Step. The first inspection is triggered and outputs a Picture Done signal, which subsequently triggers the second inspection's Snapshot Step. In essence, the first inspection is still in process as the second inspection begins. This is done by programming the trigger of the "Acq PicDone1" tool to be the Acq PicDone signal of the "Acq VIO101". Practically, program the PicDone output of the first Acquire Tool to some virtual I/O point and use it as the trigger in the second Acquire Tool.

FIGURE 1-10. Example — Synchronizing Two Inspections

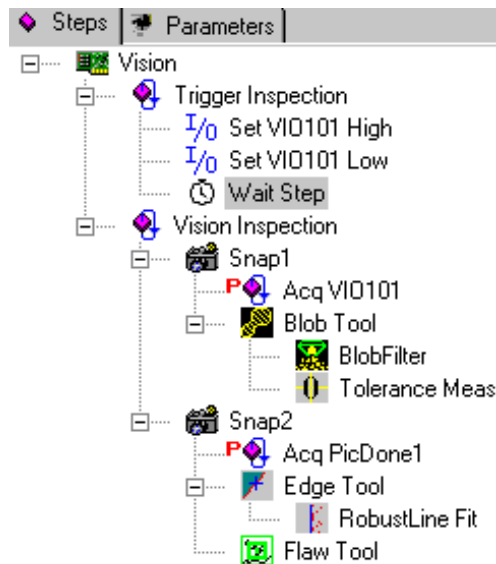
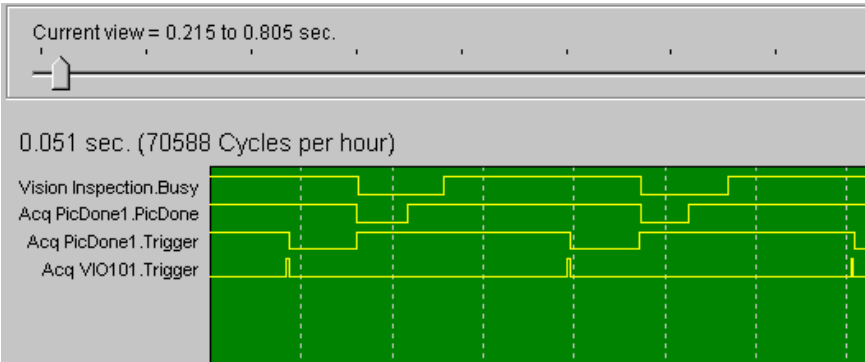


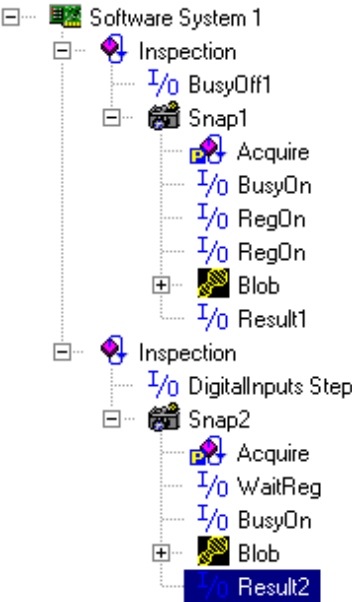
FIGURE 1–11. Triggered Inspection w/Self-Triggered 2nd Snapshot Timing



Synchronizing Two Inspections

Another useful technique is to synchronize the execution of one inspection with another. This is necessary whenever results from one inspection are used by a second inspection. It is typical in this situation to trigger the first inspection externally and trigger the second inspection internally from the first one at the appropriate time, as shown in Figure 1–12.

FIGURE 1–12. Across Inspections Synchronization with an I/O Input Step



Assume that a part is indexed into position, that two pictures (Snap1 and Snap2) from the same camera are taken, and that the lighting must be different between these two acquisitions.

Locating the part can only be performed in Snap1 but is also used by a Dynamic Locator Tool in Snap2. It is necessary to identify the part as early as possible so that the Expose signal must be used. You must switch the lights whenever the Expose Signal of Snap1 is asserted and move the part to the next index position whenever the Expose Signal from Snap2 is asserted. In this example, the control system, e.g., VB or a PLC, must provide the following signals:

1. Trigger signal to Snap1 when results have been processed for Inspection1 and Inspection2 for the previous cycle.
2. Switch light in response to Expose Signal 1, wait for PicDone1, and then trigger Snap2.
3. Identify part and switch light.
4. Combine final result whenever Inspection1 and Inspection2 are asserted. They must both be Off for results to be available for the previous cycle.

Two features are demonstrated in the previous example:

- The use of the Expose1 to switch lights at the earliest possible time.
- The use of synchronization between Inspection1 and Inspection2 using an I/O Input Step in Inspection2 to ensure that the location of the device has been completed in Snap1 before tools in Snap2 execute.

An I/O Output Step RegOn and RegOff brackets the location tool in Snap1. The I/O written by RegOn/RegOff is input to the I/O Input Step WaitReg in the Snapshot of Inspection2. The I/O Input Step waits for the transition of the Reg Signal from High to Low, which indicates that the TwoPt Locator in Inspection1 has completed execution. When location completes by the time WaitReg is run, no wait occurs and the Inspection2 can continue execution. In that mode, the I/O Input Step acts as synchronization Step.

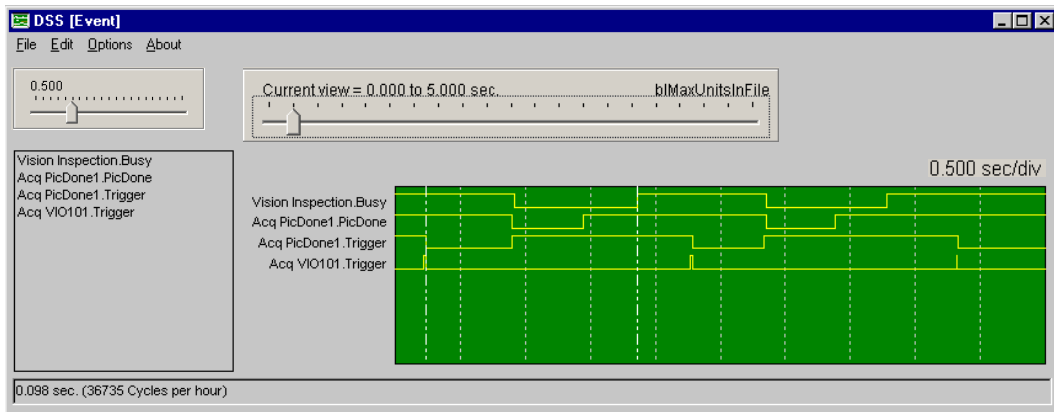
Note: Set the transition mode to “Wait for State” as opposed to “Wait for Transition” to use the I/O Input Step in this mode.

Viewing Timing Signals from Running Set of Inspections

In FrontRunner, you can display I/O signals graphically. Any I/O signal that has been programmed in the Job can be added to this display.

1. From FrontRunner, select **View > Digital Soft Scope**.
2. Set a buffer size to receive the I/O events. Select **Options > Time Base > 0.50 sec/div** (500 milliseconds, which is the default). You can now start the Job for that designated time.
3. Stop the Job.
4. Open the View menu from FrontRunner.
5. Select **Digital Soft Scope>View**. The Digital Soft Scope tool is displayed, as shown in Figure 1–13. The resulting signal trace can be viewed graphically as a timing diagram, providing a useful tool for analyzing I/O timing and interaction.

FIGURE 1–13. Digital Scope Traces for Self-Triggered Inspection



System Resource Considerations for Pipeline Operation

Image Buffer Allocation

The GigE Camera or smart cameras support a number of hardware devices that provide for pipeline execution. For example, it is possible to acquire an image from the camera and simultaneously process a previously acquired image, as well as show an older image, such as a failed image, on the PC display.

In order to implement full pipeline, memory resources must be allocated properly to keep track of these images.

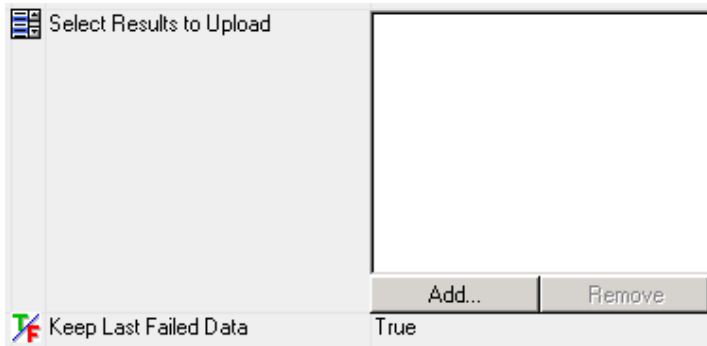
Setting Number of Image Buffers for Pipeline Operation

Number of Image Buffers represents the number of buffers available for image capture. This parameter is set in each Vision System Step properties page.

Visionscape has a built-in pipeline architecture where acquisition is overlapped with processing whenever possible. In addition, images are processed as soon as they become available, and are freed back to the pipeline at the beginning of the next cycle. You need the following minimum number of buffers:

- Two buffers for each triggered Snapshot step or one buffer for each non-triggered Snapshot step in all Inspections.
- When **Keep Last Failed Data** is enabled, as shown in Figure 1–14, you need one additional buffer for each Snapshot step in all Inspections.

Note: When **Keep Last Failed Data** is disabled, failed images are not available for display because they are not recorded when they occur. However, “Fail Next Image” continues to function, since the image that fails in the future will be sent immediately for display and still not recorded.

FIGURE 1-14. Activating Last Failed Image Recording

In some applications, bursting must be allowed when triggers come in groups. This requires more buffers, and is dependent on the inspection time. However, if the inspection time consistently exceeds the trigger-to-trigger time, an overrun condition eventually occurs, since buffers are not released as fast as the images are being acquired.

Note: These formulas maintain maximum execution speed and display images on the PC monitor. Fewer image buffers are required if the cycle is slower. Buffers are released by the Inspection Step at the beginning of each cycle.




Image memory usage can be tuned by programming the **Number of Image Buffers** in the Vision System Step properties page (see “Vision System Step” on page 1-86). This allows it to match the number of Snapshots and graphics parameter set up for that inspection.

Overrun Conditions

Two types of overrun conditions may occur during normal operation of the system. These conditions can be programmed to generate I/O transition such that VB or an external PLC can initiate the appropriate processing. Usually, these conditions are very serious, as the vision system is no longer inspecting parts and every part will fail the inspection.

The Acquire step can be programmed to assert an I/O point when either condition is true. The signal remains high until the overrun condition disappears, e.g., until an image buffer becomes available for the acquisition to succeed and/or until the trigger-to-trigger time is such that the camera can be exposed again.

FIGURE 1–15. Processing and Trigger Overrun I/O Settings

 Pic Expose IO	<Unassigned>
 Pic Done IO	<Unassigned>
 Trigger Overrun IO	<Unassigned>
 Process Overrun IO	<Unassigned>

- **Triggering Overrun** — A condition that occurs when the system is being triggered faster than the frame rate for the selected camera. Refer to one of the Visionscape GigE Camera Guide.

Trigger overruns do not relate directly to uninspected parts. Because trigger overruns may occur much faster than camera frame rate, they are compressed to a single error frame per normal camera period.

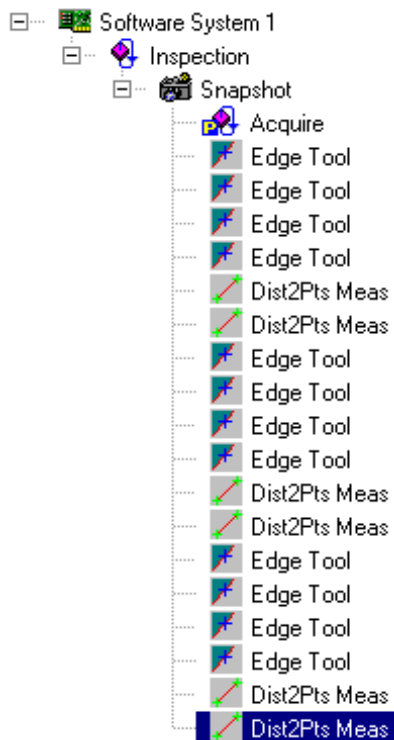
- **Processing Overrun** — A condition that occurs when the processing time takes consistently longer than the trigger-to-trigger time. Over a number of cycles, the number of buffers allocated to image capture events will be exhausted as the pipeline fills with images to be processed. There is a small buffer (500 on Visionscape GigE Cameras) to record process overruns after which process overruns and parts not inspected counters will no longer match. Based on the speed of the line this is usually anywhere from a few seconds to about 1 mn of continuous overruns.

Note: Display options affect how fast an inspection can run. The fastest possible cycle is achieved when graphics and image refresh are turned Off.

Introduction to Trajectory Inspections

There is a class of applications that requires the same set of inspection steps to be carried out at different locations in the image. For example, take a tray of the same type of electronic component whose width and height need to be measured. Normally, you would put four Edge Tools on each part to find the outside edges of the devices, and then two Point-to-Point Distance tools to get the part width and height. If you had 10 devices in the field of view (FOV), then 60 tools would have to be created and placed. Figure 1–16 shows a set of Inspection Steps that are repeated three times (four Edge Tools and two Dist2Pts Meas for each step).

FIGURE 1–16. Inspection at Different Location Without Trajectory



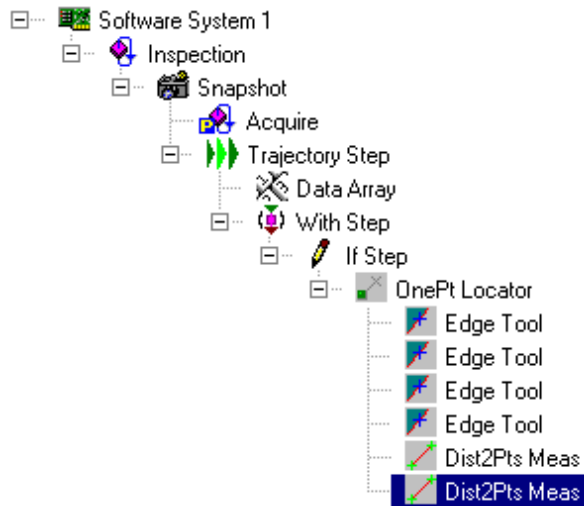
With the Trajectory Step, all you have to do is define the initial six tools, and then input a set of locations where you want to run that set of tools. The Trajectory Step runs the tools at the first location, gathers and stores the results, and then moves the same tools to the next location, runs them there, gathers and stores those results, and so on.

Trajectory Jobs end up being much smaller in total number of tools, and much easier to train and to lay out the regions of interest (ROI).

The Trajectory Step is a single tool, but is used in combination with four other helper steps that allow it to perform this functionality.

- The **main step** is the Trajectory Step. It is based on the Loop Step (for more information, see “Loop Step” on page 1-66). The Loop Step is the step responsible for actually executing the set of child steps over and over for the number of times specified.
- The **first helper step** is the Data Array Step (based on the “FlexArray Step” on page 1-42). The Data Array Step does two main things:
 - It holds the set of runtime parameters including the locations for the tools to be run, run enable flags and graphics enable flags.
 - It also holds sets of results. It accumulates one set of results from each location where the tools are run.
- The **second helper step** is the With Step (see the “With Step” on page 1-92). This is the step that determines which location the steps are being run at. It is primarily responsible for making sure the results, which can be in calibrated space, are corrected for the location that they run at.
- The **third helper step** is the If Step (see the “If Step” on page 1-54). This step simply looks at the set of enable flags and allows or disallows the actual vision steps to be run for that iteration through the loop.
- The **fourth helper step** is the OnePt Locator. It is fed the points from the Data Array Step for each location, and then it is responsible for moving the ROI of the vision steps.

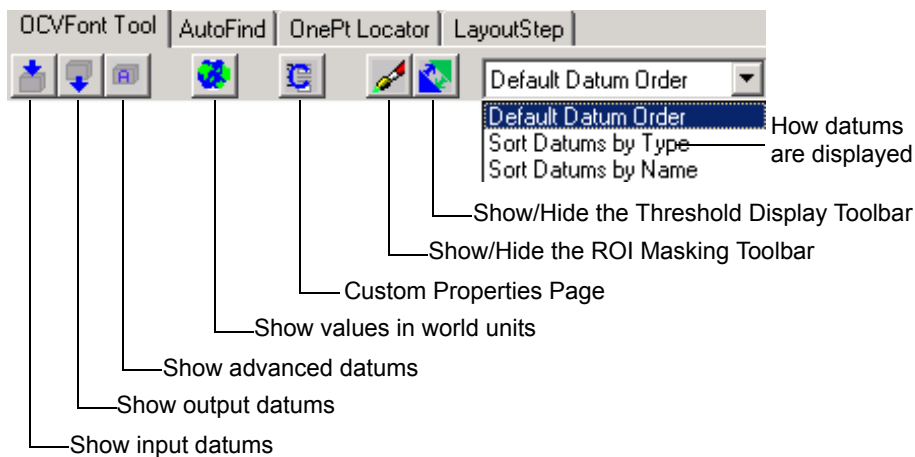
The example described above is shown in Figure 1–17.

FIGURE 1–17. Inspection at Different Location With Trajectory

Buttons on the Properties Page

Figure 1–18 shows the buttons that are displayed on a properties page.










FIGURE 1–18. Buttons on the Properties Page



Resource Datums

The icon designates what type of editor will edit that datum. Table 1–1 shows the icons for Resource datums.


TABLE 1–1. Icons for Resource Datums

Icon	What It Means
	A Command Button is used as the editor for this datum.
	The Camera Definition editor is displayed for this datum.
	The Expression Editor is displayed for this datum.
	The IO Selection editor is used for this datum.
	Use the keyboard to enter data for this datum via a text box control.
	Use a combo box to select one option for this datum from a list of options.
	Use a multi-selection combo box to select one or more options for this datum from a list of options.
	This datum has either a TRUE or FALSE setting.
	Displayed for any type that is not recognized by the DatumGrid control.

Input Datums

Input datums get their values from another datum from another step. They will always be indicated with the following arrow icon, which is displayed in the editor cell of the grid. Then, the Type column of the grid will display the icon that indicates the data type of the datum. See the Output datum section below for a list of all datum type icons. Table 1–2 shows the icon for input datums.











TABLE 1–2. Icon for Input Datum

Icon	What It Means
	This icon indicates an Input Datum. This datum “references” the value from another datum. Whenever you see this icon, you will use the Select Reference editor to select another step in your Job that will supply the datum value for this datum.

Output Datums

For output datums, the icon specifies the data type of that datum. Table 1–3 shows the icons for output datums.

TABLE 1–3. Icons for Output Datums

Icon	What It Means
	This is an Angle Datum.
	This is an Area Datum.
	This is a Distance Datum.
	This is a Double Datum.
	This is an Integer Datum.
	This is a Line Datum.
	This is a Point Datum.
	This is a Point List Datum.
	This is a Status Datum.
	This is a String Datum.

References

The remainder of this chapter describes the following steps:

- “Acquire Step” on page 1-32
- “FlexArray Step” on page 1-42
- “Formatted Output Step” on page 1-44
- “If Step” on page 1-54
- “Inspection Step” on page 1-55
- “Job Step” on page 1-65
- “Loop Step” on page 1-66
- “Sequence Step” on page 1-68
- “Snapshot Step” on page 1-70
- “Trajectory Step” on page 1-77
- “Trajectory Grid Setup Step” on page 1-80
- “VarAssign Step” on page 1-83
- “Vision System Step” on page 1-86
- “Wait Step” on page 1-91
- “With Step” on page 1-92

Acquire Step

This step works directly with the hardware to produce images for Snapshot. Acquire monitors frames received from the hardware according to its acquisition definition, then pipelines these frames to Snapshot through the frame datum list. Acquire may also be configured to retrieve an image directly from disk.

Other Steps Used

Snapshot — Acquire is inserted automatically as a pre-processing step of Snapshot. There is always a one-to-one relationship between Acquire and Snapshot. Refer to “Snapshot Step” on page 1-70 for more information.

Theory of Operation

The Acquire step works with the hardware of the parent VisionSystemStep to produce image frames for its related SnapshotStep. Depending on the configuration, Acquire receives a frame from the hardware, which is either an image acquired from the camera or a stored disk image (TIF or BMP). The frame contains the image, time stamp information, and the state of physical I/O at the time of acquisition. When the Acquire step receives the frame, it creates a frame datum, and places it into a frame datum list. This connects the Acquire step to the Snapshot step. Once the frame is available in the list, the Snapshot step can set the frame as its output buffer datum for vision steps to process.

Frames for acquisition are always pulled from the buffer pool that is configured by the VisionSystemStep. The buffer pool is a pre-allocated pool of buffers used for image acquisition. A frame is pulled from this pool, passed to the Acquire, then to the Snapshot and its vision tools. If used for display, the frame is then passed to the objects that display the image. The frame may also be stored by the Inspection if it is a “failed” image. Once all objects have released the frame, it is returned to the buffer pool for subsequent use. If the acquisition cannot get a frame from the buffer pool, a “process overrun” occurs.

When programming your buffer pool in the VisionSystemStep, be sure to account for all the acquisitions in the Job, as well as whether or not the Inspection step stores “failed” frames, then account for the depth of pipelining you will require per trigger. This will determine the appropriate size of the buffer pool. For example, three acquisitions with a image pipeline depth of two with the Inspection storing failed images will require:

$$(3 \text{ acqs} \times 2 \text{ trigs}) + 3 \text{ possible failed images (one failure per acq)}$$

buffers in the buffer pool to prevent process overrun errors from occurring.

Description

The acquisition definition of AcquireStep is edited through the Acquire properties page, as shown in Figure 1–19.

FIGURE 1–19. Acquire Properties Page

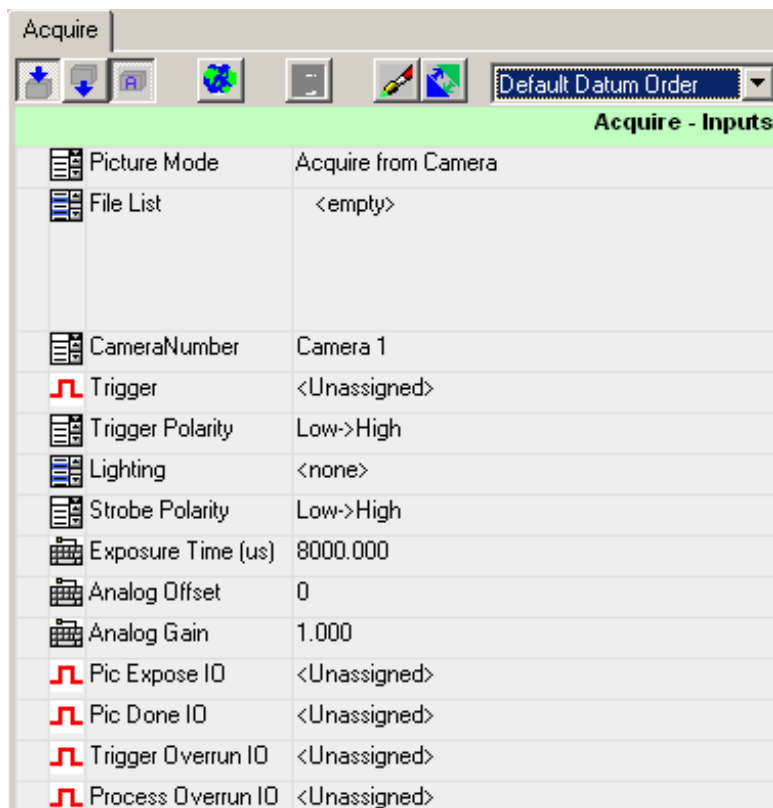


TABLE 1–4. Links to Property Descriptions

For Information About...	Go To...
Analog Gain	page 1–38
Analog Offset	page 1–38
CameraNumber	page 1–36
Exposure Time	page 1–38
External Strobe (GPIO 1)	page 1–38
File List	page 1–36
Lighting	page 1–37

TABLE 1–4. Links to Property Descriptions (continued)

For Information About...	Go To...
Options	page 1–39
Pic Done IO	page 1–39
Pic Expose IO	page 1–38
Picture Mode	page 1–36
Process Overrun IO	page 1–39
Trigger	page 1–36
Trigger Overrun IO	page 1–39
Trigger Polarity	page 1–37

Settings

- **Picture Mode** — Determines how images are acquired:
 - **Acquire from Camera** — Images are acquired from the camera.
 - **Load Images from File** — Uninitialized frames are retrieved from the hardware and the Acquire step fills the frame with image data from the disk. You can specify the TIF or BMP files in **File List**.
- **File List** — Lists the images to be used when the **Picture Mode** is Load Images From File. You can browse the disk and add images to the list, as well as re-order the images in the list.
- **CameraNumber** — For GigE cameras only. Allows you to select a camera.
- **Trigger** — Input I/O point used as the trigger for the acquisition. The trigger can be any input point supported by I/O, including general purpose I/O, sensors, software I/O, slave sensors, TTL inputs, RS422 inputs and serial inputs. This property lists the type and one-based index of the I/O point. Not all Camera I/O cards support every I/O point. For serial input triggers, select the device's UART or TCP port from the drop-down list and enter the trigger string in the edit box. The acquisition is triggered each time the trigger string is received on the port. The acquisition trigger can also be set to “Serial Trigger” to monitor a serial or TCP port for a triggered string.

To use serial triggering for acquisition, set the Trigger datum to “Serial Trigger,” select one of the device’s RS-232 or TCP ports from the dropdown list, and enter a trigger string in the trigger edit box as shown in Figure 1–20.

FIGURE 1–20. Serial Trigger for AcquisitionTrigger Strings**TABLE 1–5. Supported Escape Sequences**

Sequence	Output Character
\a	Bell (alert)
\b	Backspace
\f	Formfeed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\'	'Single quotation mark
\"	Double quotation mark
\\	Backslash
\ooo	ASCII character in octal notation
\xhhh	ASCII character in hexadecimal notation

For example, to trigger on the string only after <ENTER> is received, use the match string:

my trigger\n

When the Acquire step is set to serial trigger, tryout or runtime execution will pause on the acquisition until the trigger string is received. This can be disabled during tryout by turning off “Use Triggers During Tryout” in the Settings dialog box.

- **Trigger Polarity** — Determines the polarity of the strobe pulse (Low->High or High ->Low). When using serial triggering, either setting will work since the trigger IO will be set to high then immediately to low for each trigger event.
- **Lighting** — Allows specification of the light source powered by the camera:
 - Off — No power
 - On — Continuous strobe

- Strobe — Pulse equal to exposure duration
 - Power Strobe — Short 24V pulse
- **Strobe Polarity** — Is either of the following:
 - Low->High
 - High->Low
- **Exposure Time (us)** — This datum is only visible when using a progressive scan camera that supports electronic shutter, or a smart camera. Use this value to set the amount of time, in micro seconds, that the camera's CCD will integrate light. **Exposure Time** can freeze motion when not using a strobe, or if you are using a strobe, it can prevent ambient light from affecting your image.

Range: 1 to 1e+006 (Camera I/O 740 Card)
 64 to 50000us (smart camera)
- **Analog Offset** — Allows an offset to be applied to the incoming video signal. Offset can vary from:

Range: -128 to +127 (analog camera I/O cards)
 0 to 20 (smart camera)
- **Analog Gain** — Allows the gain of the camera I/O card to be applied to the incoming video signal. Gain can vary from:

Range: 0.80 to 2.17 (analog camera I/O cards)
 0.76 to 34.00 (smart camera)
- **Options** — Allows you to select a partial scan mode for the smart camera:
 - Partial Scan Half (648x227)
 - Partial Scan Quarter (648x81)
 - Partial Scan Add Size (324x242)
- **Pic Expose IO** — I/O point that is asserted when the camera has exposed the image. This I/O point is cleared when the image capture trigger occurs or when the capture is initiated as a non-triggered acquisition. It is set when the strobe is fired. If the acquisition is interlaced and non-strobed, the I/O point is set after the first field is captured.

Note: The image has not yet been transferred to hardware memory (ASIC or general), but has essentially been captured by the camera.

- **Pic Done IO** — I/O point that is asserted when the acquisition is complete and the image has been transferred to its proper location in memory (ASIC or general). This I/O point is cleared when the image capture trigger occurs or when the capture is initiated as a non-triggered acquisition. It is set when the image capture is complete and the image has been transferred to ASIC or general memory, or when an overrun occurs.
- **Trigger Overrun IO** — Output I/O point that is asserted when a trigger overrun occurs. You can select the type and index, including GPIO, software I/O, strobos, analog outputs, and TTL outputs.
- **Process Overrun IO** — Output I/O point that is asserted when a process overrun occurs. You can select the type and index, including GPIO, software I/O, strobos, analog outputs, and TTL.

Photometry

The full range of Photometry controls of the CCD can be programmed on the Acquire properties page. They can also be dynamically changed from within the avp by using a Perl Custom Tool to that effect or while the camera is running using the parameter connection part of the vskit programming framework (this functionality is not available in FrontRunner, though).

From the Acquire properties page, you can control the following parameters:

- **Exposure Time (us)** — from 64 usec to 50000 usec.
- **Analog Gain** — from 0.76 (no gain) to 34.0 maximum gain.
- **Analog Offset** — from 0 to 20 units.

The actual formula that controls the way the CCD signal is conditioned before being digitized into a pixel array is:

*CCD Signal (just before Digitized by ADC) in Volts =
Gain * (CCD Signal Raw in Volts) + Offset*

On a smart camera, Gain is a TRUE gain (i.e., a signal multiplier that is intuitive) rather than using raw HW units that are logarithmic in nature, as shown in Table 1–6. dB for the Gain in Table 1–6 will give you the conversion info if you need it since, typically, Gains are expressed in dB.

The Offset are in HW units in a smart camera. The offset has a limited range (0 to 20) and is primarily used to adjust the black level of the CCD. There should be no need to change this value from its default set by the avp.

Typically, the CCD dynamic range (Raw signal before Gain applied) is about 0.5V max for our design. However, the ADC (Analog to Digital Converter) has a 1V reference, i.e., digitize 8 bits (0->255) over 1V.

The Gain is an 8 bit value (0 to 255) with each unit being a 0.132dB change in gain (see Table 1–6).

Maximum Gain multiplier of about 37.0 (or 255 units).

In dB, note that about $6\text{dB} = 2X\text{Gain}$.

Of course, higher gain means higher noise, so practically, 15.0 (Vout/Vin) is about the maximum usable gain for this CCD.

TABLE 1–6. Gain Units Conversion

Vout/Vin	Gain in DB	Setting (dec)	Setting (hex)
1.00	0.00	18	12
1.40	2.92	40	28
2.02	6.11	64	40
2.42	7.68	76	4C
3.04	9.66	91	5B
4.06	12.17	110	6E
5.02	14.01	124	7C
6.02	15.59	136	88
7.01	16.91	146	92
8.04	18.11	155	9B
9.07	19.15	163	A3
10.09	20.08	170	AA
15.21	23.64	197	C5
20.00	26.02	215	D7

TABLE 1–6. Gain Units Conversion (continued)

Vout/Vin	Gain in DB	Setting (dec)	Setting (hex)
25.12	28.00	230	E6
30.14	29.58	242	F2
35.09	30.90	252	FC
36.73	31.30	255	FF

Default values are:

- Exposure time (us) — 16000 usec.
- Analog Gain — 2.5 default.
- Analog Offset — 0

Training

None.

Results

- Trigger Time Stamp — The time at which the trigger occurred.
- Acquire Frame Queue — The queue of frames waiting to be processed.
- Current Image File — In Acquire from File mode, this is the name of the tiff image file that is currently being processed.
- Current Image Index — This is the index into the list of image files that the Acquire Step accesses when in Acquire from File mode.

I/O Summary

None.

FlexArray Step

This step contains the FlexArrayDm, which contains a variable array of datum information. FlexArray Step is a placeholder only; it has no runtime and does not require training.

Other Steps Used

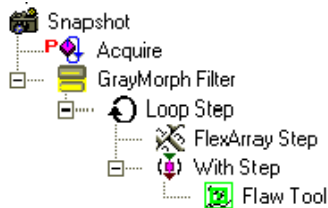
With Step — A FlexArray step will usually be used in conjunction with a With Step to provide the context switching of the data.

Theory of Operation

The connected With Step converts results recorded in the FlexArrayDm to the coordinate system of the FlexArrayStep. Therefore, the FlexArrayStep should be placed according to the desired coordinate system of the results.

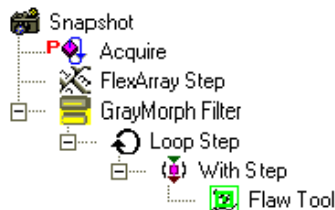
The results recorded into the FlexArrayStep will be in the coordinate system of the GrayMorph Filter's output buffer, as shown in Figure 1–21.

FIGURE 1–21. Example 1



The results will be in the coordinate system of the Snapshot output buffer, as shown in Figure 1–22.

FIGURE 1–22. Example 2



The FlexArray Step should not be placed inside of a coordinate system that will change due to a Loop or Trajectory Step.

Description

The FlexArray Step properties page has no properties.

Settings

None.

Training

None.

Results

None.

I/O Summary

None.

Formatted Output Step

This step generates a string that can be sent out a UART or TCP port or used elsewhere in the Job. The output string consists of a header string, a set of body strings, and a footer string. Each body string can be enabled or disabled by using a control expression. The final output string is a concatenation of these sub-strings.

Setting Console TTY to Zero

If Console TTY is set to 2, no inspection data is sent out the RS-232 port. You must set it to 0 (the default) to send data out the RS-232 port. Use the following procedure to set Console TTY to 0:

1. Open a HyperTerminal session by selecting
Start > Programs > Accessories > Communications > HyperTerminal.

Settings: 115200, 8, None, 1, None

2. While you hold down the Escape key, cycle power on the camera.
3. At the prompt, type `adv.`

At the `>>` prompt, type `BP_UpdateConsoleTTY (0 , 1 , 2) .`

4. Press the Return key until you see:

```
Console TTY:    2
```

5. Press:

```
0 <Return>
```

6. Press the Escape key. HyperTerminal displays the following message:

```
Save parameters to flash? [y/n] -
```

7. Press:

```
y <Return>
```

8. When you see the `>>` prompt, you can exit the HyperTerminal session.

Other Steps Used

None.

Theory of Operation

The Formatted Output step enables you to construct a string from a series of formatted sub-strings. Each sub-string consists of a format specifier, using “C” type conventions, and an argument list of datums or values.

The output string consists of an inspection prefix, a header string, a series of body sub-strings and a footer string. A checkbox on the property page selects whether to include the inspection prefix or not. When included, the inspection prefix contains the text '[Insp x]' where x is the inspection index. The output string always includes the header and footer sub-strings, which may be empty strings. The number of body sub-strings is user selectable, and each sub-string includes an associated expression to control whether or not to include the sub-string in the output.

The output string of the step is a combination of all the sub-strings appended in order. For example:

“Output String” = “Inspection Prefix” + “Output Header” + “Output String1” +
“Output String2” + ... + “Output String n ” + “Output Footer”

String Formatting

Each sub-string is formatted using the “C” language syntax. It consists of a quoted format string followed by a comma-separated list of arguments. The location and format of each argument is indicated by a format specifier '%' in the format string. The format string and arguments for each sub-string are entered in the edit box on the property page of the Formatted Output step (see Figure 1–23). A sub-string that displays “N/A” is empty and will not affect the output.

FIGURE 1–23. Formatted String Property Page Data Entry

Property	Value
Port Connection	NONE
TCP Address	N/A
Inspection ID Prefix	<input checked="" type="checkbox"/>
Number of Expressions	1
Output Header	N/A
Output1 Enable	1
Output1 String	"DMR content is %s\r\n",Insp1.Snapshot1.SYMTool1.Text Edit...
Output Trailer	N/A

The string format supports “C” language argument formatting syntax. Each argument to be included in the output string must have an associated format specifier of the format:

%[flags] [width] [.precision] type

The flags, width and precision parameters are optional, and the type specifier must match the argument type. In the output string, the format specifier is replaced by the formatted result. For example, the output string from the step shown in Figure 1–23 will be DMR content is: 123<CRLF>, where “123” are the contents of the Data Matrix and <CRLF> is the non-printable character sequence of carriage return and linefeed.

The parameters of the format specifier depend on the argument type:

TABLE 1–7. Floating Point Numbers

Flag	-	Left-align results in width
	+	Prefix number with sign (default for negative only)
	0	Pad with zeros instead of spaces to meet width requirement
	#	Force the output to include a decimal point
precision	The number of digits after the decimal point (default 6)	

TABLE 1-7. Floating Point Numbers (continued)

Type	f	Decimal output format.
	e or E	Exponential output format. The exponent is segmented from the mantissa by either an 'e' or 'E' depending on which is used for the type specifier.
	g or G	Output in f or e format (f or E format when G is specified). When the exponent is less than -4 or greater than or equal to the precision, the e format is used; otherwise the f format is used.
width	Minimum number of characters to output, padding with spaces as needed	

TABLE 1-8. Integers

Flag	-	Left-align results in width
	+	Prefix number with sign (types d or i only)
	#	Force the output to include a decimal point
precision	Minimum number of digits to output, padding with zeros as needed	
Type	d or i	Signed decimal integer.
	u	Unsigned decimal integer.
	o	Unsigned octal integer.
	x	Unsigned hexadecimal integer, using "abcdef".
	X	Unsigned hexadecimal integer, using "ABCDEF".
width	Minimum number of characters to output, padding with spaces as needed.	

TABLE 1-9. Strings

Flag	-	Left-align results in width
precision	Limits the number of characters to output, truncating to fit.	
Type	s	The argument string is output.
width	Minimum number of characters to output, padding with spaces as needed.	

TABLE 1-10. Binary Integers

Flag	+	Output data as Unsigned
type	p	This type specifier indicates the IntDm argument should be transmitted in binary format to the serial connection.
width	1,2,4	Specifies the number of bytes output. Output is the lowest order set of bytes for the given precision in big-endian byte order. Default precision is 4 bytes if unspecified.

TABLE 1–11. Binary Image Buffer

type	p	This type specifier indicates that a BufferDm may be selected for output, and that its raw pixel data should be transmitted in binary format.
width, height	w, h	By default, the entire buffer will be transmitted. You may choose to specify a width and height, and the buffer will be scaled down to your specified size.

When using %p to transmit an image buffer, the entire buffer will be transmitted by default. The format of the data will be as follows:

Bytes	Data
4	Buffer Width
4	Buffer Height
4	Number of Row Bytes (buffers are always 4-byte aligned, so the number of bytes per row may be different than the buffer width. This is important if you wish to create a bitmap).
<image size>	Pixel data

Optionally, you may specify an image width and height, and the image will be scaled down to your specified size.

%www.hhhp where ‘www’ is your desired width, and ‘hhh’ is your desired height.

Use the following format string to get an image that is 160x120:

%160.120p

When using this format, the output data will appear as:

Bytes	Data
4	Buffer Width
4	Buffer Height
<image size>	Pixel data

TABLE 1–12. Binary Status

type	p	This type specifier indicates the StatusDm argument should be transmitted in binary format to the serial connection. Output is a byte value of 1 for true status or 0 for false.
------	---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

TABLE 1–13. Binary String

type	p	This type specifier indicates the StringDm argument should be transmitted in binary format. Input is a string of hexadecimal string pairs. Output is a string of binary characters half the length of the input string. This allows for non-printable characters to be transmitted over serial connections.
------	---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Non-Printable Characters

The string format also supports “C” language escape sequences for non-printable characters. A carriage return and line feed (CRLF) sequence is represented by `\r\n` within the quoted string format. Table 1–14 contains the complete list of supported escape sequences.

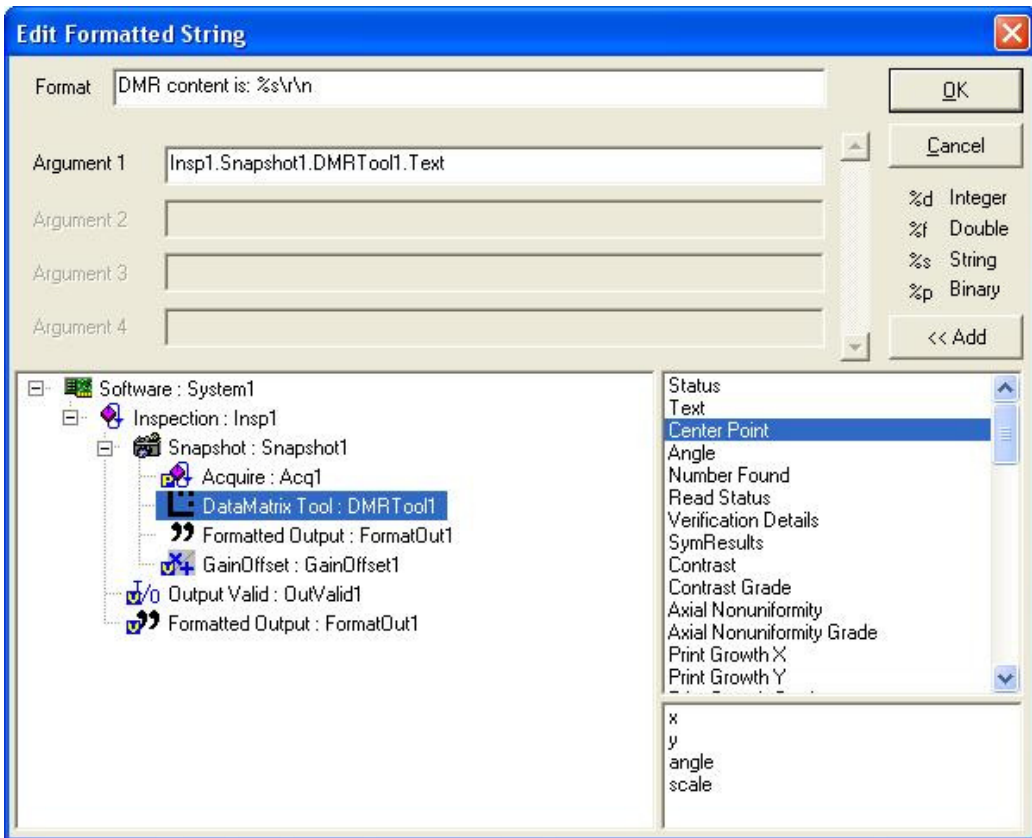
TABLE 1–14. Supported Escape Sequences

Sequence	Output Character
<code>\a</code>	Bell (alert)
<code>\b</code>	Backspace
<code>\f</code>	Formfeed
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\</code>	'Single quotation mark
<code>\"</code>	Double quotation mark
<code>\\</code>	Backslash
<code>\ooo</code>	ASCII character in octal notation
<code>\xhhh</code>	ASCII character in hexadecimal notation

Formatted String Editor

In addition to entering the format specifier string and argument list in the edit box on the property page, you can enter the same information using the Formatted String Editor (Figure 1–24) by clicking on Edit. This editor allows you to enter arguments by browsing the Job Tree and datum lists. It also separates the format string and each argument for easier reading.

FIGURE 1–24. Formatted String Editor



The number of argument edit boxes that are visible depends on the number of format specifiers in the format string. Enter a format specifier to make its associate argument edit box visible before entering its argument. Quotation marks do not need to be included when entering the format string in this editor; they will be added when you click OK and the format string and arguments are converted into a comma separated list.

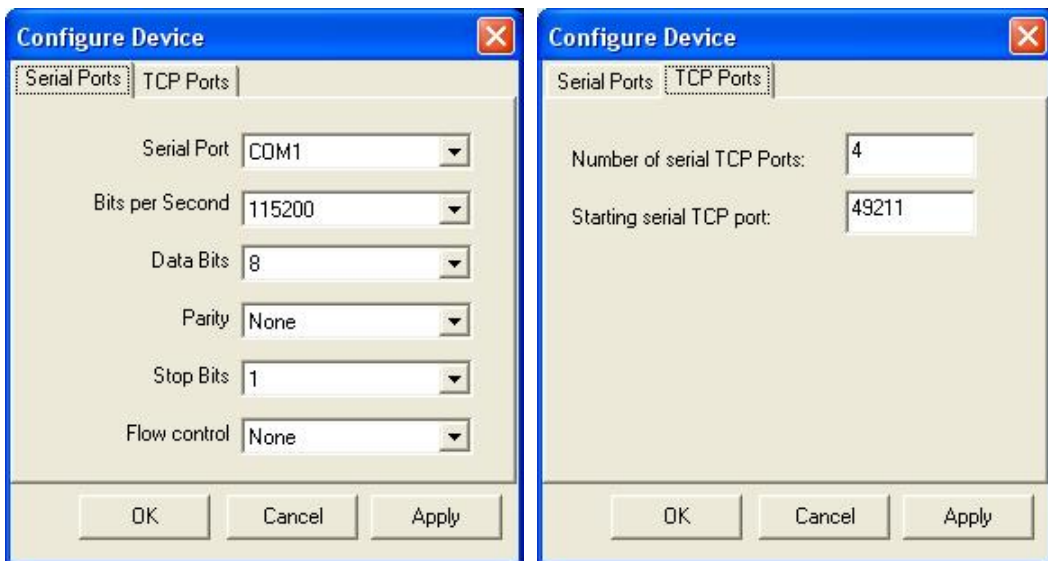
To enter an argument, click on the desired step in the Step Tree and choose a datum from the list on the right, optionally choosing a component datum from the bottom right pane. The datum type must match the format specifier type. For example, to output the x coordinate of the center of a matrix, the format specifier is numeric '%f' and the argument will be the x component of the Center Point output. Angle, Area and Distance arguments can also be used with numeric format specifiers (%f). Arguments for numeric types can include mathematical expressions.

Port Configuration

The serial UART settings and TCP port range can be configured using the Configure Device setup dialog, as shown in Figure 1–25. To open the configuration dialog, from FrontRunner, select File > Configure Device....

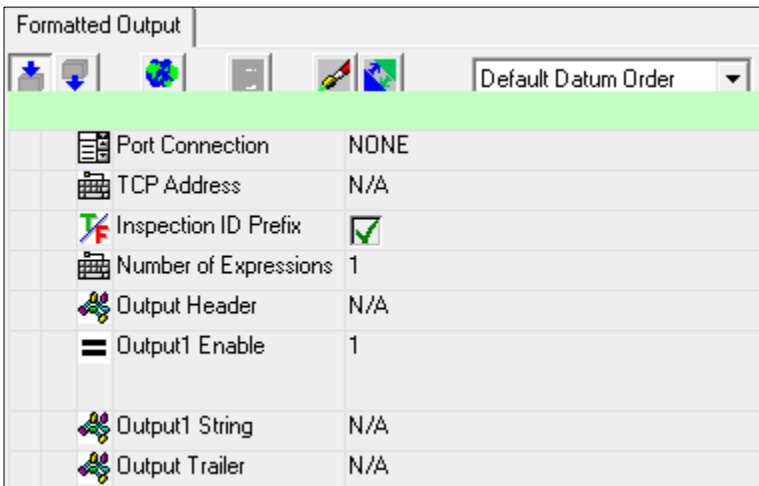
When choosing a Port Connection on the Formatted Output Step properties page, TCP1 will correspond to the Starting serial TCP Port specified in this dialog.

FIGURE 1–25. Device Configuration Dialog



Description

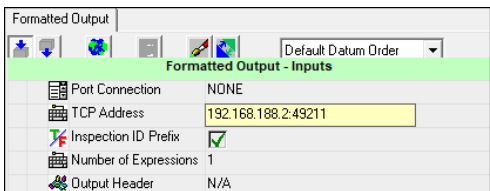
The Formatted Output Step allows editing through the Formatted Output Step properties page, as shown in Figure 1–26.

FIGURE 1–26. Formatted Output Step Properties Page

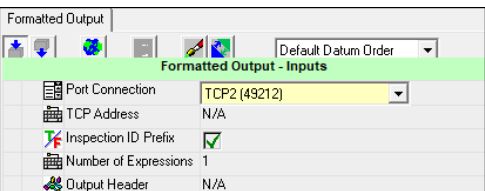
Settings

- **Port Connection** — Specifies the serial port on which the output string will be transmitted. It can be a UART or TCP port. The COM ports are indexed to the target platform’s available UARTs. The TCP ports are indexed according to the port settings specified in the Configure Device settings (select Configure Device... from the File menu of FrontRunner). For example, if the “Starting Serial TCP Port” is 49211, then selecting TCP1 will output on port 49211, TPC2 on port 49212, etc. The default is NONE, which will construct the output string datum but not transmit the result on any port.
- **TCP Address** — Enables the camera to be used as a client for pushing data to a server or to another camera on the network (which can act as a server). This string datum specifies the TCP/IP address of the server and port to connect to on the local network to transmit the output string. The address and port is entered using a colon separator character: “192.168.188.2:49211”.

TCP Client Mode



TCP Server Mode



- **Inspection ID Prefix** — When this box is checked, the output string will be prefixed by the text “[INSPx]” where x is the inspection index. This allows the program receiving the output from this step to parse the text and identify the inspection from which it is being sent.

- **Number of Expressions** — The number of body strings contained in this step. Each string has an associated Enable expression.
Default: 1
Range: 1 to 100
- **Output Header** — This formatted string is always output ahead of the body of sub-strings.
- **Output Enable** — The enabling expression for the associated sub-string. Each expression is calculated at runtime; if the result of the expression is non-zero, the associated sub-string is appended to the output string. If the expression result is zero, the associated sub-string is not included in the output. The default is 1, which will always include the sub-string.
- **Output String** — Formatted sub-string that is appended to the output string if the associated enabling expression is non-zero. Contains a “C” style formatted string expression and argument list. The datum should include a quoted format string followed by a series of arguments of the format:
“<format string>”, <argument 1>, <argument 2>, ...
- **Output Trailer** — This formatted string is always added to the end of the body of sub-strings.

Training

None.

Results

The completed output string is available via the output datum **Output String**; it will also be transmitted out the port specified via **Port Connection**.

I/O Summary

The Formatted Output Step provides an I/O summary in the Status Bar located at the bottom of the Train Window.

Inputs: Compile Error (yy): xxx

Outputs: <string>

Where: xxx = description of compilation error or OK

yy = error code

<string> = the combined output string result

If Step

This step provides a means of flow control. You can create a boolean expression and, if the expression evaluates to TRUE (non-zero), then the children of the step are executed. There are no specific inputs or outputs of this step. You can also decide whether or not the If Step itself fails with the expression.

Other Steps Used

None.

Theory of Operation

The If Step is a flow control container. This step contains an expression datum:

- If the expression evaluates to **non-zero**, then the children of the step are executed normally.
- If the expression evaluates to **zero**, the children of the step are not executed, and execution passes on to the next sibling of the IfStep.

The step also has a StatusDm selection, “Fail With Expression,” that, if set, causes the step to fail when the expression evaluates to zero.

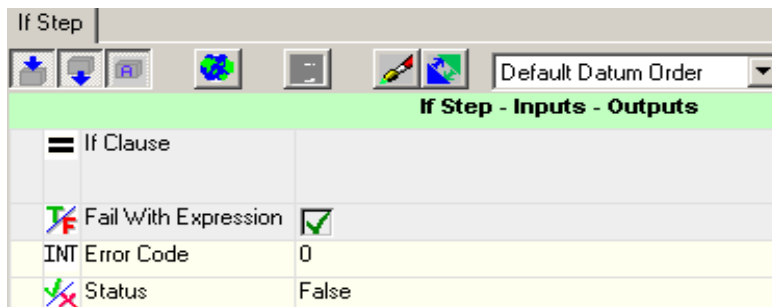
Description

There is no explicit user interface for the If Step. You can edit the expression datum of the step in the Data View window.

Settings

The If Step contains only one expression datum, named “IfClause” (see Figure 1–27). This datum is an expression datum which, when evaluated, determines the execution of the children of the If Step.

FIGURE 1–27. If Step



You can enter any expression into the datum (see “Expressions” on page 8-59 for details on expressions). You can also enable/disable Fail With Expression.

Training

The If Step is not trainable. The expression’s variable references are resolved when you set them. If there are errors, you are prompted to fix them. If subsequent altering of the vision program causes the expression to be invalid, the step will not be able to run.

Results

The results of the step are optionally defined by the contained expression:

- If the expression evaluates to **non-zero**, the tree executes normally.
- If the expression evaluates to **zero**, then any steps contained within the If Step are not executed.

If Fail With Expression is set, then the Step itself fails when the expression fails.

I/O Summary

The I/O summary is either “OK”, or a string indicating the compilation error of the IFClause expression.

Inspection Step

This step contains all steps involved in the image capture and processing in a single thread that are enclosed within the step. When the Inspection step is run, it outputs the overall status of the operations.

Other Steps Used

- **Formatted Output** — This step is executed at the end of an inspection cycle and can be used to send an inspection result string to a serial port.
- **Output Valid** — This step selects an IO point to signal at the completion of an inspection cycle.

Theory of Operation

The Inspection step groups other steps into an inspection set. It is analogous to a container for all steps comprised of a particular inspection. Programs must contain at least one Inspection step but may contain as many as necessary.

Inspection makes result and statistical inspection data available to the host. You can set up the specific results. The Inspection step allows you to set up the criteria that determines whether an inspection passes or fails.

Inspection has no explicit user interface. You enter data for each of the inspection properties in the Inspection properties page.

Description

Inspection allows editing through the Inspection properties page, as shown in Figure 1–28.

FIGURE 1–28. Inspection Properties Page





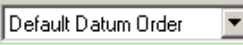























Inspection		Snapshot	Output Valid	Formatted Output
    				
Inspection - Inputs				
	Use Default Pass Criteria	<input checked="" type="checkbox"/>		
	Criteria for Inspection Pass			
	Busy Signal IO	<Unassigned>		
	Minimum Busy Signal Duration (ms)	0		
	Busy Signal Polarity	Low		
	Part Queue Enabled	<input type="checkbox"/>		
	Part Queue Storage Mode	Store No Images (Just Results)		
	Part Queue Qualifier	1		
	Part Queue Image Graphics	Include Graphics		
	Part Queue Size (Cycles)	0		
	Record Entered Into Queue IO	<Unassigned>		
	Part Queue Almost Full IO	<Unassigned>		
	Part Queue Full IO	<Unassigned>		
	Select Results to Upload			
	Keep Last Failed Data	<input checked="" type="checkbox"/>		
	Status Output	<Unassigned>		
	Insp Step Priority	Normal		
	Use Processor At Runtime	<none>		
	Inspection Timeout (msec)	0		
	Timeout Type	Cycle Time		
	Ready to Run Output	<Unassigned>		
	Results Upload Qualified Condition	1		
	Freeze Qualified Condition			

TABLE 1–15. Links to Property Descriptions

For Information About...	Go To...
Action When Part Queue Full	page 1–61
Busy Signal IO	page 1–59
Busy Signal Polarity	page 1–59
Criteria for Inspection Pass	page 1–59
Freeze Qualified Condition	page 1–63
Insp Step Priority	page 1–62
Inspection Timeout (msec)	page 1–62
Keep Last Failed Data	page 1–61
Minimum Busy Signal Duration (ms)	page 1–59
Part Queue Almost Full IO	page 1–61
Part Queue Enabled	page 1–59
Part Queue Full IO	page 1–61
Part Queue Image Graphics	page 1–60
Part Queue Qualifier	page 1–60
Part Queue Size (Cycles)	page 1–60
Part Queue Storage Mode	page 1–59
Ready to Run Output	page 1–63
Record Entered Into Queue IO	page 1–60
Results Upload Qualified Condition	page 1–63
Select Results to Upload	page 1–61
Status Output	page 1–62
Timeout Type	page 1–63
Use Default Pass Criteria	page 1–59
Use Processor At Runtime	page 1–62

Settings

- **Use Default Pass Criteria** — Enables/disables the use of the user-specified criteria. By default, every step must pass in order for the inspection to be considered a pass inspection. When disabled (not checked), the default criteria is overridden by **Criteria for Inspection Pass**. When enabled, the default criteria is used.

Default: Enabled

- **Criteria for Inspection Pass** — Sets the criteria for an inspection to pass. When **Use Default Pass Criteria** is disabled, and **Edit...** is clicked, the expression entered into the text box is evaluated at the end of each inspection. When the expression evaluates to TRUE, the inspection passes.

Default: blank

- **Busy Signal IO** — Selects a physical or virtual I/O point to be used as a busy signal. The busy signal indicates that the inspection is in progress, and can be used to handshake with external equipment. If the inspection is triggered, the busy signal does not activate until a trigger has been received and processing is about to begin. The default is <none>, indicating that there is no busy signal. Other selections include digital and virtual I/O points available on the GigE Camera or a smart camera.
- **Part Queue Enabled** — Enables or disables the Part Queue. For complete information about the Part Queue, see Chapter 4 of the Visionscape FrontRunner User Manual.
- **Minimum Busy Signal Duration** — Specifies a minimum length (in milliseconds) for the busy signal. This length is applied both to the time the signal is active and the time it is Off. In this manner, it can be guaranteed that equipment polling the signal will not miss any transitions.

Default: 0 msec

- **Busy Signal Polarity** — Specifies the polarity of the busy signal. By default, the signal is Low during inspection processing. Settings are High and Low.
- **Part Queue Storage Mode** — Selects when camera images are to be stored in CPU memory every cycle:
 - **Store All Images** — Saves all camera images in the inspection.

- Store No Images (Default) — Saves no camera images in the inspection.
- Store Failed Images — Saves only those camera images in the inspection when the part fails.
- Store Passed Images — Saves only those camera images in the inspection when the part passes.
- Store Qualified Images — When you select this option, it looks to the expression that you specify in **Part Queue Qualifier**. If **Part Queue Qualifier** evaluates to TRUE, then the image is saved in the Queue; if it evaluates to FALSE, then it is not.

When an inspection runs and meets the image storage criteria, the inspection step saves all camera images in CPU memory for that part, which can then be stored to be reviewed at a later time.

- **Part Queue Qualifier** — Allows you to specify the expression that determines whether or not to store a graphic in the queue. For more information about expressions, see “Expressions” starting on page 8-59.
- **Part Queue Image Graphics** — Allows you to specify whether or not to store graphics with the images for each record in the queue. Drawing graphics takes time in the Inspection cycle, so you can disable this to prevent graphics from being drawn.

Default: Include Graphics

- **Part Queue Size (Cycles)** — Specifies the number of images to be stored by the Inspection Step. Based on **Part Image Storage Mode**, images are stored in a first-in, first-out queue. When the number of images stored reaches the size specified by this parameter, the oldest image in the queue is overwritten so that the queue never expands beyond this size.

Default: 0

- **Record Entered Into Queue IO** — Selects a virtual I/O point from the drop-down list, forcing the Inspection Step to set that point when an image is entered into the part image queue. You use this feature to keep track of the queue size at any point in time.

Default: <none>

- **Part Queue Almost Full IO** — Selects a virtual I/O point from the drop-down list, forcing the Inspection Step to set that point when there is one less than the maximum number of images in the part image queue. You can monitor this point to know when the queue is almost full, and can stop running inspections before any images are discarded.

Default: <none>

- **Part Queue Full IO** -- Specifies an IO point to signal when the Part Queue is full.
- **Action When Part Queue Full** — Tells the Inspection what action to take when the Part Queue is full. The Part Queue is a ring buffer of inspection records. This means when the queue is full, by default, the oldest data is thrown out to make room for the newest data. You can change this functionality using this property. If the property is set to “Block and Wait For Upload,” then when the queue is full, the Inspection blocks until the queue is uploaded and emptied. This guarantees that data is not dropped from the queue when it’s full.

Default: Don’t Block; Reuse Image Records

- **Select Results to Upload** — Selects which inspection results to upload to the host at the end of an inspection for which the **Result Upload Mode** is valid. By default, no inspection results are uploaded. A list of every available output value appears in the drop-down list after you have created and trained your Job. Click a particular list member to activate it as part of the set to be uploaded. Click it a second time to deactivate it. The buttons along the bottom of the dialog box facilitate large result manipulations, as described next:
 - **Add** — Displays the Select Datums For Report dialog for adding results to the upload list.
 - **Clear** — Removes all items from the upload list.
 - **Full Path** — When selected, the list displays the complete datum name including parent steps.
 - **Remove** — Removes the selected items from the upload list.
- **Keep Last Failed Data** — When enabled, all images contributing to the last inspection failure are stored in image memory; they can be displayed in realtime.

Default: Enabled

- **Status Output** — The I/O point selected from the drop-down list reflects the status of the inspection. Upon completion of the inspection, the results are evaluated and compared to the user-specified or default pass criteria, and the designated I/O point is set high if the criteria is met.

Default: <none>

- **Insp Step Priority** — Raises or lowers the priority of an inspection in the Job. If two or more inspections are waiting for the same resource, then, when that resource becomes available, the inspection with the highest priority will run:
 - **Lower** — The inspection has lower priority than any inspection set to Normal or Higher priority and the same as one with Lower priority.
 - **Normal (Default)** — The inspection has lower priority than any inspection set to Higher priority, the same as one with Normal priority, and higher priority than one set to Lower priority.
 - **Higher** — The inspection has higher priority than any inspection set to Lower or Normal priority and the same as one with Higher priority.
- **Use Processor At Runtime** — Indicates which processor to use at runtime for running the inspection. The number of processors available to the current process is listed and the processor(s) to use are selected. If no processors are selected, then any processor can be used.
- **Inspection Timeout (msec)** — When set to a value other than 0, this setting will force the inspection to time out after the set number of milliseconds. The elapsed time is checked before the execution of each step in the inspection, and if the set time has expired, the remaining steps fail and the inspection quits. This is either a maximum cycle time or process time, depending on the setting for the Timeout Type. Note the following two characteristics of this timeout:
 - The timeout granularity is based on the processing time of the tools in the inspection. Since the timeout is only checked between each tool, it may be detected later than what is set. This can be either a slight delay or a longer one, depending on which tool is running at the time the timeout expires.

- To avoid out of sequence inspections and/or buffer memory leaks, there are different methods of handling inspections with multiple snapshots. If there are triggered snapshots pending when the timeout occurs, they will be executed so that the corresponding trigger and buffer are not lost. This could potentially cause a delay in posting the timeout if these triggers have not yet occurred. If there are non-triggered snapshots pending when the timeout occurs, they will not be executed because there is no danger of missing a trigger or losing a buffer. In this case, there will be little or no delay in posting the timeout.

Default: 0 (no timeout)

- **Timeout Type** — When the Inspection Timeout is set to a value other than 0, this setting determines the starting point for the timeout. The choices for the type of timeout are:
 - **Cycle Time (Default)** — The beginning of the cycle is defined by the starting trigger, I/O input to the inspection, or by the start of the inspection if there are no trigger events in the inspection.
 - **Process Time** — The beginning of processing is defined by the time at which the first vision tool begins to run. This setting allows the variable image acquisition time to be eliminated from the timeout calculation.
- **Ready to Run Output** — Selects an IO to be activated while an inspection is running.

Default: <none>

- **Results Upload Qualified Condition** — This expression is evaluated after the Inspection has run to determine whether or not to upload the image(s) to the PC. If non-zero, the images can be uploaded to the PC. This expression overrides any freeze-mode setting.

Default: 1

- **Freeze Qualified Condition** — The Image Upload mechanism has a new freeze mode, “Freeze Next Qualified”. When this mode is selected, the evaluated value of this property indicates when the “freeze” criteria has been met. When signaled, the image(s) are uploaded and the mode changes to “Freeze This”.

Default: Blank

Training

None.

Results

All steps contained in the inspection are run. At the end of the run:

- Inspection results are sent to the host under the appropriate conditions.
- The status of the inspection is set.
- Statistics are updated.

I/O Summary

Inspection provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs:

- When Part Image Storage Mode is not Store No Images, and Part Image Queue Size > 0, and Results Upload Mode is any selection other than Never, and Select Results to Upload has no selection, the line reads:
Store: xxx Images Results: yyy selected
Where: xxx = Part Image Queue Size
yyy = Number of Select Results to Upload
- When Part Image Storage Mode is not Store No Images, and Part Image Queue Size > 0, and Results Upload Mode is set to Never, or Select Results to Upload has no selection, the line reads:
Store: xxx Images Results: none
Where: xxx = Part Image Queue Size
- When Part Image Storage Mode is set to Store No Images, or Part Image Queue Size = 0, and Keep Last Failed Data is enabled, and Results Upload Mode is set to Never, or Select Results to Upload has no selection, the line reads:
Store: Last Failure Results: none
- When Part Image Storage Mode is set to Store No Images, or Part Image Queue Size = 0, and Keep Last Failed Data is disabled, and Results Upload Mode is set to Never, or Select Results to Upload has no selection, the line reads:
Store: none Results: none

Outputs: None

Job Step

This step is the root of all steps in the tree. It is the parent of all Vision System steps and provides a simple entry point to the tree as its root. The JobStep is used:

- To denote the “root” of the Step Tree.
- For Job persistence; that is, the JobStep loads and saves vision Jobs to and from the hard disk.
- To maintain the vision Job in memory.

The user/programmer always creates a JobStep as the root of the tree, then inserts VisionSystemSteps into the Job for the various sets of hardware.

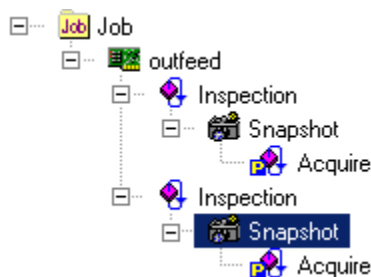
Other Steps Used

None.

Theory of Operation

The Job step, as shown in Figure 1–29, provides a common root to the step Tree for all targets. It is created once and contains all targets for the tree.

FIGURE 1–29. Job Step



The Job step is not used by the target vision system. That is, the Job step is never downloaded to the target. The Vision System step is downloaded, saved to disk, and loaded from file. Jobs maintain the Vision System Step.

Description

The Job step has no properties page.

Settings

None.

Training

None.

Results

None.

I/O Summary

None.

Loop Step

This step provides looping of a set of child steps. **Loop Count** specifies the number of times to run the group of child steps. **Iteration Index** is incremented each time, and can be used by child steps for conditional processing based on which run of the loop is executing.

By itself, the Loop Step only provides the execution control of the child steps. In order to provide different parameters for each iteration or result recording for steps within the loop, the Loop Step can be combined with the FlexArray Step and With Step. This will provide similar functionality to the Trajectory Step without all the components of the Trajectory step, such as the Trajectory Shape and Locator Step.

Other Steps Used

None.

Theory of Operation

When the Loop Step runs, **Iteration Index** is reset to 0. For each iteration, the Exit Loop I/O is evaluated to determine if the Loop Step should continue to execute. The child steps of the Loop Step are run normally, as they would if they were not inside a Loop Step. Then, **Iteration Index** is incremented and the next iteration runs. This continues until the **Iteration Index** reaches Loop Count - 1. Then, the status of the Loop Step is set to true only if all steps of every iteration also pass.

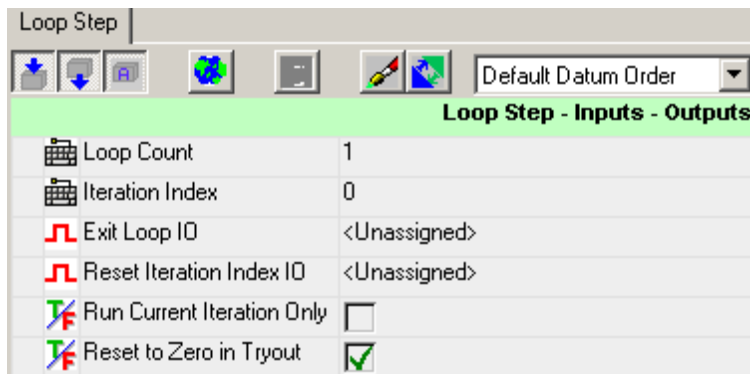
When **Run Current Iteration Only** is set to true, **Iteration Index** is not reset to 0 or incremented. The child steps are only run once. This can be useful when analyzing the results from one specific iteration, by allowing a single iteration to be run each time.

When inspecting for an overall pass/fail only, Exit Loop I/O can abort subsequent iterations of the loop once an error has been found. This can be done with an I/O Output step, with the expression set to the desired step's status output.

Description

Loop Step allows editing through the Loop Step properties page, as shown in Figure 1–30.

FIGURE 1–30. Loop Step Properties Page



Settings

- Loop Count** — Specifies the number of times to run the group of child steps.
 Default: 1
- Iteration Index** — Specifies the active loop index. When the step is run, this property is set to 0, then incremented for each time the set of child steps are run. During setup mode, you can set this value.
 Default: 0
- Exit Loop IO** — Specifies an I/O number to be examined at the start of each iteration. If the value of the I/O is true, the remaining loop iterations are skipped, and program execution continues with the step immediately following the loop step.

- **Reset Iteration Index IO** — When this Input turns ON, the Loop step's iteration index will be reset to 0. This provides you with a method for resetting the loop (if needed).
- **Run Current Iteration Only** — When enabled, the set of sub-steps will be run only once, and **Iteration Index** will not be reset or incremented.

Default: Disabled

- **Reset to Zero in Tryout** — If checked, then after running this step in Setup mode (does not apply to run mode), the iteration index will be reset to 0 and any child steps will be run again.

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.

I/O Summary

None.

Sequence Step

This step is a container whose Status reflects the pass/fail status of all its contained steps.

Other Steps Used

Output Valid Step — Used by this step in order to provide an Output Data Valid signal of programmable duration.

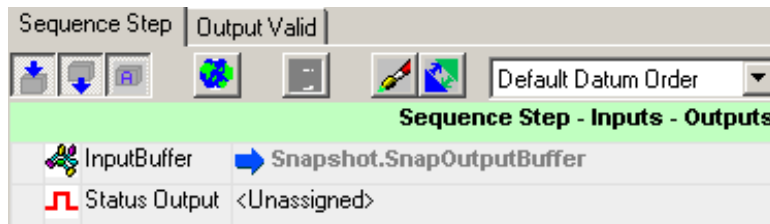
Theory of Operation

The Sequence step is a container whose Status reflects the pass/fail status of all its contained steps. Like the Inspection, the Sequence executes all its contained steps first, and then combines all the Status results of the contained steps as its own Status. However, this step does not execute under its own thread. This result can also be set to an I/O point.

Description

Sequence Step allows editing through the Sequence Step properties page, as shown in Figure 1–31.

FIGURE 1–31. Sequence Step Properties Page



Settings

- **Status Output** — Indicates which I/O point reflects this step's status.

Note: There is also an Output Valid page in order to set up the Output Data Valid Signal.

Training

None.

Results

The Status signal that has been described above.

I/O Summary

None.

Snapshot Step

This step is a container for steps that analyze and process images. Snapshot uses the Acquire step to capture an image from either a camera or an image file, then makes this image available to its contained steps for subsequent processing. Additionally, Snapshot applies a user-defined gain and offset to the image, which allows for simple brightness and contrast adjustment. You can accomplish this with the embedded GainOffset Step in the software, or with the Analog Gain/Offset settings in the hardware. Refer to “GainOffset Filter” on page 13-11 for more information.

Other Steps Used

Acquire — Automatically inserted as a component of Snapshot. There is always a one-to-one relationship between Acquire and Snapshot. Refer to “Acquire Step” on page 1-32 for more information.

Theory of Operation

Snapshot and Acquire coordinate their operations through a frame datum list or queue. The frame datum list is a list containing images with additional timing and status information. The AcquireStep contains the parameters required to set up an acquisition. It works with the hardware to generate image frames that it pumps into the frame datum list. Snapshot, in turn, pulls the image frames from the list and passes them on to the vision tools in the inspection for further processing.

When steps are inserted into a Snapshot, all their input buffer datums are automatically connected to the output buffer datum of Snapshot. The output buffer datum of Snapshot is the last image acquired. These steps then process or analyze this image.

Acquire defines the acquisition and passes it to the camera I/O card. The acquisitions can be defined as triggered or non-triggered. In either case, the camera I/O card or smart camera manages the acquisition and the creation of frames for each image. There are two distinct modes of operation:

- **Non-Triggered** — As a normal pre-processing component step of Snapshot, Acquire is run before Snapshot is run. Acquire starts an acquisition on the hardware, then queues the frame to the Snapshot. Snapshot waits on the completion of the frame and makes it its output buffer datum. Then, control passes to the post-processing steps that process the image, execute vision, and produce results. In this mode, Acquire does not run again until the entire Inspection has completed and starts again.

- **Triggered** — Acquire adds a triggered acquisition to the hardware and receives a frame when the trigger occurs. The hardware pipelines the frame to Acquire, which in turn pipelines the images to the Snapshot. This allows the hardware to immediately respond to the next trigger, and allows Acquire to immediately respond to the result of the next trigger as soon as it has queued the frame of the previous trigger. The next image can be acquired while the previous image is being processed. Acquire and Snapshot always coordinate through the frame datum list. This threaded approach allows the acquisition subsystem to capture images at very high speeds. There are a number of error conditions (see Table 1–16) that can arise during acquisition available through **SnapStatus**, which is the result of the Snapshot.

TABLE 1–16. Error Codes

Error Code	Comment	Description
0	Done	The acquisition completed successfully.
1	Waiting	The acquisition is still active, waiting for DMA completion.
2	Empty	No acquisition occurred at all.
3	Timeout	The acquisition on the camera I/O card did not complete in the required time.
4	Rpc time out	Communication with the GigE Camera or smart camera timed out. May require rebooting the GigE Camera or smart camera to recover.
5	Trigger Overrun	Triggers arrive too frequently for Acquire to process. The maximum achievable rates for interlaced TV format cameras are 30 fps NTSC or 25 fps PAL. For digital cameras, the frame rate is a function of the pixel rate and exposure time.
6	Process Overrun	The images were not processed fast enough. The system has limited set of image buffers in the bufferpool. When images are processed, image buffers are freed for reuse. When the buffers are not returned to the bufferpool fast enough (because the processing takes too long), the acquisition process fails. That is, there are no more buffers available in the bufferpool for the acquisition to complete. You can increase the size of the bufferpool or slow down the triggers to a rate acceptable for inspection processing to recover from this error.
7	Aborted	The acquisition was aborted.
8	DMA done	The acquisition completed successfully.

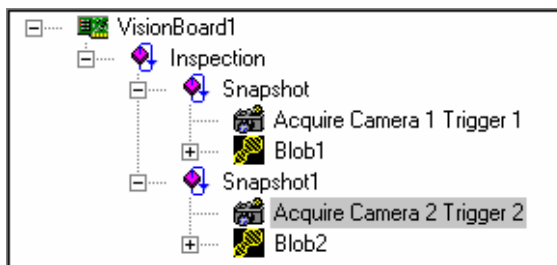
TABLE 1-16. Error Codes (continued)

Error Code	Comment	Description
9	DMA time out	The acquisition on the camera I/O card did not complete in the required time.
10	BadFile	The image file specified does not exist or could not be accessed.
11	EmptyList	There are no image files specified at all.
12	FIFOOverrun	The PCI bus bandwidth is exhausted and pixels have been dropped.

Sequential Acquisition

The Job shown in Figure 1-32 demonstrates a two-camera inspection using two different triggers. An example would be the inspection of a single part that is moved to another location under a second camera.

FIGURE 1-32. Job — Two Snapshot Acquisition

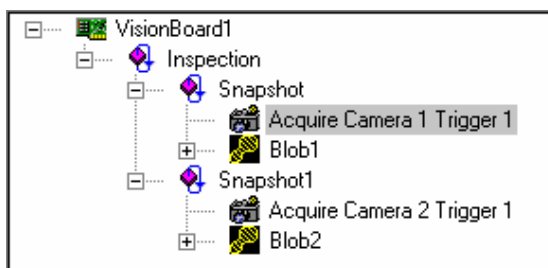


The Job expects the triggers to arrive in a pre-determined order: Trigger 1, then Trigger 2. When Trigger 1 or Trigger 2 asserts, acquisitions are executed and pipelined to the specific Snapshot steps. However, the images are always processed in order of trigger 1 image, then trigger 2 image. Also, if the triggers are not signaled properly, then the buffer pool can be quickly consumed as the various Snapshot steps are not processing the image appropriately.

Synchronous Acquisition

Using strobe lighting, it is possible to capture images from multiple cameras at the same moment in time. Synchronous acquisition with multiple cameras is enabled by configuring the Acquire steps to trigger off the same trigger input. Each acquisition can react to the same trigger event. The image in each camera is frozen until the camera I/O card becomes available to capture it. The Job, as shown in Figure 1–33, demonstrates synchronous capture inside a single inspection.

FIGURE 1–33. Job — Synchronous Capture Inside Single Inspection

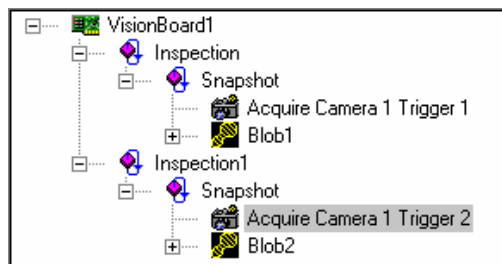


To set up a synchronous Job:

1. Enable the same trigger in each Acquire step for all cameras.
2. Assign different cameras for each Acquire step. The cameras numbers assigned can be in any order.
3. Enable the strobe(s) for all cameras in each Acquire step.
4. Train your Job and proceed as you normally would to build a Job.

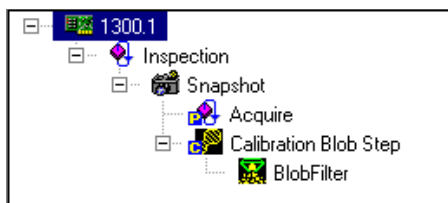
Asynchronous Acquisition

Asynchronous acquisition is required when the system has to support multiple independent inspections, and images must be taken in any order at any time. An example is using the inspection system to inspect parts on two separate production lines that are not synchronized and run at different part rates. To support asynchronous acquisition, multiple Inspection Steps are necessary to define different execution threads, one for each independent inspection process. Because the inspections are independent, the executions do not block each other and can process the images appropriately. The Job shown in Figure 1–34 contains two independent inspection processes. Acquire steps share the single camera I/O card resource.

FIGURE 1–34. Job — Asynchronous Acquisition

Calibration

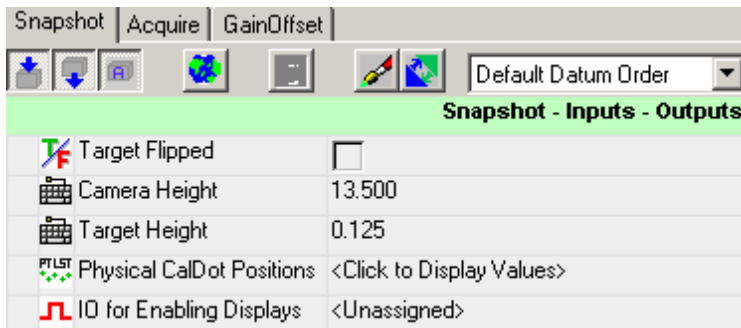
The SnapshotStep also defines a point of calibration in the Step Tree. When calibrated, the Snapshot contains a special Part tree that defines the calibration data, as shown in Figure 1–35:

FIGURE 1–35. Special Part Tree

A typical calibration Job is simply a Blob Step. The Blob Step is used by the Calibration Manager to find the calibration blobs in the image and update its “PhysCalDots” Point List datum, calculate the calibration matrices, then update the Calibration Result Datum in the Snapshot. The Calibration Result Datum (“CalResult”) contains the mean and max residuals, the pixels per unit and units per pixel in x and y, the camera angle, and the UX and VY perspectives.

Description

Snapshot can be edited through the Snapshot properties page, as shown in Figure 1–36.

FIGURE 1–36. Snapshot Properties Page


Snapshot - Inputs - Outputs	
Target Flipped	<input type="checkbox"/>
Camera Height	13.500
Target Height	0.125
Physical CalDot Positions	<Click to Display Values>
IO for Enabling Displays	<Unassigned>

Settings

- **Target Flipped** — Indicates that the calibration target is flipped.
- **Camera Height** — Specifies the distance from the part surface to the face of the camera lens. Used by BGA only.
- **Target Height** — Specifies the thickness of the calibration target. Used by BGA only.
- **Physical CalDot Positions** — A list of points that holds the physical locations entered by you for each dot in the calibration target.
- **IO for Enabling Displays** — Allows you to define an input I/O point to control the display of the image and tool graphics. When the specified point is turned on, the displays will be updated normally; when it is off, no display will be available.

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.
- **Acquire Time Stamp** — The time that the acquisition started.
- **Snap Output Buffer** — Contains the next image to be processed by its component steps.

- Snap Status — Status as to what the condition that the frame was generated (Trigger Overrun, Process Overrun, Valid Frame, Camera Timeout etc.).
- IO State — The state of the field inputs and outputs at the time the snapshot was taken.
- Free Buffers — The number of buffers still available that can acquire snapshots.
- Calibration Results Data — The forward and inverse calibration matrices obtained from calibration processing.

I/O Summary

Snapshot provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Camera: aaa Gain: bbb Offset: ccc

Where: aaa = number of the camera
bbb = the gain in format bb.b
ccc = the offset, ± 255

Error Messages:

- Aborted
- BadFile
- DMAdone
- DMAtimeout
- Done
- Empty
- EmptyList
- Process Overrun
- Rpttimeout
- Timeout
- Trigger Overrun
- Waiting

Trajectory Step

This step combines the functionality of several steps to execute a group of child steps at a set of predefined locations. It also records a set of results from the execution at each location. You can insert any tool into the Trajectory step.

Other Steps Used

- Data Array — Container for trajectory locations, enable flags and result recording at each location.
- If — Allows for conditional execution of particular iterations of the loop. By default, it will connect to an enable flag in the Data Array.
- OnePt Locator — Moves a set of shapes based on positions specified in the Trajectory Step that are stored in the Data Array.
- With — Provides the context switching for each iteration of the loop.

Theory of Operation

The Trajectory Step is based on the Loop Step. It provides execution control of a group of child steps. It adds additional interfaces to simplify the specification of runtime locations and result recording.

FIGURE 1–37. Example of Trajectory Step Job



The runtime of the Trajectory Step is the same as the Loop Step, except that each iteration will execute the embedded With, If and Locator steps as well.

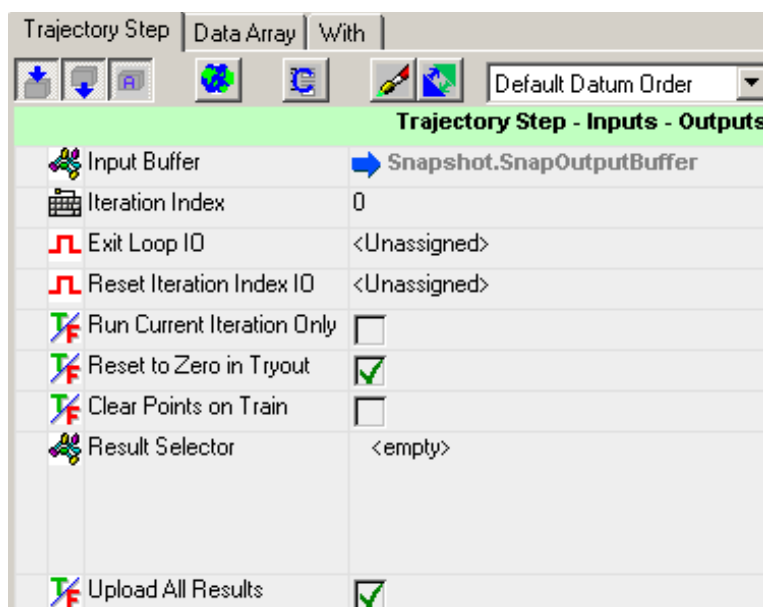
When the Trajectory Step runs, the Iteration Index is reset to 0. Then, for each iteration, Exit Loop I/O is evaluated to determine if the loop step should continue to execute. The child steps of the Trajectory Step are then run normally as they would if they were not inside of a Trajectory Step, including the embedded With, If and Locator steps. Then, Iteration Index is incremented and the next iteration runs. This continues until each location of the Trajectory Step is run. The status of the Trajectory Step is then set to true only if all steps of every iteration also pass.

When Run Current Iteration Only is set to true, Iteration Index is not reset to 0 or incremented. The child steps are only run once. This can be useful when analyzing the results from one specific location, by allowing a single location to be run each time.

Description

Trajectory Step allows editing through the Trajectory Step properties page, as shown in Figure 1–38.

FIGURE 1–38. Trajectory Step Properties Page



Settings

- **Iteration Index** — Specifies the active loop index. When the step is run, it is set to 0, then incremented for each time the set of child steps run. During setup mode, you can set this value.

Default: 0

- **Exit Loop IO** — Specifies an I/O number to be examined at the start of each iteration. If the value of the I/O is true, the remaining loop iterations are skipped and program execution continues with the step immediately following the loop step.

- **Reset Iteration Index IO** — When this Input turns ON, the Loop step's iteration index will be reset to 0. This provides you with a method for resetting the loop (if needed).
- **Run Current Iteration Only** — When enabled, the set of sub-steps will be run only once and the Iteration Index will not be reset or incremented.

Default: Disabled

- **Reset to Zero in Tryout** — If checked, then after running this step in Setup mode (does not apply to run mode), the iteration index will be reset to 0 and any child steps will be run again.
- **Clear Points on Train** — When enabled, the point list is cleared when the step is trained. This allows the point list to be regenerated by any inserted Trajectory Setup steps each time the step is trained. It should be disabled when specifying the trajectory locations manually or through VB, to keep the specified locations intact.

Default: False (True when a Trajectory Setup step is inserted)

- **Result Selector** — Contains a list of datums whose results are to be recorded from each iteration of the loop. Otherwise, the local copy of each step's results get overwritten from each run of the loop. When a datum is added to this list, a corresponding datum of the same type is added to each page of the FlexArrayDm.
- **Upload All Results** — When TRUE, each result added for recording by the Result Selector has every iteration tagged for upload. When disabled, individual results can be tagged for upload through the InspectionResultsDm of the Inspection Step.

Default: True

Training

Before training the Trajectory Step, the parameters of child Trajectory Grid Setup steps should be set. When the Trajectory Step is trained, all child Trajectory Grid Setup steps are first run to generate a list of trajectory locations.

Because the number of locations may change during training, **Iteration Index** is reset to 0, and the embedded locator step is run to move shapes to location 0.

After you select **Clear Points on Train**, the current location list of the Trajectory Step is erased, to be replaced by the newly generated location list. Otherwise, the locations generated by the set-up steps are appended to the current location list. The locations are retrieved and added to the Trajectory Steps location list.

Then, location 0 is activated, which may be different than before training, and the embedded locator step is trained at location 0. This establishes a baseline relationship between the trajectory location and shapes to be moved by the trajectory step.

Except when changing the list of trajectory locations with the Trajectory Grid Setup, the Trajectory Step does not need to be re-trained.

Results

- **Status** — This result is true when each iteration of sub-steps is run successfully.

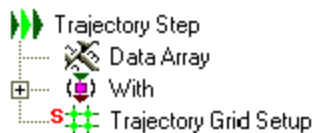
I/O Summary

None.

Trajectory Grid Setup Step

This step generates regular grids of points for a Trajectory Step. The Trajectory Step can contain any number of Trajectory Grid Setup steps or none of them. It can only be inserted into a Trajectory Step, as shown in Figure 1–39.

FIGURE 1–39. Trajectory Grid Setup Example



Other Steps Used

None.

Theory of Operation

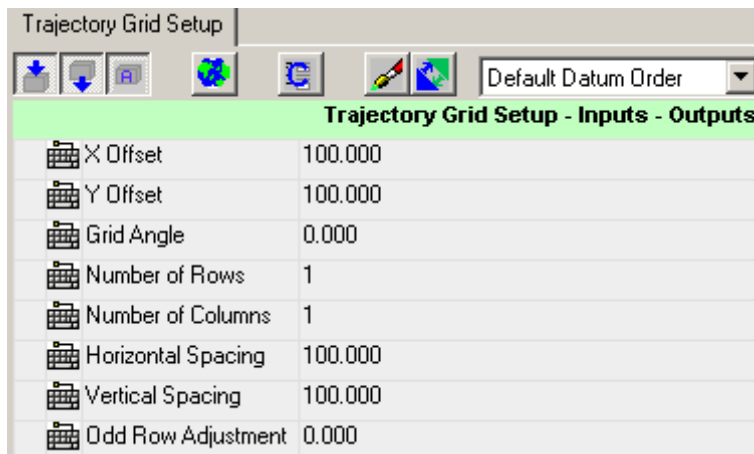
When the Trajectory Step is trained, the Trajectory Grid Setup steps will first create a list of points based on the grid specifications. The list of points is placed in the Point List output datum.

After all set-up steps have generated their output list of points, the Trajectory Step will collect the new point locations and add them to its list of trajectory locations. If the Trajectory Step's datum **Clear Points on Train** is enabled, the new set of points generated by the set-up steps will replace the current list of points in the Trajectory Step. Otherwise, the new set of points will be appended to the current list of trajectory locations.

Description

Trajectory Grid Setup Step allows editing through the Trajectory Grid Setup Step properties page, as shown in Figure 1–40.

FIGURE 1–40. Trajectory Grid Setup Step Properties Page



Settings

- X Offset** — The X component of the Pixel location of point 0.
 Default: 100
- Y Offset** — The Y component of the Pixel location of point 0.
 Default: 100
- Grid Angle** — Angle of the X axis of the grid in degrees. The center of rotation is point 0, as defined by X Offset and Y Offset.
 Default: 0°
 Range: -360° to 360°

- **Number of Rows** — Number of rows of points in the grid.
Default: 1
- **Number of Columns** — Number of columns of points in the grid.
Default: 1
- **Horizontal Spacing** — Pixel spacing between columns of points in pixels.
Default: 100
- **Vertical Spacing** — Pixel spacing between rows of points in pixels.
Default: 100
- **Odd Row Adjustment** — Additional horizontal pixel spacing added to Odd rows for generating interstitial grids. The first row of points is row 0 (even). The spacing is applied to the second row and every other row after that.
Default: 1

Training

The Trajectory Grid Setup will generate the set of trajectory points when the parent Trajectory Step is trained.

Results

- **Point List** — This PtListDm contains the set of trajectory points as described by the parameter settings. It is updated when the parent Trajectory step is trained.

I/O Summary

None.

VarAssign Step

This step allows you to generate a double datum with a value based on an input condition. This provides the capability of adding a datum where one is not already supplied by a step. When the VarAssign Step is run, it evaluates the condition and adjusts its output accordingly.

Other Steps Used

None.

Theory of Operation

The VarAssign Step provides a way to introduce a new parameter into a Job. When all parameters of the VarAssign Step are left at their default values, the step adds one to its output each time it is run. In this mode, it is a simple inspection counter. With a simple change, the step can count a specific event. When **Condition for Execution** is defined to be the occurrence of an event, the output datum increments by 1 each time the event occurs. To use the step more typically, you can also define **Value to Assign to Output**. The step evaluates that expression and defines the output datum accordingly each time the step is run and **Condition for Execution** evaluates to TRUE.

Description

VarAssign Step allows editing through the VarAssign Step properties page, as shown in Figure 1–41.

FIGURE 1-41. VarAssign Step Properties Page

VarAssign Step - Inputs - Outputs	
Condition for Execution	1
Value to Assign to Output	Out + 1
Initial Output Value	0.000
Upload Output With Results	<input type="checkbox"/>
Condition to Re-Initialize Output	0
Minimum Allowed Output Value	-100000000000.000
Maximum Allowed Output Value	100000000000.000
Error Condition Output	<Unassigned>

Settings

- **Condition for Execution** — Assigns the user-defined value to the output double datum. As a result, the value defined in **Value to Assign to Output** is evaluated and assigned each time the step is run. This property is an expression. As a result, you can choose to enter any condition here, such as when another tool fails or when another output is equal to some specified value.

Default: 1

- **Value to Assign to Output** — Defines the value that should be assigned to the output double datum of the VarAssign Step. This means that the output is incremented by one each time the step is run and the Condition for Execution evaluates to TRUE. This property is an expression. As a result, you can choose to enter any value here, such as the sum of two other datum values.

Default: Out + 1

- **Initial Output Value** — You can specify an initial value for the output double datum. This is the value assigned when the inspection in which the VarAssign Step is contained enters Run mode.

Default: 0

- **Condition to Re-Initialize Output** — When this expression is non-zero, the output is reset to the Initial Output Value.

Default: 0

- **Minimum Allowed Output Value** — If the Output Value is less than this value, the Error Condition Output datum is set to true.

Default: -100000000000.0

- **Maximum Allowed Output Value** — If the Output Value is greater than this value, the Error Condition Output datum is set to true.

Default: 100000000000.0

- **Error Condition Output** — Selects the I/O point to be set if the Output Value falls outside of the range set by the Minimum and Maximum Allowed Output Values.

Default: <none>

Training

None.

Results

The result of running the VarAssign Step is that the condition for execution is evaluated. When TRUE, the value of the output double datum matches the setting you made.

- **Output Value** — Contains the result of evaluating the expression.
- **Error Condition Occurred** — This output is true when the Output Value falls outside of the valid range set by the Minimum and Maximum Allowed Output Values.

I/O Summary

None.

Vision System Step

This step represents a given GigE Camera or smart camera installed on the host PC and encapsulates the features of that device. Some features, which you can configure, include camera selection and bufferpool configuration, I/O point configuration, and hardware acceleration.

Other Steps Used

None.

Theory of Operation

The Vision System step is the parent of the entire Job for one GigE Camera or smart camera.

A Vision System step is always created with an Inspection step, which represents an inspection task. You may add additional inspection steps. Multiple inspection steps are necessary when an application has to support multiple asynchronous inspections.

A Vision System step is created based on the hardware chosen when a new job was created, as shown in Figure 1–42.

FIGURE 1–42. Target Job Tree



The Vision System step creates a series of resources used during image acquisition and processing.

When there is no vision system inside the PC, one Vision System step is created and can be used without I/O, camera or hardware acceleration support by using a GigE Camera. The Job can be saved and later downloaded to the hardware where it can run with full functionality.

The Vision System step provides access to GigE Camera or smart camera resources such as the camera digitizer, ASIC, and I/O.

I/O consists of both physical I/O and virtual I/O. Virtual I/O provides the PC with a set of I/O points that behave much like physical I/O points, but can only be accessed by software. They have the advantage of being both inputs and outputs at the same time, and they can hold 32-bit values instead of a binary state. This enables software on the PC to communicate with the Job using a mechanism that is conceptually similar to using physical I/O but without requiring special hardware and wiring. By default, the system has 2048 virtual I/O points.

The AvpIOClient COM Object is capable of writing and reading all I/O points with any Vision System. Refer to Chapter 8 of the Visionscape Programmers Kit (VSKit) Manual for comprehensive information on using the AvpIOClient COM Object.

Multiple Resolution

Note: Multiple Resolution is only available using progressive scan cameras (Sentech A33 and Sentech A152).

In Visionscape V9.0.0, you can select up to two different camera types when using the GigE Camera. You select the Camera Definition for the cameras connected to your GigE Camera by selecting it for each channel in the **Camera Definitions And Buffer Counts** property of the VisionSystemStep.

You can program a maximum of two different types for the four camera channels available in any mix. For example, 1-3, 2-2, 3-1, or other combinations when less than four channels are used.

You allocate the number of buffers desired for one of the resolutions in the **Camera Definitions And Buffer Counts** property; the system shows how many buffers this means for the other resolutions on the other camera channels.

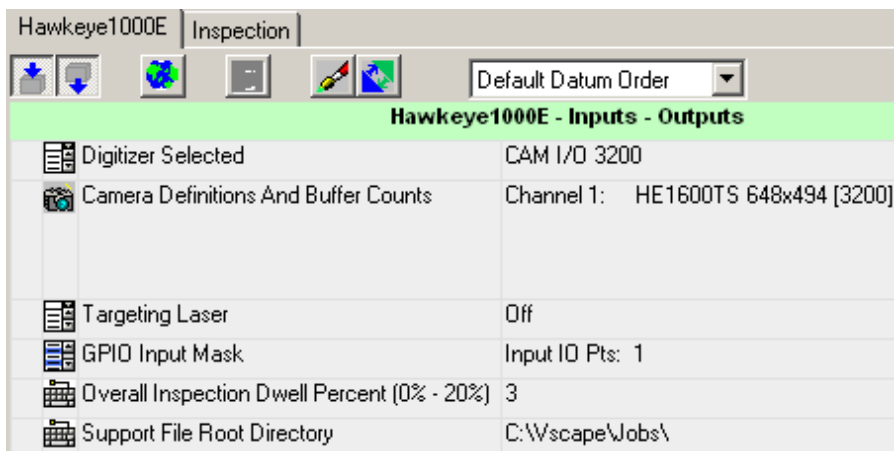
The memory used to store the captured images (the BufferPool) is common to all resolutions. In other words, you decide how many buffers the highest resolution camera are required and the system allocates enough memory for those.

This allocated memory is “shared” by both camera types; there is no separate Buffer pool for each camera resolution or camera channel. Therefore, the amount of buffers available at any given time when the system is running is a function of which channels have already acquired or are acquiring an image at that particular time.

Description

Vision System step allows editing through the Vision System Step properties page, as shown in Figure 1–43.

FIGURE 1–43. Vision System Step Properties Page



Settings

- **Digitizer Selected** — Lists the available digitizers for the vision system and “<none>”. When you select the digitizer, the set of cameras available for the system is automatically updated to those cameras specifically used with that digitizer. If you select “<none>”, you can select any camera, but each AcquireStep must be using image files as image sources.
- **Digitizer Mode** — Lists the various master/slave modes for the digitizer. When in Master/Slave mode, the digitizer is coupled to other devices for trigger control. One device can be the Master and all other coupled devices are slaves.

- Camera Definitions And Buffer Counts** — Lists all the available cameras for this system based on the selected digitizer. Also lists the number of buffers in the bufferpool for each camera selection. The bufferpool is a pool of buffers that are used by image acquisition. As each acquisition is executed, a buffer is used from this pool. When no other object requires the image anymore, it can be returned to the pool. The size of the pool is determined by this value multiplied by the dimensions of the selected camera. Some digitizers (740) allow different camera types to be selected. In this case, the bufferpool memory is mapped according to different camera dimensions. Each count reflects the number of buffers in the pool for the specific camera type. All camera types use the same bufferpool. The number of buffers is always constrained by the amount of available memory on the 0th bank of the ASIC, or by the amount available in DMA memory. Typically, you can create the bufferpool to be, at most, 80% of the available memory. The total amount of memory to be used in the bufferpool is displayed in MB at the bottom. If you change the bufferpool count to be larger than what is available, the numbers turn red and will be altered when the system prepares to run. The camera definition entries are read from the camdef files installed on the system under the Visionscape software root (e.g., C:\Vscape\Drivers\CamDefs).
- Targeting Laser** — Enables or disables (default) the targeting laser.
- GPIO Input Mask** — Displays the current input selection mask for the general purpose I/O. Any entry in the list that is selected is programmed on the digitizer as an input point. All others are programmed as output points. By default:

TABLE 1-17. GPIO Points

GPIO Points	Input/Output
1 - 8	Input (Selected)
7 - 16	Output (Not Selected)

- Overall Inspection Dwell Percent (0% - 20%)** — (Default is 3%) The Overall Inspection Dwell Percent (OIDP) property allows the user to set the amount of CPU cycles that should be reserved for Reports and the User Interface. A setting of 0 (zero) disables the OIDP feature and should only be used when maximum throughput is required and when using cameras. Be aware, however, that on slower and less performing PCs, the UI may become unresponsive when the this property is set to 0. The maximum setting permitted is 20%.

The OIDP applies to all inspections, and operates regardless of the target device or platform. It operates by suspending all inspections on a recurring basis so that the specified percentage of dwell time is available for other tasks (user interface, reports, etc.). The duration that the inspections are suspended at any given time is the minimum permitted by the target operating system – approximately 1.7 ms for Windows and 1 ms for vxworks. The Dwell percentage correlates to the amount of CPU free usage as shown by the Windows Task Manager when Visionscape is pretty much the only application running in the system.

Note: Consideration must be given to the type of vision job being performed to determine the best value for the OIDP. For example, a value of 5% would translate to an occasional jump in process time of ~1.7 ms for short duration (<10ms) inspections, but for lengthy inspections, the time would be more significant. As an example for an inspection whose process time is 1000ms, an OIDP setting of 5% will result in a process time increase of 50ms.

The older Inspection Dwell Time Setting (In the Inspection Step) has been removed. There is no automatic translation from the old scheme to the new, so if an older job is imported the dwell is replaced by the default 3% setting.

- **Support File Root Directory** — Points to the support file root for companion files (fonts, Perl scripts). For new Jobs, by default, it points to the `\Vscape\Jobs` directory and is set up by the Visionscape Installer.

Training

None.

Results

- **Status** — Set to true after successful execution of the step.
- **Device1** — Vision descriptor that represents a software target when no other devices are present in the system. If devices are present, the Vision descriptor uses the device supplied to the system.

I/O Summary

None.

Wait Step

This step provides a configurable delay that can wait during a Job execution. For example, when Digital I/O switches on lights, it may be necessary to wait a few milliseconds for the lights to reach full intensity.

Other Steps Used

None.

Theory of Operation

The WaitStep provides a configurable optional delay so that a Job can pause for a defined amount of time. This is often useful when using handshake communications with external equipment or when waiting for such external equipment to become active.

Description

The WaitStep allows editing through the WaitStep properties page, as shown in Figure 1–44.

FIGURE 1–44. WaitStep Properties Page



Settings

Wait time (ms) — A configurable delay time in milliseconds that causes the Job to delay when this step is run.

Default: 0

Training

None.

Results

- **Elapsed Time** — Contains the actual number of milliseconds the step paused.

I/O Summary

Inputs: Wait time: x msecs

Where: x is the user-specified number of milliseconds

Outputs: Last wait period: x msecs

Where: x is the number of milliseconds paused

With Step

This step functions as the switch to activate a single page within a FlexArrayDm. It also performs the coordinate system fixes for any output datums in the FlexArrayDm.

Other Steps Used

None.

Theory of Operation

Page Index is retrieved by following the input connection. This input usually connects to the Loop Step's **Iteration Index**. The context of the connected Flex Array is then switched to the corresponding Page.

Child steps of the With Step execute normally. Then, all Output datums within the current active page of the connected Flex Array are modified to correct for coordinate system changes.

Description

With Step allows editing through the With Step properties page, as shown in Figure 1–45.

FIGURE 1–45. With Step Properties Page



Settings

- **FlexArray** — Specifies the datum container that the With Step will operate on. This input is usually connected to a FlexArrayDm that is contained by a FlexArray Step.
- **Page Index** — Specifies the active datum page of the associated input Array. This input is usually connected to the Iteration Index of a Loop or Trajectory Step.

Training

None.

Results

- **Status** — This output is always true for the With Step.

I/O Summary

None.

Color Channel Selection for Vision Tool Steps

When working with color images, the vision tools run on synthesized images derived from the color image. These “channels”, defined as arithmetic manipulation of the RGB color planes, can be created from the camera’s RGB format to aid in processing and/or analyze the color data. These channels will be selectable as an input datum from the vision tools themselves. The following table describes the available channels and what the expected image results would be when synthesized from the R, G, B data coming from the camera or from a .TIF image file.

Note: Channel synthesis is done on demand based on the tool. The channel pixel data will be cached for buffers such that a different tool with the same channel would process the pixel data cached by the first tool. In addition to channels (buffers generated using the functions defined in the table below) the Visionscape Gain and Offset Step can also be used to enhance contrast of each channel digitally. The Gain and Offset Step has a mechanism to select to which channel the gain and offset will be applied.

TABLE 1-18. Color Channel Synthesis

Channel	Synthesis	Implementation
R	R Planar Pixel Values directly from Camera, File or through Bayer Extraction	Camera or CPU
G	G Planar Pixel Values directly from Camera, File or through Bayer Extraction	Camera or CPU
B	B Planar Pixel Values directly from Camera, File or through Bayer Extraction	Camera or CPU
R-G (Saturated)	$\forall r \in R; \forall g \in G;$ $(R - G)_{r,g} = \begin{cases} 0 & r - g \leq 0 \\ r - g & r - g > 0 \end{cases}$	CPU

TABLE 1-18. Color Channel Synthesis

Channel	Synthesis	Implementation
R-B (Saturated)	$\forall r \in R; \forall b \in B;$ $(R-B)_{r,b} = \begin{cases} 0 & r-b \leq 0 \\ r-b & r-b > 0 \end{cases}$	CPU
G-R (Saturated)	$\forall r \in R; \forall g \in G;$ $(G-R)_{r,g} = \begin{cases} 0 & g-r \leq 0 \\ g-r & g-r > 0 \end{cases}$	CPU
B-R (Saturated)	$\forall r \in R; \forall b \in B;$ $(B-R)_{r,b} = \begin{cases} 0 & b-r \leq 0 \\ b-r & b-r > 0 \end{cases}$	CPU
G-B (Saturated)	$\forall b \in B; \forall g \in G;$ $(G-B)_{g,b} = \begin{cases} 0 & g-b \leq 0 \\ g-b & g-b > 0 \end{cases}$	CPU
B-G (Saturated)	$\forall b \in B; \forall g \in G;$ $(B-G)_{r,g} = \begin{cases} 0 & b-g \leq 0 \\ b-g & b-g > 0 \end{cases}$	CPU

TABLE 1–18. Color Channel Synthesis

Channel	Synthesis	Implementation
R-G (Scaled)	$\forall r \in R; \forall g \in G;$ $(R-G)_{r,g} = (r - g + (2^8 - 1)) / 2$	CPU
R-B (Scaled)	$\forall r \in R; \forall b \in B;$ $(R-B)_{r,b} = (r - b + (2^8 - 1)) / 2$	CPU
G-R (Scaled)	$\forall r \in R; \forall g \in G;$ $(G-R)_{r,g} = (g - r + (2^8 - 1)) / 2$	CPU
B-R (Scaled)	$\forall r \in R; \forall b \in B;$ $(B-R)_{r,b} = (b - r + (2^8 - 1)) / 2$	CPU
G-B (Scaled)	$\forall b \in B; \forall g \in G;$ $(G-B)_{b,g} = (g - b + (2^8 - 1)) / 2$	CPU
B-G (Scaled)	$\forall b \in B; \forall g \in G;$ $(B-G)_{b,g} = (b - g + (2^8 - 1)) / 2$	CPU
Difference(R,G)	$ R - G $	CPU

TABLE 1–18. Color Channel Synthesis

Channel	Synthesis	Implementation
Difference(R,B)	$ R - B $	CPU
Difference(G,B)	$ G - B $	CPU
Intensity(R,G,B)	$\forall r \in R; \forall g \in G; \forall b \in B: \forall i \in I$ $i_{r,g,b} = \frac{r + g + b}{3}$	CPU
Saturation(R,G,B)	$\forall r \in R; \forall g \in G; \forall b \in B: \forall s \in S$ $s_{r,g,b} = \left(1 - \frac{3 \times \min(r, g, b)}{(r + g + b)} \right) \times (2^8 - 1)$	CPU

TABLE 1–18. Color Channel Synthesis

Channel	Synthesis	Implementation
Hue(R,G,B)	$\forall r \in R, \forall g \in G, \forall b \in B: \forall h \in h$ $h_{r,g,b} = \cos^{-1} \left[\frac{\frac{1}{2}[(r-g) + (r-b)]}{\sqrt{(r-g)^2 + (r-b)(g-b)}} \right]$ $\text{if } b > g \quad h_{r,g,b} = 2\pi - h_{r,g,b}$ $h_{r,g,b} = \frac{h_{r,g,b}}{2\pi} \times (2^8 - 1)$ <p>Note that the starting point (0 degrees) can be assigned to either the red, green or blue plane. The output values of the equation are then added to this offset.</p>	CPU
Normalized(R)	$\forall r \in R: G(r) = g; B(r) = b$ $r_{Normalized} = \frac{r}{r + g + b} \times (2^8 - 1)$	CPU
Normalized(G)	$\forall g \in G: R(g) = r; B(g) = b$ $g_{Normalized} = \frac{g}{r + g + b} \times (2^8 - 1)$	CPU

TABLE 1–18. Color Channel Synthesis

Channel	Synthesis	Implementation
Normalized(B)	$\forall b \in B : R(b) = r; G(b) = g$ $b_{Normalized} = \frac{b}{r + g + b} \times (2^8 - 1)$	CPU
Equalized(R)	<p>Compute Histogram for R plane (h(x) denotes bin height for given bin)</p> <ul style="list-style-type: none"> Find min and max values of histogram. Remap based on the following: $\forall r \in R$ $T(x) = \sum_{r_{min}}^x h(x) - \frac{1}{2} [h(r_{min}) + h(r_{max})]$ $r_{equalized} = \frac{(2^8 - 1)^2}{T(r_{max})} T(r)$	CPU
Equalized(G)	<p>Compute Histogram for G plane (h(x) denotes bin height for given bin)</p> <ul style="list-style-type: none"> Find min and max values of histogram. Remap based on the following: $\forall g \in G$ $T(x) = \sum_{g_{min}}^x h(x) - \frac{1}{2} [h(g_{min}) + h(g_{max})]$ $g_{equalized} = \frac{(2^8 - 1)^2}{T(g_{max})} T(g)$	CPU

TABLE 1–18. Color Channel Synthesis

Channel	Synthesis	Implementation
Equalized(B)	<p>Compute Histogram for B plane ($h(x)$ denotes bin height for given bin)</p> <ul style="list-style-type: none"> • Find min and max values of histogram. • Remap based on the following: $\forall b \in B$ $T(x) = \sum_{b_{\min}}^x h(x) - \frac{1}{2} [h(b_{\min}) + h(b_{\max})]$ $b_{\text{equalized}} = \frac{(2^8 - 1)^2}{T(b_{\max})} T(b)$	CPU

Color Channel Selection Examples

Color Channel datums are available when a color image is present and are not available when a monochrome image is present.

FIGURE 1-46. Blob Tool with Intensity Color Channel Selected

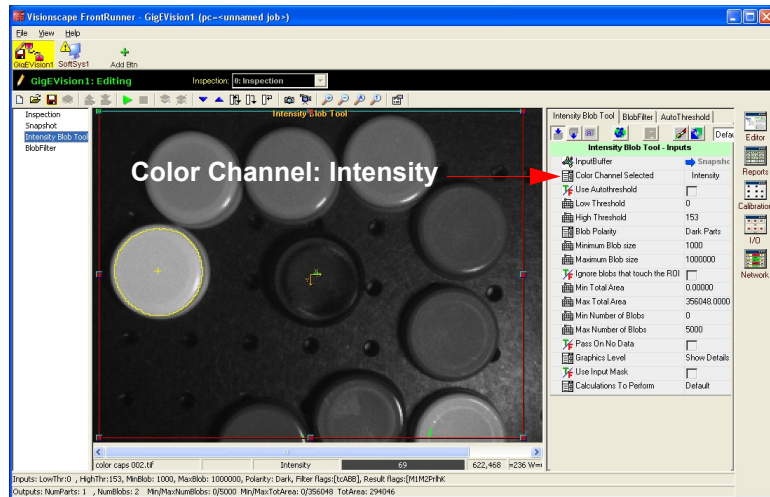


FIGURE 1-47. Blob Tool with B-R Saturated Color Channel Selected

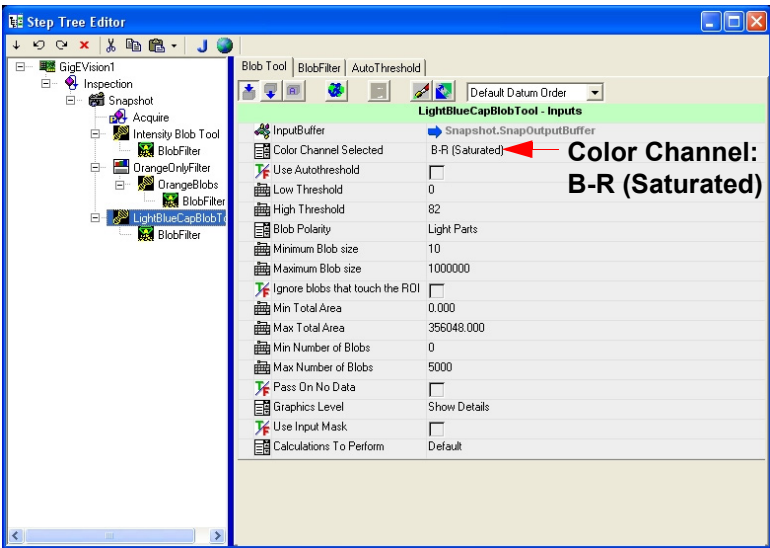
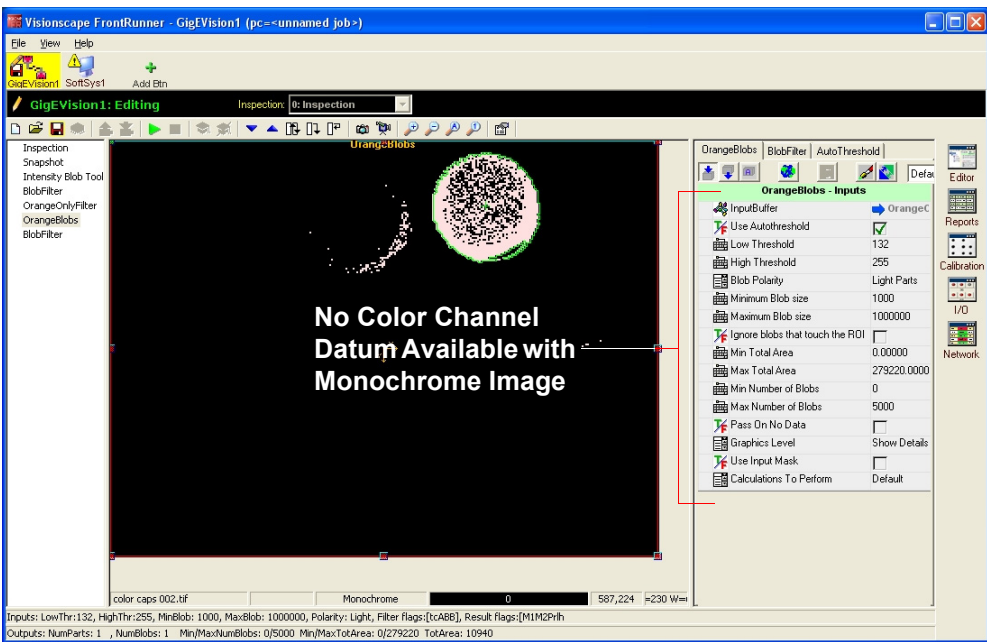


FIGURE 1-48. Blob Tool with Monochrome Image and No Color Channel Selected

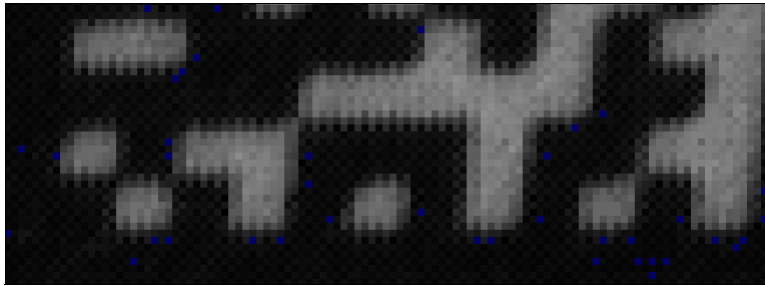


Green Interpolation Color Channel

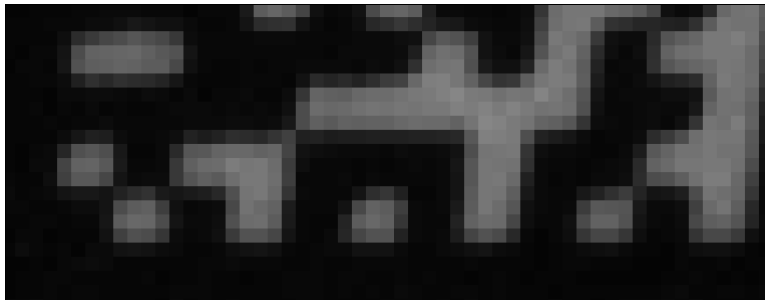
This channel uses only the green pixels to generate a monochromatic buffer for processing. The resulting output of this channel is similar to the luminance channel, but evaluated at every pixel in in the x,y domain. It looks very similar to a smoothed luminance channel. The processing is efficient and local processing is not subject to changes in white balance (white-balancing the unit can have a minor impact on overall DC level).

Color Channel Comparison

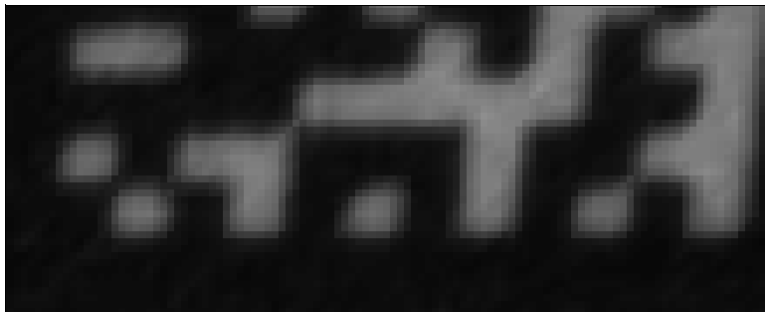
RB Interpolate



Luminance



Green Interpolation



Color Channels

Index	Color Channel Description
0	Luminance
1	Red Plane (R)
2	Green Plane (G)
3	Blue Plane (B)
4	Hue: 0 = Pure Red
5	Hue: 0 = Pure Green
6	Hue: 0 = Pure Blue
7	Saturation
8	Intensity
9	Lightness
10	Maximum (R,G,B)
11	Minimum (R,G,B)
12	Green Interpolation
13	RB Interpolate
14	Normalized R
15	Normalized G
16	Normalized B
17	Equalized R
18	Equalized G
19	Equalized B
20	R-G (Saturated)
21	R-B (Saturated)
22	G-R (Saturated)
23	G-B (Saturated)
24	B-R (Saturated)
25	B-G (Saturated)
26	R-G (Scaled)
27	R-B (Scaled)
28	G-R (Scaled)
29	G-B (Scaled)
30	B-R (Scaled)
31	B-G (Scaled)
32	R-G
33	R-B
34	G-B
35	RAW Data

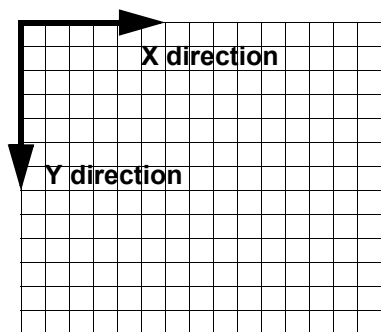
CHAPTER 2

Calibration

In machine vision, calibration is the process of mapping the pixel coordinate system of the camera sensor to a “world” coordinate system. This mapping defines the relationship between a distance measured in pixels in the camera versus the actual distance in inches or millimeters of the object being imaged.

The camera sensor is divided into discrete pixels. The image recorded in the vision system memory is an array recording the amount of charge generated at each sensor pixel location. The convention in machine vision and electronic imaging is to treat the top left corner of the image as the origin and to make all references to the pixel locations in a coordinate system in which X runs across the rows and Y runs down the columns of the sensor, as shown in Figure 2–1.

FIGURE 2–1. Sensor Coordinate System

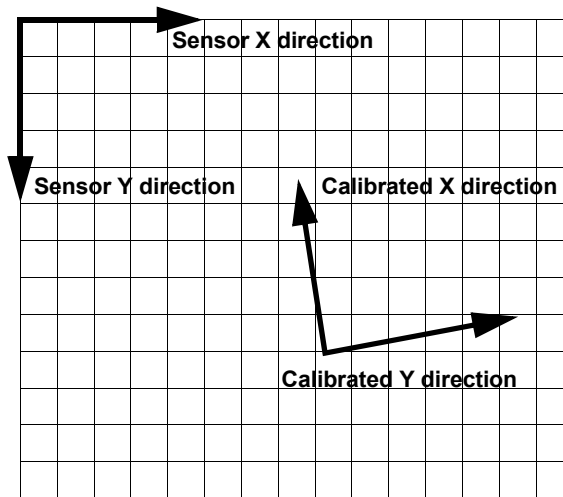


All features located in the image such as points, lines, edges, and distances are initially reported in this camera sensor frame of reference and are scaled to the scale prevailing in this frame. This means that, in the absence of a calibration, all locations are in pixel coordinates with respect to the top left of the image and all distances are in pixels. The process of calibration allows you to specify another more useful system of coordinates and scale. It also causes the results obtained from machine vision programs to be presented in this new system.

The calibration process supported by Visionscape allows you to specify the important features of the new Calibrated coordinate system. These are:

- The Location of the origin of the Calibrated Coordinate System — This is the position of (0,0) in the Calibrated Coordinate System relative to Camera Sensor (0,0).
- The Direction of the Major Axes of the Calibrated Coordinate System — This is the direction of plus X and plus Y in the Calibrated Coordinate System.
- The Angle between the Calibrated Coordinate System and the Camera Sensor Coordinate System — This is the amount of rotation between the Calibrated Coordinate System and the Camera Sensor.
- The Scale of the Calibrated Coordinate System — This is the relationship between the units of distance in the Calibrated Coordinate System and the pixel spacing in the camera sensor.

FIGURE 2–2. Relationship, Sensor Coordinate and Calibrated Coordinate Sys.

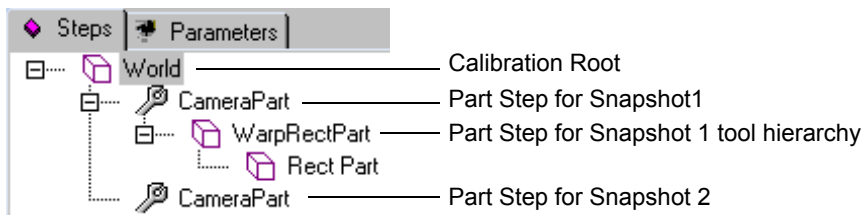


In addition to the moving, stretching, and folding of coordinate systems, Visionscape calibration corrects for one form of optical distortion called linear perspective distortion. This type of distortion occurs when the optical axis is not perpendicular to the object being imaged. The calibration system does not compensate for spherical distortions and aberrations introduced by the lens.

Calibration Tree Hierarchy

There is a PartStep tree structure not visible to you, which describes the relationship of all the system of coordinates defined by the Job. The root step of the part hierarchy is the World Part, which defines the permanent coordinate system of all cameras in all Vision systems. As Snapshots are created in various targets on the system, Part steps are created to define the calibrated coordinate space of that camera (World space). Steps inserted within the Snapshots that define coordinate spaces (warp, arith, etc.), create Part steps in the Calibration Tree to define the transformation needed for its particular output buffer coordinate system, as shown in Figure 2–3.

FIGURE 2–3. Calibration Tree — Example



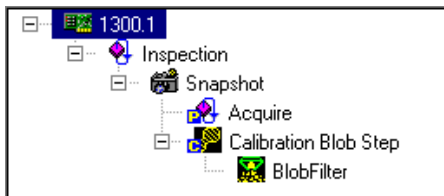
These Part steps do not require specific training, but the Part step inside the Snapshot does, to relate the camera pixels to world coordinates. This procedure is known as Camera Calibration. Once this Part tree is fully defined, results from one view can be combined with results from another view by transforming all results into the common world coordinate system.

Calibration Procedure

The process of calibration is based on matching specified real world positions of the features on a calibration target with their measured locations on the camera sensor. Forwards and backwards transformation matrices that relate world to pixel and pixel to world are created and stored in the PartStep of the Snapshot.

When you calibrate a camera with the Calibration Manager, a special calibration tool is created as a child of the Snapshot, as shown in Figure 2–4.

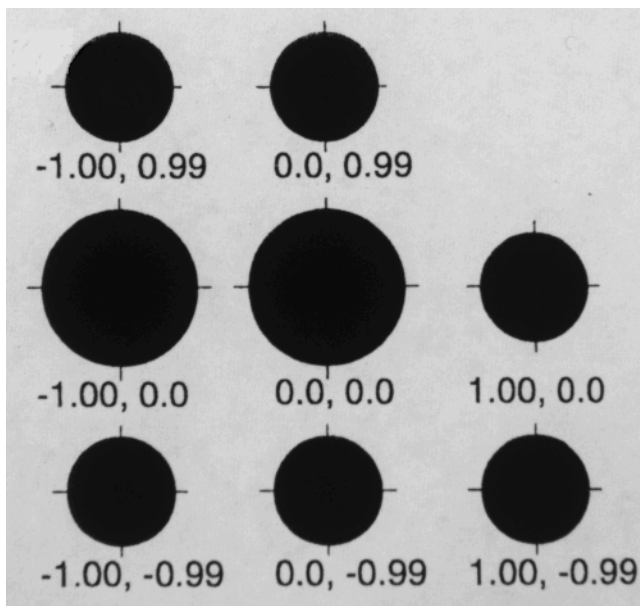
FIGURE 2–4. Example



This simple inspection contains Blob, which locates the calibration dots from which the calibration matrix is determined.

In the Visionscape implementation, the calibration target contains 8 dots arranged in the manner shown in Figure 2–5.

FIGURE 2-5. 8 Dot Calibration Target



To begin, obtain a Calibration Target with a dot pattern equivalent to Figure 2-5, and place it in the FOV of the camera to be calibrated.

The accuracy of the calibration will depend on the accuracy with which the relative locations of the dots on the calibration target are specified to the system. The calibration targets available from Omron Microscan are traceable to NIST and are, typically, certified to an accuracy of 0.00002 inches or 0.00005 inches, depending on target size.

A calibration target should be chosen to fill approximately 80% of the FOV. Table 2-1 shows the part numbers and FOV range of the more common target sizes available from Omron Microscan. Please contact Omron Microscan for other sizes and styles.

TABLE 2-1. Common Calibration Target Sizes

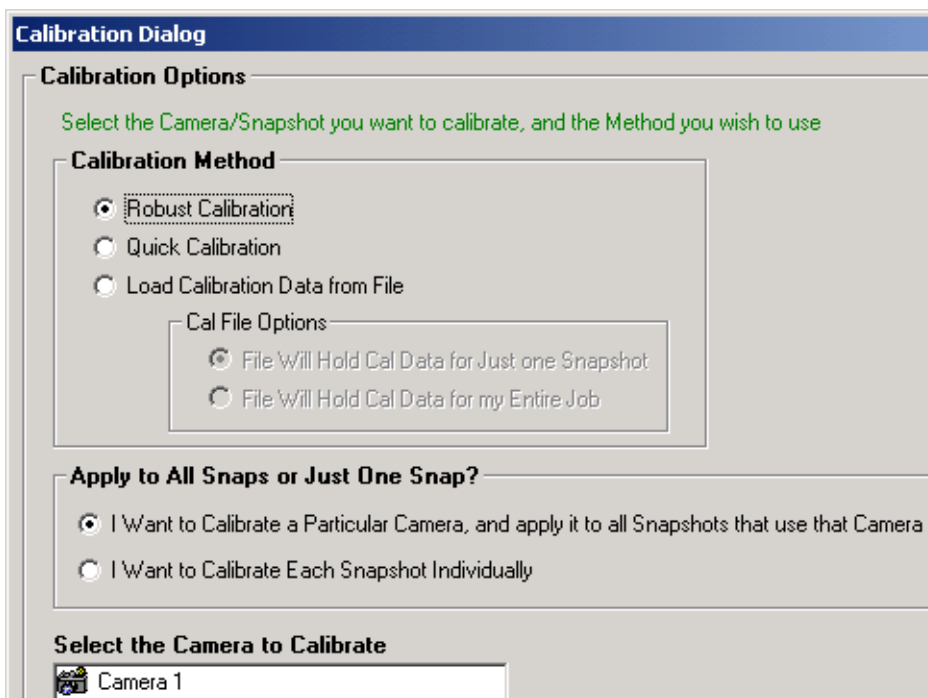
Part Number	Glass Size (mm) Square	Minimum FOV Width (in.)	Target Accuracy (in.)
001-904091	25.4	0.0736	0.00002
001-904092	25.4	0.1102	0.00002
001-904093	25.4	0.1654	0.00002
001-904094	25.4	0.2496	0.00002
001-904095	25.4	0.3673	0.00002
001-904096	25.4	0.5508	0.00005
001-904097	25.4	0.8079	0.00005
001-904098	69.85	1.8626	0.00005
001-904099	114.30	2.4850	0.00005
001-904100	139.7	3.7287	0.00005
001-904101	203.2	6.2173	0.00005

Using Robust Calibration

Use the following procedure to robustly calibrate a camera:

1. Start FrontRunner™ by selecting **Start > Visionscape > Visionscape FrontRunner**.
2. Add and select a camera (for more information, see Chapter 2 of the Visionscape FrontRunner User Manual).
3. Take control of the camera (for more information, see Chapter 2 of the Visionscape FrontRunner User Manual).
4. Create your Job.
5. Place the calibration target under the camera.
6. Click **Live Video**. Adjust the exposure and focus. Click **Live Video** again.
7. Click **Calibration**.

FrontRunner™ displays the Calibration dialog box, as shown in Figure 2-6.

FIGURE 2–6. Calibration Dialog Box

8. Under “Calibration Method,” select **Robust Calibration**.
9. Under “Apply to All Snaps or Just One Snap,” select either of the following:
 - I Want to Calibrate a Particular Camera, and apply it to all Snapshots that use that Camera
 - I Want to Calibrate Each Snapshot Individually
10. Under “Select the Camera to Calibrate,” highlight (to select) a camera.
11. Click **Next**.

FrontRunner™ displays the Calibration screen shown in Figure 2–7.

FIGURE 2-7. Enter Calibration Target Dot Locations

The screenshot shows a software window titled "Calibration Dialog" with a sub-header "Robust Calibration: Enter Cal Target Dot Locations". The window contains a grid of colored circles (blue and purple) representing calibration target dots. Each dot is associated with a small table for its X and Y coordinates. Below the grid, there are input fields for "Calibration Target Height" (0.125) and "Camera Height" (13.5). A checkbox labeled "Calibration Target is Viewed from Behind a Mirror" is unchecked. A green instruction at the bottom reads: "Enter the initial calibration parameters for the test target. When complete place the test target under the camera and click Next."

Dot Color	X	Y
Blue	-0.98400	-0.66250
Blue	0.00000	-0.66250
Purple	-0.98400	0.00000
Purple	0.00000	0.00000
Blue	0.68920	0.00000
Blue	-0.98400	0.66260
Blue	0.00000	0.66260
Blue	0.68920	0.66260

Calibration Target Height: 0.125
Camera Height: 13.5
Calibration Target is Viewed from Behind a Mirror: ☐

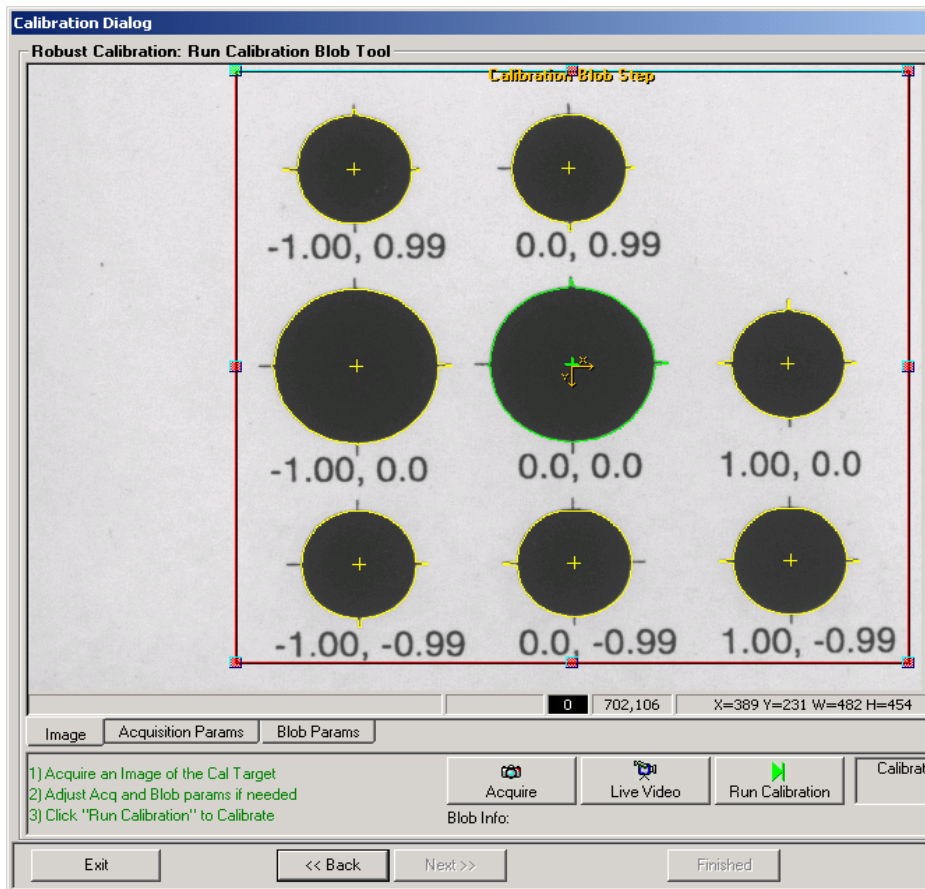
Enter the initial calibration parameters for the test target. When complete place the test target under the camera and click Next.

12. Enter the Calibration Dot locations.

13. Click Next.

FrontRunner™ displays a screen similar to the one shown in Figure 2-8.

FIGURE 2–8. Run Calibration Blob Tool

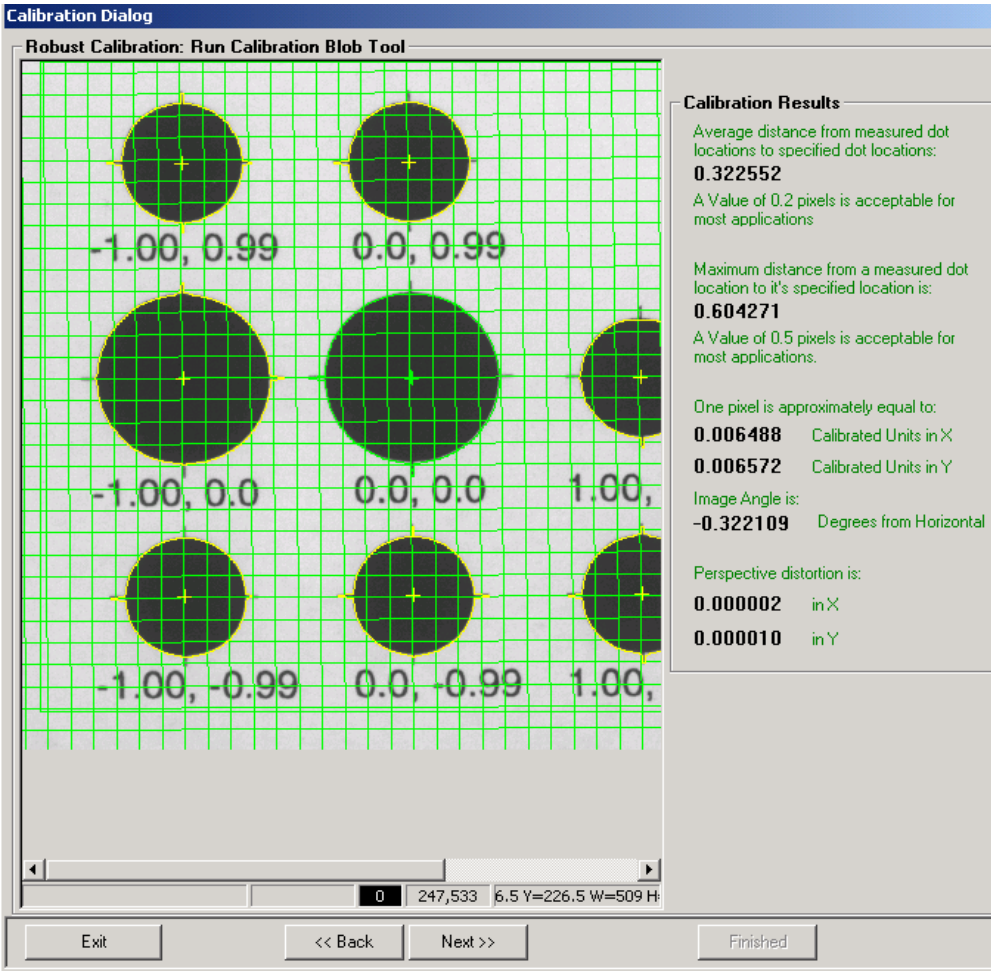


Note: You may need to enlarge the window.

14. Adjust the Calibration Blob ROI.
15. Click Acquire.
16. Click Run Calibration.

FrontRunner™ displays a screen similar to the one in Figure 2–9.

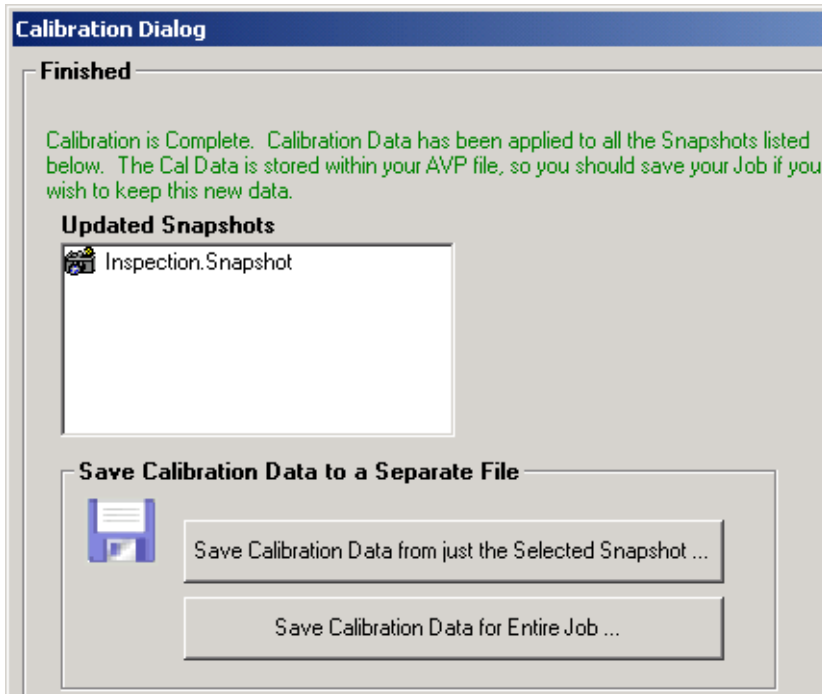
FIGURE 2-9. Calibration Results



17. Click Next.

FrontRunner™ displays the Finished screen, as shown in Figure 2-10.

FIGURE 2–10. Calibration Finished



18. Select one of the Save options and click Finish:

- **Save Calibration Data from just the Selected Snapshot** — This button allows you to save the calibration data for just the Snapshot that you just calibrated. Then, this data can be reloaded into any other Snapshot step in this Job or any other Job by using the **Load Calibration Data From File** option on the main Calibration page.

If you select this save option, FrontRunner™ displays the **Save Calibration from Current Snapshot** dialog box. Type in a file name. Click **Save**, and then click **Finished**.

- **Save Calibration Data for Entire Job** — This button saves the calibration data from every Snapshot in your Job to a single file. Then, this data can be reloaded into this Job or some other Job using the **Load Calibration Data From File** option on the main Calibration page, and then selecting the **File Will hold Cal Data for my Entire Job**.

Note: You can only reload this data into an Job that has the same number of Snapshots.

- If you select this save option, FrontRunner™ displays the **Save Calibration for Entire Job** dialog box. Type in a file name. Click **Save**, and then click **Finished**.

19. Save your Job.

Using Previously Saved Calibration Data

When calibration is performed, FrontRunner™ stores the data in the Snapshot step(s) of the Job. This means you must save a Job after calibration is completed.

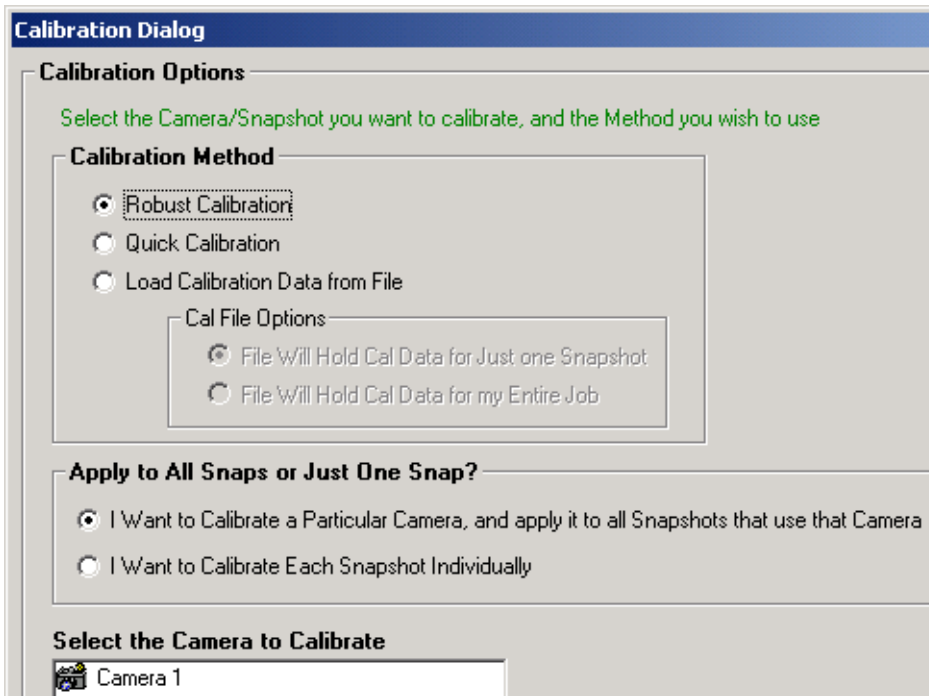
You do not need to use the Save options on this page unless you want to load the Calibration Data into another Job, or unless you want to simply have a backup.

Use the following procedure to load calibration data that was previously saved:

1. Start FrontRunner™ by selecting **Start > Visionscape > Visionscape FrontRunner**.
2. Add and select a camera (for more information, see Chapter 2 of the Visionscape FrontRunner User Manual).
3. Take control of the camera (for more information, see Chapter 2 of the Visionscape FrontRunner User Manual).
4. Create a new Job or open an existing Job.
5. Click **Calibration**.

FrontRunner™ displays the Calibration dialog box, as shown in Figure 2–11.

FIGURE 2–11. Calibration Dialog Box



6. Under “Calibration Method,” select Load Calibration Data from File.

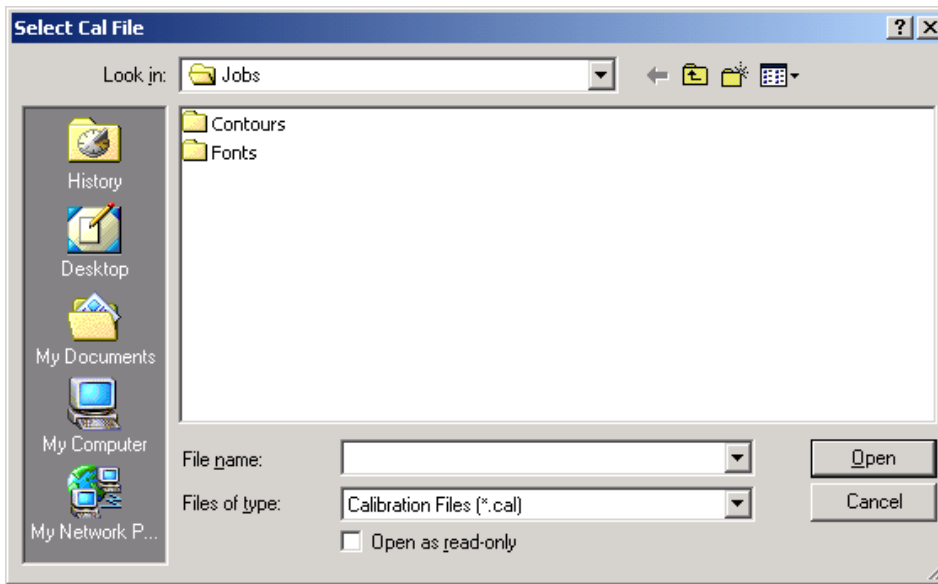
Note: After you select Load Calibration Data from File, the Cal File Options are no longer grayed out.

7. Select one of the Cal File options:
 - File Will Hold Cal Data for Just one Snapshot
 - File Will Hold Cal Data for my Entire Job
8. Under “Apply to All Snaps or Just One Snap,” select either of the following:
 - I Want to Calibrate a Particular Camera, and apply it to all Snapshots that use that Camera
 - I Want to Calibrate Each Snapshot Individually

9. Under “Select the Camera to Calibrate,” highlight (to select) a camera.
10. Click Next.

FrontRunner™ displays the Select Cal File dialog box, as shown in Figure 2–12.

FIGURE 2–12. Select Cal File Dialog Box



11. Highlight (to select) a calibration file (.cal) and click Open.

FrontRunner™ displays the Calibration Finished screen.

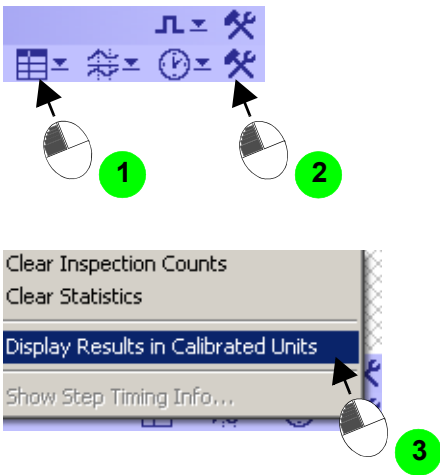
12. Click Finished.
13. Save your Job.

Your Job now contains the calibration data your saved previously.

Using Calibration in the Result Report

After you’ve calibrated your inspection, and while your Job is running, you can display results in calibrated units, as shown in Figure 2–13.

FIGURE 2–13. Displaying Statistics and Results in Calibrated Units



Results in calibrated units are shown in Figure 2–14.

FIGURE 2–14. Results in Calibrated Units

BlobFilter Tool.Center Point		1.603	1.258	-.975	1.000
BlobFilter Tool.Left Point		1.543	1.238	.000	1.000
BlobFilter Tool.Area		.006			
BlobFilter Tool.Unrotated Width		.125			

When calibrated units are chosen, the measurements displayed are in the units that specify the locations of the dots on the calibration target when the camera was calibrated. Measurements will be in the same coordinate system used to specify the locations of the dots on the calibration target.

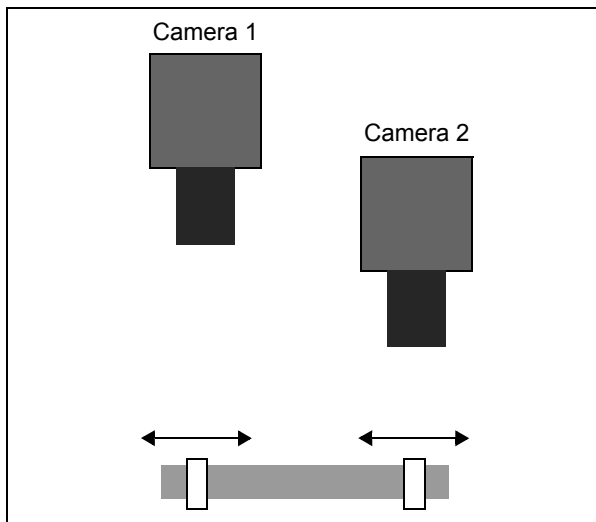
To display calibrated results for one of the measurement tolerance steps, check the Use Calibration checkbox for that step.

Multiple Camera Calibration

To perform measurements between two or more cameras requires a special technique. This technique is useful when a desired measurement requires more than one camera to acquire the necessary vision data, because the FOV for one camera would not provide sufficient resolution to perform the measurement.

Multiple camera calibration can best be explained by the example in Figure 2–15.

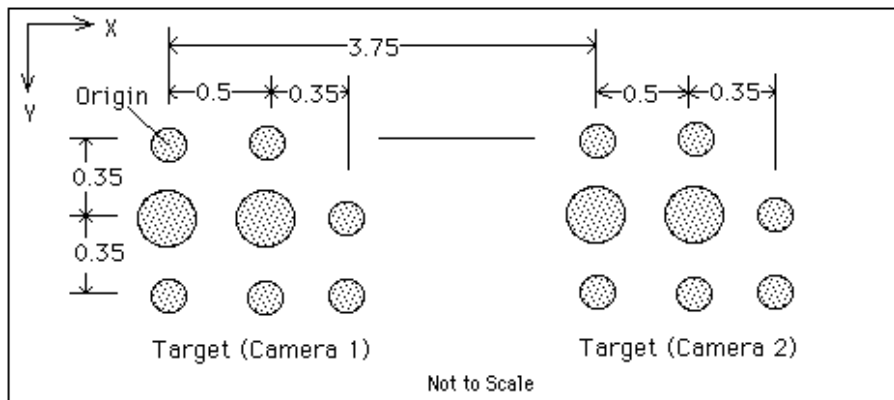
FIGURE 2–15. Two Camera Measurement Application



Two cameras are to measure the distance between two collars on a shaft. A calibration target with two separate eight (8) calibration dots patterns for each camera is required to perform this calibration. High accuracy measurements must be made to define the relationship between all sixteen dots on the double barrelled calibration target. Refer to “Calibration Target Design Criteria” on page 2-19 for more information.

An example of a fabricated calibration target for two cameras is shown in Figure 2–16.

FIGURE 2-16. Fabricated Calibration Target-2 Cameras — Example



The origin (0.0, 0.0) of the coordinate system has been located at the centroid of the upper left hand dot of calibration target 1. All other dot centroids have been measured with respect to this point. Any of the other dots could have been used as the origin, or even a point not on the calibration fixture, as long as the relationship of the calibration dots are known and fixed within the coordinate system.

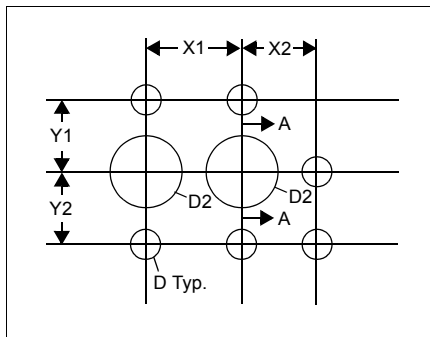
The calibrated results obtained for measurements between and within the cameras' fields of view (FOV) will be in the common coordinate system of the calibration targets.

Calibration Target Design Criteria

Visionscape expects a calibration target with the following criteria:

- The calibration target must have eight round dots. The center of these dots must be known in the coordinate system in which the camera is to be calibrated, i.e., inches, millimeters, etc.
- The calibration target must be designed and constructed with the dots oriented as shown in Figure 2–17, or its mirrored equivalent.

FIGURE 2–17. Typical Calibration Target Layout



The two dots in the positions shown must be at least 35% larger in area (17% larger in diameter) than the other dots. All other dots should be of the same size. The dots should be round and clearly distinguishable from the background. We recommend either white dots on a flat black background or black dots on a white background.

- The calibration target should be designed to nearly fill the FOV. This results in a more accurate calibration over the entire FOV, as the calibration includes information at the outer portions of the cameras sensor array.

Note: This step is **extremely** important.

Masking

Introduction

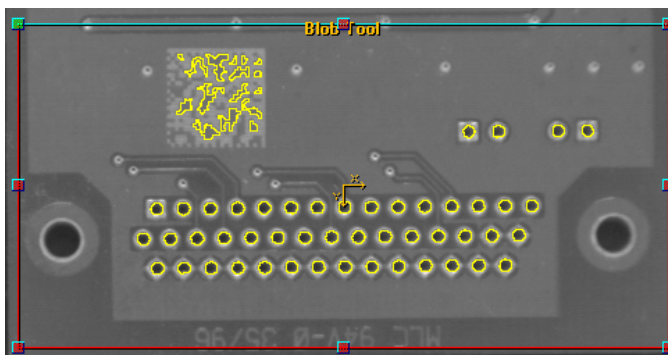
In some applications, it is not desirable to always inspect every pixel location within an ROI. When inspecting a surface area for defects, the areas that are expected to contain contrast, such as a Data Matrix code or a printed mark, will be reported as defects. With masks, you can specify areas of the ROI that a tool should not process.

The mask may be a rectangular region to cover an area such as a Data Matrix code whose content will change from one image to the next, or it may contain an image pattern that is expected to be the same from one image to the next, such as a logo.

Masking

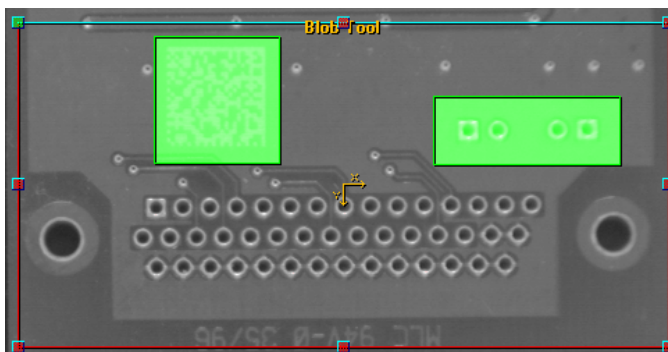
A mask specifies areas of an ROI that a tool should ignore during processing. For example, a Blob Tool can count the number of connector holes in the circuit board shown in Figure 3–1. However, nearby holes of a similar size and Data Matrix cells are inside the ROI and will be included in the analysis.

FIGURE 3–1. Example - Unmasked Tool

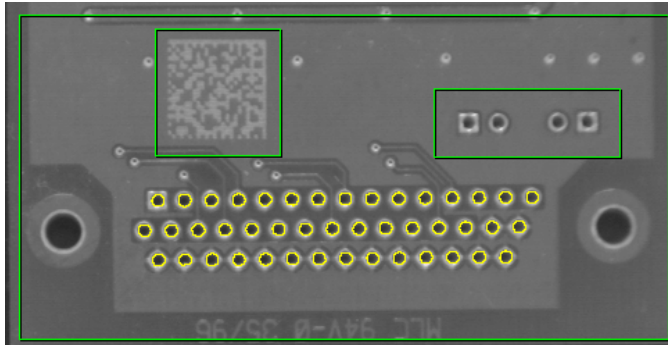


To specify that the Blob Tool should ignore these areas, you can place masks over them, as shown in Figure 3–2.

FIGURE 3–2. Example — Two Rectangular Masks

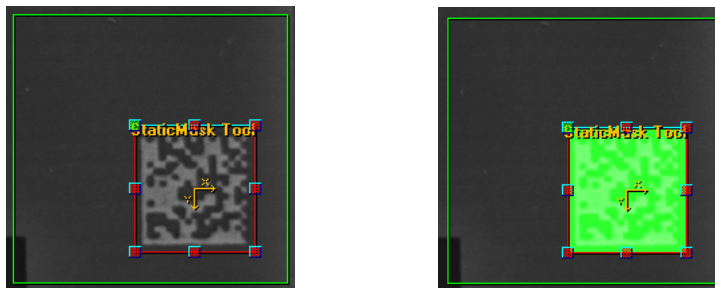


When the Blob Tool runs, only holes of the specified size are found if they are within the ROI but not covered by one of the masked regions, as shown in Figure 3–3.

FIGURE 3–3. Blobs Found with Masking

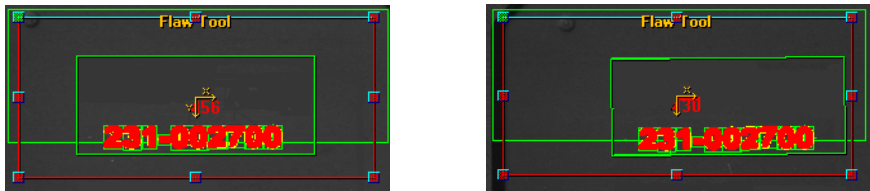
Static Masks

The pattern and location of a static mask is defined at set-up time. The mask pattern cannot be changed at runtime, and the mask location is fixed with respect to the associated ROI. This allows a tool to cover a larger area with its ROI but ignores a specified portion of that area. For example, a Flaw Tool can inspect the surface of an object, but ignore an area where a Data Matrix code is expected, as shown in Figure 3–4.

FIGURE 3–4. Masking a Data Matrix

Dynamic Masks

A dynamic mask is regenerated each time the tool executes, so the pattern and location of the mask image can change at runtime. For example, the OCV tool can generate a mask that matches the symbol positions found at runtime. These mask locations will adjust along with the symbols as they move, as shown in Figure 3–5.

FIGURE 3–5. Mask Locations — Adjusting with the Symbols

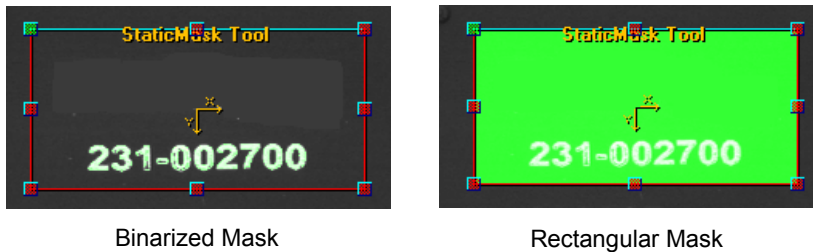
For a dynamic mask to be used by a tool, a DynamMask Tool is required between the dynamic mask and the step that is to use it. The DynamMask Tool provides the necessary coordinate transforms to align the dynamic mask to the ROI of the vision tool at runtime.

Generating Masks

Static Mask Tool

The StaticMask tool generates a mask image at training time by binarizing an image or by completely masking a rectangular area, as shown in Figure 3–6.

FIGURE 3–6. Static Mask Tool



A static mask can be inserted directly into a step and become attached to the ROI of the step that it is inserted into. The pattern of this mask will not change at runtime and the location of the mask will not move with respect to the tool that it is inserted into. If the tool is dynamically located at runtime, the static mask will move with it and always match the ROI position. The Flaw Tool and Blob Tool can generate static masks. By default, the static mask will binarize the image at train time and make a mask of what it decides are “foreground pixels”. You can make a mask of the background pixels by changing the **Polarity** in the Autothresholding step. Optionally, to get a rectangular mask, set **Mask-Generation Method** to **Fill**.

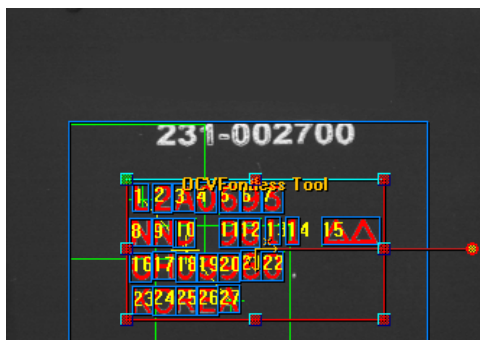
OCV Tool

The OCV tool generates a mask at runtime by combining each of its symbol masks at their runtime locations. Because the symbol masks are combined at runtime, the mask generated by the OCV tool is dynamic. This means the mask will move as the mark moves around in the view. Because the mask is dynamic, it cannot be directly used by other steps. To use this mask in another step, insert a **Dynamic Mask Step** between the OCV tool and the other step.

The OCV tool will only generate a mask image when **Enable Mask Output** is enabled (checked). The mask can be enlarged from the trained symbol templates by increasing the number of dilations. When **Graphics Level** is set to **Show Details and Mask**, the masked pixels will be drawn in red, as shown in

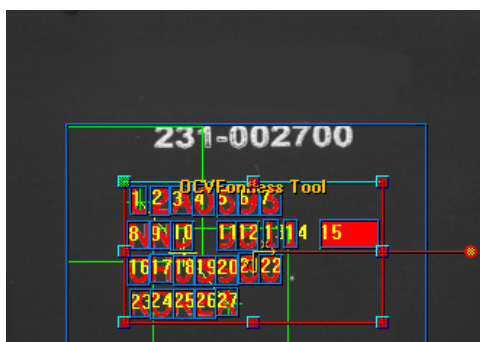
Figure 3–7. By default, each symbol will be masked by the template pattern it receives at training time but in the position found at runtime.

FIGURE 3–7. OCV Masked Pixels



The type of mask generated by each symbol can be adjusted by the **Output Mask Type** property of the OCV Symbol. In addition to masking the template pattern, the symbol's mask can be disabled, or it can mask the entire ROI of the symbol. By disabling a mask for one symbol, the symbol can be isolated for additional inspections. By masking out the entire ROI, large defects in the printing of that symbol will not affect the inspection. In Figure 3–8, symbol 15 is masked by a rectangular region so that defects in the printing of the triangle symbols are ignored.

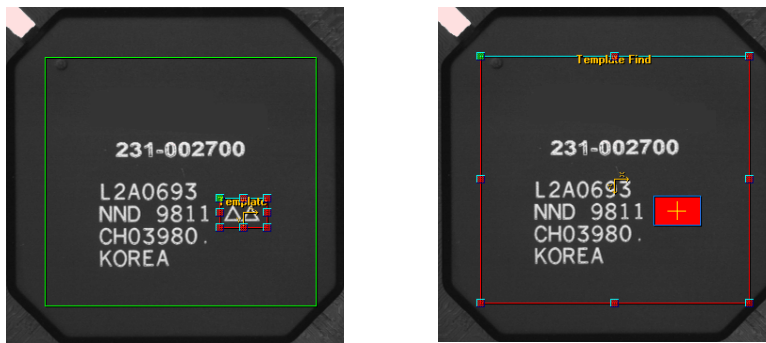
FIGURE 3–8. Mask ROI of Symbol 15



Template Find

The Template Find (Figure 3–9) will generate a rectangular mask the size of its template ROI when **Enable Mask Output** is enabled. Every time the step is run, the mask will move to the point located by the Template Find. Because it is moving at runtime, this output mask is a dynamic mask that can only be used as an input mask by a DynamMask Tool.

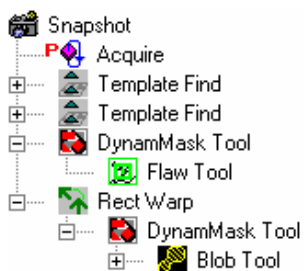
FIGURE 3–9. Template Find



To use this mask in a step, insert a Dynamic Mask between the Template Find and the step to be masked. The DynamMask Tool will align the moving mask generated by the Template Find to the ROI at runtime.

DynamMask Tool

The DynamMask Tool combines static and dynamic masks from earlier tools into a large composite mask. It generates a mask that can be used as input to steps that are inserted into it. This mask can also be used as an input to a second dynamic mask tool elsewhere in the Job. As shown in Figure 3–10, the first DynamMask Tool combines the masks generated by the two Template Finds and applies the mask to the flaw tool. The second DynamMask Tool uses the mask generated by the first one and applies it to the Blob Tool. This reduces the runtime by eliminating the need for the second DynamMask Tool to redo the combining of the Template Find masks.

FIGURE 3–10. Job Tree

Using Masks

Use Input Mask

Tools that accept a mask have a **Use Input Mask** parameter in their property page. **Use Input Mask** is enabled when a mask is added to a step and disabled when a mask is removed.

These tools can accept a mask image, from either the StaticMask or DynamMask Tool, which is aligned with its ROI. When these tools run, they can ignore pixels that are disabled by the mask image. Two tools that accept masks are Blob Tool and Flaw Tool.

Blob Tool

The Blob Tool will not process masked areas as parts when **Use Input Mask** is enabled. It is important to select the correct Blob Polarity when using masks. Otherwise, every masked area will be identified as a part, as shown in Figure 3–11.

FIGURE 3–11. Blob Polarity



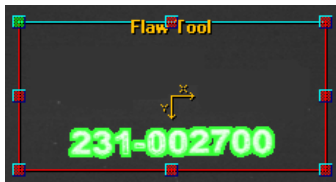
Flaw Tool

The Flaw Tool inspects for defects as small as one pixel. The pixels around the edges of the mark can be inadvertently detected as flaws, as shown in Figure 3–12.

FIGURE 3–12. Unwanted Flaw Detected

Therefore, to avoid detecting unwanted flaws, it may be useful to dilate masks that are to be used by the Flaw Tool.

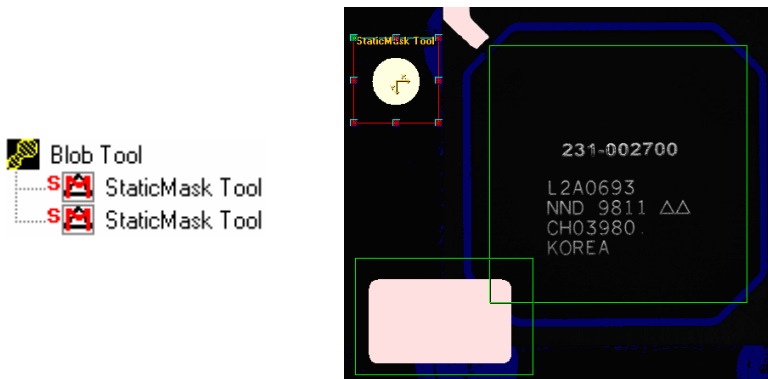
By enlarging the mask, these boundary pixels are not processed, as shown in Figure 3–13.

FIGURE 3–13. Mask Enlarged

Custom Mask Templates

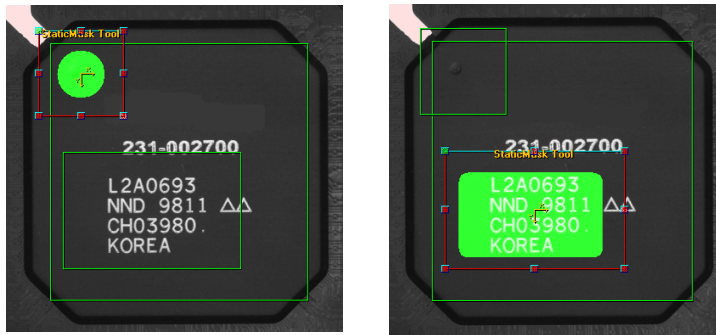
You can create custom masks by training a static mask on an image loaded from disk. You can modify the image from disk in a drawing application to include circles and other geometric shapes. In Figure 3–14, two static masks are inserted into a Blob Tool.

FIGURE 3–14. Custom Mask



With **Align Rgn at Training** disabled, they can be moved separate from the Blob Tool ROI. In setup, each of the static mask ROIs are positioned around a geometric shape that you drew onto an image. The steps are then trained to generate mask images by applying a threshold to the image for each static mask ROI.

After the mask image is generated, the image can be locked, preventing future changes. To lock a mask image, enable **Lock Mask Image** on the property page of each static mask. This allows the static masks to be positioned over the desired area of the image to be masked, without changing the mask image, as shown in Figure 3–15.

FIGURE 3–15. Positioning of Locked Masks

Once positioned, the Blob Tool can be trained again. When the Blob Tool is trained, the static masks will not train on the new image because they are locked. This will train the mask associated with the Blob Tool ROI, combining the two static masks at their new location, as shown in Figure 3–16.

FIGURE 3–16. Combined Custom Masks

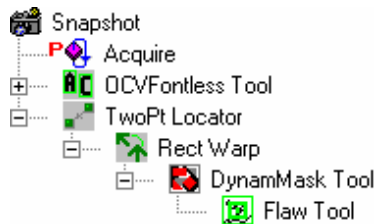
The Blob Tool will use this new mask when it runs.

Sample Applications

Simple Mark/Package Inspection

To inspect a printed mark on a package, an OCV tool can be used. To inspect a package surface and ignore the marked area, the mark needs to be masked out. This Job application has been created, as shown in Figure 3–17.

FIGURE 3–17. Mark/Package Inspection



The OCVFontless Tool will verify the marked area. The Flaw Tool will inspect the surface area. To mask out the mark area from the Flaw Tool, a mask generated from the OCVFontless tool is applied to the Flaw Tool by way of a DynamMask Tool.

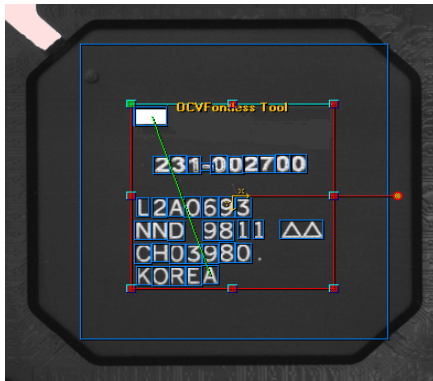
By default, the OCVFontless Tool will not generate a mask. To generate a mask, display the OCVFontless Tool properties page and enable (check) **Enable Mask Output**. Add some dilations to mask out boundary pixels around the mark by typing 2 in the **Mask: Number of Dilations** parameter.

Note: Dilation can be applied right in the OCV tool or, in the case of a Dynamic Mask Step that combines many masks, the dilation can be performed in the Dynamic Mask, thus doing it only once.

The TwoPt Locator can be connected to the OCVFontless tool's Autofind method to keep the two locators synchronized.

From Setup view, train the OCVFontless Tool over the mark, as shown in Figure 3–18.

FIGURE 3–18. Train OCVFontless Tool



Step through the Job and position the Flaw Tool ROI over the surface of the object to inspect. When the Flaw Tool is run, the marks found by the OCVFontless Tool will be ignored, as shown in Figure 3–19.

FIGURE 3–19. Marks Ignored



Static Masks From Different Buffers

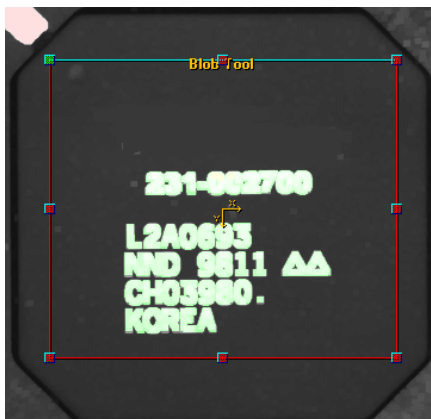
Static masks can only be directly attached to one step. When a static mask is inserted into a step, other steps can also use the mask image with the help of a DynamMask Tool. In Figure 3–20, a StaticMask Tool is inserted into a Blob Tool that is operating on the morphed image. The Flaw Tool that is in a different image can use that same mask by using a DynamMask Tool between the StaticMask Tool and the Flaw Tool.

FIGURE 3–20. Job Example



Training the Blob Tool generates a static mask that will be used by the Blob Tool at runtime. This static mask is created by applying a threshold to the morphed image, as shown in Figure 3–21.

FIGURE 3–21. Applying a Threshold to Morphed Image



The DynamMask Tool will take this static mask as input, convert it to the reference frame of the Snapshot, and apply the mask to the Flaw Tool ROI, as shown in Figure 3–22.

FIGURE 3-22. Apply Mask to Flaw Tool

References

The remainder of this chapter describes the following steps:

- “AutoThreshold” on page 3-17
- “Compute Polarity” on page 3-19
- “DynamMask Tool” on page 3-21
- “StaticMask Tool” on page 3-27

AutoThreshold and Compute Polarity automatically create static masks. StaticMask Tool and DynamMask Tool are the masks themselves.

AutoThreshold

AutoThreshold cannot be inserted directly into a Job. It only appears in Job programs when it is used by other steps that need it.

AutoThreshold determines a threshold, which is the gray scale value that best separates the foreground pixel values from the background pixel values in the given ROI. In conjunction with the Computer Polarity step, the Output Threshold value determines which pixels are foreground pixels and which pixels are background pixels. Masks can be generated from either group.

The inputs to AutoThreshold are the image and ROI in which to process. The outputs of this step are the pass/fail status and the calculated threshold value.

Other Steps Used

None.

Theory of Operation

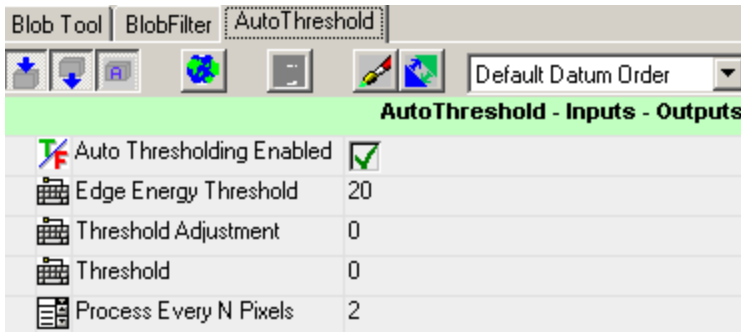
AutoThreshold uses edge enhancement and pixel counting to calculate the best gray scale value for separating foreground pixels from background pixels. A user adjustment allows manual adjustment of this dynamically calculated threshold value.

The ROI within which the threshold is to be determined can be adjusted by moving and sizing the search area shape.

Description

AutoThreshold allows editing through the AutoThreshold properties page, as shown in Figure 3–23.

FIGURE 3–23. AutoThreshold Properties Page



Settings

- **Auto Thresholding Enabled** — Enables and disables automatic thresholding:
 - When **enabled** (default), a threshold will be calculated using the ROI of the parent step. This calculation uses edge detection to determine foreground/background information. The calculated threshold is displayed in **Threshold**.
 - When **disabled**, no calculation is performed. The threshold used by the parent step is the value **Threshold**. **Edge Energy Threshold** and **Threshold Adjustment** are not used when this property is disabled.
- **Edge Energy Threshold** — Defines the pixel value at which a pixel in a Sobel Edge Enhancement is considered to be an edge pixel. This property is only used when **Auto Thresholding Enabled** is enabled.

Default: 20

Range: 0 to 255

- **Threshold Adjustment** — Used to offset or bias the dynamically calculated threshold, when **Auto Thresholding Enabled** is enabled.

Default: 0

Range: -255 to 255

- **Threshold** — Displays the dynamically calculated threshold when **Auto Thresholding Enabled** is enabled. When **Auto Thresholding Enabled** is disabled, the value of this property is the threshold that will be used by the parent step.

Default: 135

Range: 0 to 255

- **Process Every N Pixels** — This property can enhance significantly the tool performance at a small cost in accuracy by processing less boundary pixels.

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.
- **Calculated Threshold** — The final threshold value equal to the threshold value plus the Threshold Adjustment value.

I/O Summary

None.

Compute Polarity

ComputePolarity is a private step in that it cannot be inserted directly into a step program. It only appears in step programs when it is used by other steps.

ComputePolarity determines the foreground to background relationship within the given ROI. For example, this step can determine if there is light print on a dark background or dark print on a light background.

The inputs to ComputePolarity are the image and ROI on which to process, as well as a threshold for binarization of the ROI. The outputs are the pass/fail status and the computed polarity value. The computed polarity determines which pixels generate the mask. By default, foreground pixels will be chosen, but either polarity can be selected deliberately.

Other Steps Used

None.

Theory of Operation

When automatic polarity calculation is enabled, this step samples pixels within the ROI and counts the number above the input threshold and below the input threshold. The greater count is considered to be that of background pixels.

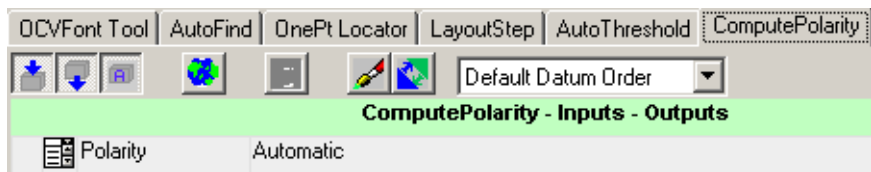
There is one user adjustment for enabling/disabling automatic polarity calculations.

The ROI within which the polarity is to be determined can be adjusted by moving and sizing the search area shape associated with the parent step. For automatic polarity calculations, it is assumed that the feature being inspected is centered within the ROI.

Description

ComputePolarity allows editing through the ComputePolarity properties page, as shown in Figure 3–24.

FIGURE 3–24. ComputePolarity Properties Pages



Settings

- **Polarity** — Allows the step to be set up to always return LIGHT_ON_DARK, always return DARK_ON_LIGHT, or return an automatically determined polarity.

Default: Automatic

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.
- **Computed Polarity** — The polarity determined by the step when in automatic mode or in equivalent to the selected polarity where:

0 — Light on Dark

1 — Dark on Light

I/O Summary

None.

DynamMask Tool

A DynamMask Tool generates a mask during runtime. Because this tool combines masks generated by other steps, it should be placed after these masks to be combined. It needs to execute and generate its mask before any of the steps that will use it.

The DynamMask Tool should be inserted just above the tools that are to use the mask generated. The DynamMask Tool will not generate an output image other than the mask, so any steps inserted into it will use the same input buffer as the DynamMask Tool. The DynamMask Tool does not perform any operations on the input image, but the placement of the tool will determine the reference frame of the mask generated. This mask needs to be in the same reference frame as all of the steps that will use the generated mask.

In Figure 3–25, the first DynamMask Tool will generate a mask in the reference frame of the Snapshot and apply it to the Flaw Tool, which will also use the Snapshot image. The second DynamMask Tool will generate a mask in the reference frame of the output image generated by the GrayMorph filter. The Flaw Tool will then operate on the image generated by the GrayMorph with the mask generated by the parent DynamMask Tool.

FIGURE 3–25. DynamMask Tool — Example Job



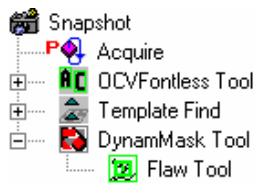
Other Steps Used

None.

Theory of Operation

The DynamMask Tool combines masks that are generated by other tools, so those input masks must be valid before the DynamicMask Tool is run. This requires that the DynamicMask Tool be inserted after any of the mask-generating steps whose output is to be used by the DynamMask Tool, as shown in Figure 3–26.

FIGURE 3–26. Job - Example 1



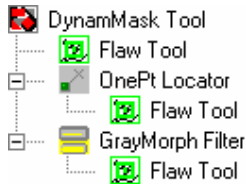
Only child steps of the DynamicMask Tool can use the mask generated by the DynamicMask Tool. This will guarantee that the dynamic mask tool will execute before the any of the steps using the mask.

When a mask-using step is inserted into a DynamMask Tool, the step will be automatically added to the DynamMask Tool parameter **Steps to Mask** list. You can remove child steps that were added automatically or add additional steps to be masked here.

The **Use Input Mask** parameter of the step will be enabled when it is added to this list.

Only steps that use the same input image as the dynamic mask tool can be added to the list.

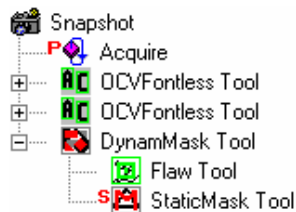
In Figure 3–27, the Flaw Tool under the OnePt Locator can be masked by this DynamMask Tool because it will use the same image as the DynamMask Tool. The OnePt Locator does not generate a separate output image. The GrayMorph Filter does generate a separate output image to be used by the last Flaw Tool.

FIGURE 3–27. Job -Example 2

This DynamMask Tool cannot mask the last Flaw Tool. Only vision tools that are eligible to be masked by this tool will appear as blue in the selection dialog.

When the DynamMask Tool runs, it will combine all input masks into a single mask in the reference frame of the input buffer. This combined mask is then sectioned to fit the ROI of each child step that is attached. If the DynamMask Tool contains a static mask, it will also be merged into the combined output mask.

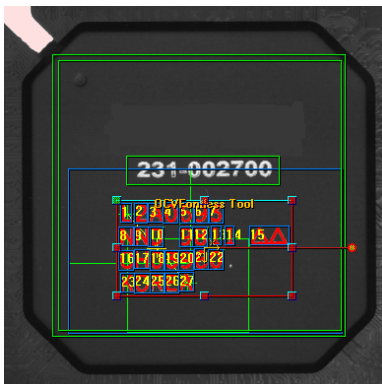
In Figure 3–28, two separate OCVFontless Tools are used.

FIGURE 3–28. OCV Example

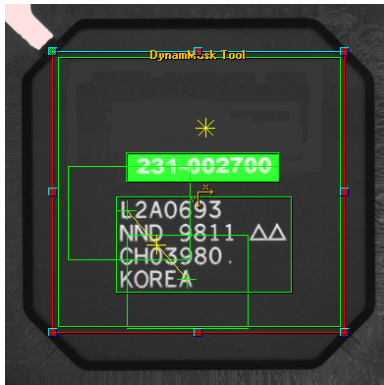
The first OCVFontless Tool will treat the logo as a single symbol, as shown in Figure 3–29.

FIGURE 3–29. Single Symbol

The second OCVFontless Tool will treat each of the lower characters as separate symbols, as shown in Figure 3–30.

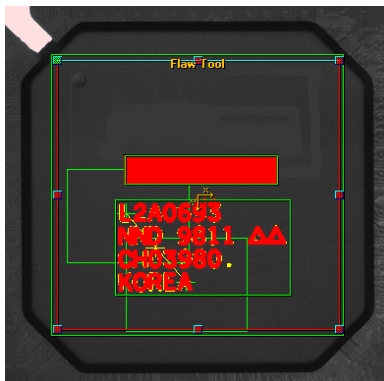
FIGURE 3–30. Separate Symbols

In addition, a static mask covers the numbers below the logo, as shown in Figure 3–31.

FIGURE 3–31. Static Mask

When **Graphics Level** is set to **Show Details and Mask**, the **DynamMask Tool** will be drawn in red on the image after the step that contains the mask is run. The static mask is shown as transparent green and will move with the **DynamMask Tool ROI**.

When the step is run, all of the input masks are combined into a single mask that is applied to the **Flaw Tool**. By setting the **Flaw Tool** parameter **Graphics Level** to **Show Details and Mask**, the mask used by the **Flaw Tool** is drawn when it executes, as shown in Figure 3–32.

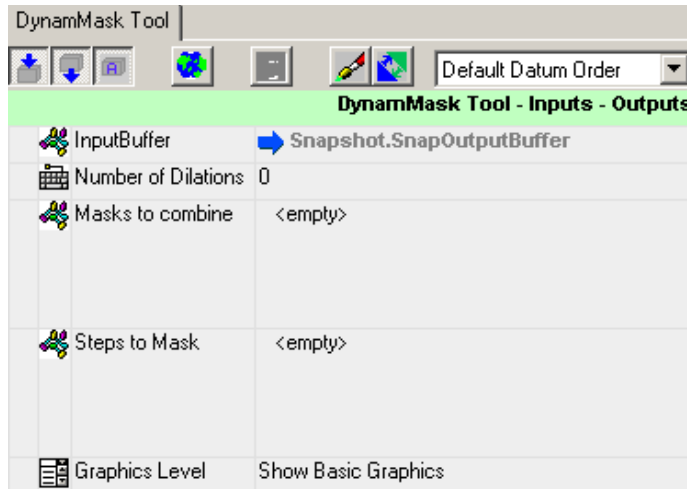
FIGURE 3–32. Single Mask

The mask used by each of the steps can be viewed by setting the **Graphics Level** to **Show Details and Mask** of those steps.

Description

DynamMask Tool allows editing through the DynamMask properties page, as shown in Figure 3–33.

FIGURE 3–33. DynamMask Tool Properties Page



Settings

- **Number of Dilations** — Can be increased to enlarge the masked area. When a mask is generated by applying a threshold to an image, there will be some pixels around the edge of the symbol that will be between the foreground and background gray levels. These border pixels may produce undesirable results when the step runs.

Default: 0

- **Masks to combine** — Contains a list of masks to be merged into one mask by the DynamMask Tool. By default, any masks that exist where the tool is inserted are added to the list when the DynamMask Tool is inserted. The mask output must be enabled by the step that generates it before the step can be added to the list. You can add and remove masks from the list as applicable. The masks are noted by the name of the step that produces the mask.

- **Steps to Mask** — Contains a list of steps to receive the mask generated by the dynamic mask tool. When the step runs, each step listed here will receive a mask image that matches its ROI size and reference frame. When a step is inserted into the dynamic mask tool, it will be added to the list automatically. Only direct child steps are added automatically. However, other descendants can also be added if they are in the same reference frame. **Use Input Mask** of the step must be enabled for it to use the mask when it runs. **Use Input Mask** will be enabled automatically when the step is added to this list.
- **Graphics Level** — When set to Show Details or Show Details and Mask, the mask generated by the DynamMask Tool will be drawn in red on the image.

Training

None.

Results

- **StatusDm** — Status of step execution. Set to true when the step successfully executes.
- **MaskDm** — Resulting combined dynamic mask.

I/O Summary

None.

StaticMask Tool

The StaticMask Tool can be inserted into any step that supports masking, such as the Flaw Tool or Blob Tool. With this program structure, the mask will be created at set-up time and is attached to the ROI of the Flaw Tool. If the Flaw Tool ROI is moved after setup, the mask will move with it. Since the StaticMask Tool is a set-up step, denoted by an S next to the icon, it is configured through the parent step it is inserted into. A tool becomes trainable when a static mask tool is inserted into it.

A tool can accept a combination of static and dynamic masks. The StaticMask Tool can be inserted into a DynamicMask Tool or into a tool that is in a DynamicMask Tool, as shown in Figure 3–34.

FIGURE 3–34. Mask Configurations

In each case, the static mask is generated at training time (by a Blob Tool or a Flaw Tool), and combined with the dynamic mask at runtime. However, in the first case, the static mask is attached to the DynamMask Tool ROI and is trained by training the DynamMask Tool. In the other case, it is attached to the Flaw Tool ROI and trained by training the Flaw Tool.

Multiple static masks can be inserted into a step or the dynamic mask step. When the parent step is trained, the static masks will be combined into a single mask and attached to its ROI. Because the StaticMask Tool is a set-up step, no steps can be inserted into it.

Other Steps Used

- **Auto Threshold** — Automatically calculates the threshold for binarizing the area of interest.
- **ComputePolarity** — Determines if there are dark characters on a light background or light characters on a dark background.

Theory of Operation

To create the mask area, the StaticMask Tool first calculates a threshold and binarizes the ROI. The inverse of the binary image is then stored in a bit-packed template, called the mask. When the mask is used at runtime, 0's indicate no computations should be performed and 1's indicate computations should be performed. You can adjust this to allow expanding of the area that is masked.

The ROI within which the mask is to be created can be adjusted by moving and sizing the search area shape associated with the to-be-masked step.

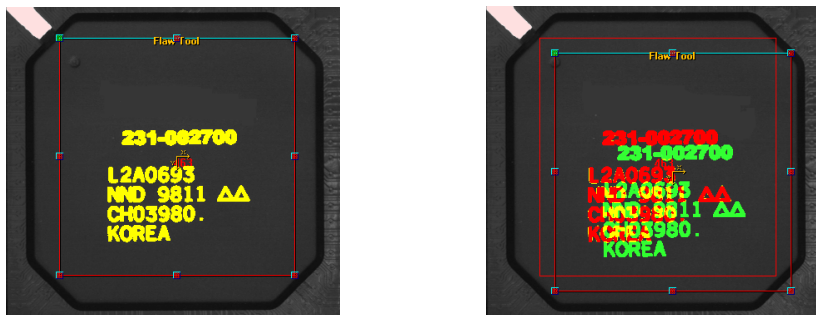
Because the mask image of a StaticMask Tool is determined at training time, there is no runtime associated with this step. However, the step that it is inserted into will generally require additional processing time when using a mask.

After a tool containing a static mask is trained, the mask that is attached to the ROI will appear as a transparent green. This green mask shows you the static mask pattern while positioning the ROI, as shown in Figure 3–35. It will not appear during runtime.

FIGURE 3–35. Static Mask

At runtime, the mask will be drawn as solid red when the parameter Graphics Level is set to Show Details and Mask.

In Figure 3–36, the mask used by the Flaw Tool at runtime is drawn on the image as solid red. While in setup, the static mask is drawn in transparent green. The solid red combined with the transparent green static mask will appear as yellow in setup when they overlap.

FIGURE 3–36. Masks

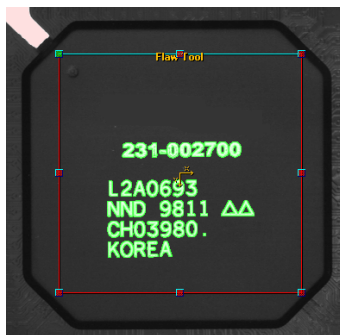
Runtime and Static mask overlapped Static mask repositioned

The red runtime mask image is drawn onto the image buffer as a result of the last run and will not change until the next tryout or run. The green static mask is attached to the ROI and will move as the ROI position is adjusted.

ROI Alignment

By default, the parameter **Align Rgn at Training** is enabled. This allows the StaticMask Tool to share an ROI with the tool it is inserted into. In applications that only require one static mask applied to the tool's entire ROI, this will make the setup task easier, by eliminating one ROI you have to position, as shown in Figure 3–37.

FIGURE 3–37. Align Rgn at Training — Enabled



When disabled, you will need to position the StaticMask Tool ROI separately from positioning the vision tool's ROI, as shown in Figure 3–38.

FIGURE 3–38. Align Rgn at Training — Disabled



Fill vs. Auto-Threshold

By default, the StaticMask Tool will use the Auto-Threshold method to generate a mask. This will create a mask by thresholding the image using the AutoThreshold and ComputePolarity steps, as shown in Figure 3–39.

FIGURE 3–39. Mask Generation Method — Auto-Threshold



By selecting Fill, the entire area of the StaticMask Tool ROI is masked. In this case, the mask is not dependent on the image used during training. However, the mask will always cover a rectangular area of the image, as shown in Figure 3–40.

FIGURE 3–40. Mask Generation Method — Fill



Multiple static masks set to Fill can be used to cover sections of an image, as shown in Figure 3–41.

FIGURE 3–41. Multiple Static Masks

It is generally not advisable to enable both the Fill and Align Rgn at Training parameters. This would cause the entire ROI to be masked, leaving no pixels to be processed by the step, as shown in Figure 3–42.

FIGURE 3–42. No Pixels Processed

Auto Threshold Adjustments

When Auto Thresholding Enabled of mask generation is enabled, the threshold and polarity are determined automatically. By disabling Auto Thresholding Enabled, you can set the threshold level directly.

To let the system pick a threshold automatically but offset it by a fixed amount, use Threshold Adjustment. Usually, Threshold Adjustment should be used only when Auto Thresholding is enabled, and set to 0 when Auto Thresholding is disabled.

By default, the polarity of the mask will be determined automatically. You can set the polarity manually on the ComputePolarity step properties page.

Mask Adjustments

When a mask is generated by thresholding an image, there will often be pixels around the edge of the mask that are between the background and foreground gray levels. In Figure 3–43, the pixels around the edge of the mark are detected as errors by a Flaw Tool.

FIGURE 3–43. Pixels Detected as Errors



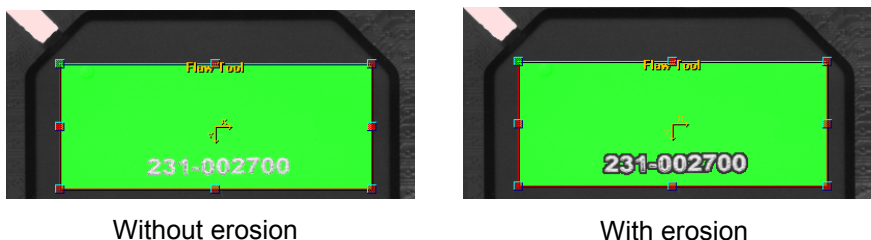
Use Number of Adjustments of the StaticMask Tool to enlarge the mask to cover these boundary pixels. To enlarge the mask, set Adjustment Type to Dilation and Number of Adjustments to the number of pixels to expand the mask boundary. In Figure 3–44, the boundary of the mask is expanded by two pixels to mask the edges of the mark.

FIGURE 3–44. Boundary Expanded by Two Pixels



The mask can also be reduced by selecting Erosion as the Adjustment Type. This is useful when creating windows for inspection. By selecting the reverse mask polarity in the ComputePolarity property page, a mask is generated covering the entire ROI except for the mark area, as shown in Figure 3–45. Selecting Erosion can expand non-masked areas.

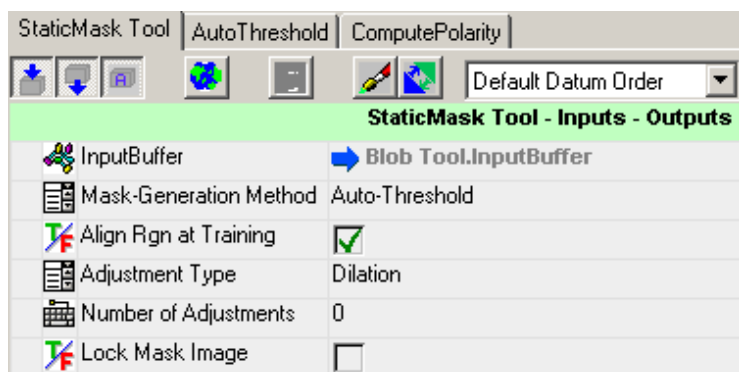
FIGURE 3–45. Reverse Mask



Description

StaticMask Tool allows editing through the StaticMask Tool properties page, as shown in Figure 3–46.

FIGURE 3–46. StaticMask Tool Properties Page



Settings

- **Mask-Generation Method** — Specifies the type of mask to create at training time:
 - **Fill** — Masks the entire ROI area.
 - **Auto-Threshold (Default)** — Auto Threshold and Compute Polarity will be used to generate a binary mask from the image.
- **Align Rgn at Training** — When enabled, the ROI of the static mask will be the same as the ROI of the parent step. When disabled, the ROI of the static mask tool can be moved independent of the parent step's ROI position.

Default: Enabled

- **Adjustment Type** — Specifies whether to perform a mask expansion or reduction. This will only have an effect when the **Number of Adjustments** is greater than 0 and **Mask-Generation Method** is set to Auto-Threshold:
 - **Dilation (Default)** — Expands the masked area.
 - **Erosion** — Reduces the masked area.
- **Number of Adjustments** — Specifies the number of dilations or erosions to perform. A value of 0 disables any modification of the mask.

Default: 0

- **Lock Mask Image** — When enabled, this mask pattern will not change after the parent step is trained. However, its position within the parent step's ROI may change.

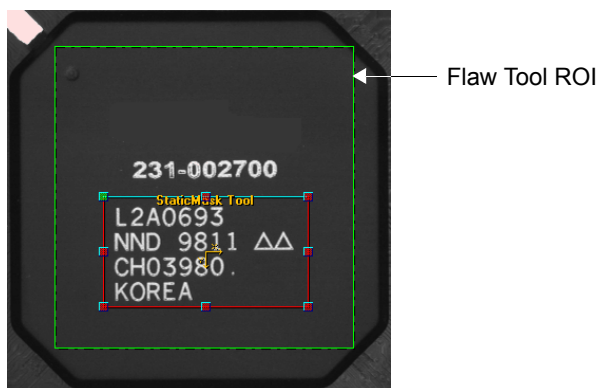
Default: Disabled

Training

The StaticMask Tool is a set-up step and is trained when its parent step is trained. Static masks will not be displayed in the Job Tree. However, the StaticMask Tool properties page is visible as a separate tab on the parent step's properties page.

Each static mask will have its own ROI when **Align Rgn at Training** is disabled. Otherwise, the ROI of the static mask is the same as the ROI of the parent step. This ROI can be positioned anywhere in the image, independent of the position of the parent step ROI.

In Figure 3–47, the static mask ROI covers the text to be masked so the Flaw Tool will only examine the remaining text.

FIGURE 3–47. Static Mask

When trained, the masked area will appear transparent green, as shown in Figure 3–48.

FIGURE 3–48. Static Mask Trained

The static mask ROI can extend beyond the ROI of the parent step. However, only the area that overlaps the parent step ROI will have any effect on the step.

There are two mask images created at training. The first is attached to the StaticMask Tool ROI and is the full size of that ROI, as shown in Figure 3–49.

FIGURE 3–49. StaticMask Tool ROI

The second is the static mask as applied to the FlawTool ROI. This mask is cropped to fit this ROI and will move with the FlawTool ROI as it is positioned, as shown in Figure 3–50.

FIGURE 3–50. Static Mask Applied to Flaw Tool ROI

Results

- Status — Set to true after a successful execution of the step.
- MaskDm — The bit-packed output mask can be used by the originator step to exclude areas within the ROI from being included in the calculations.

I/O Summary

None.

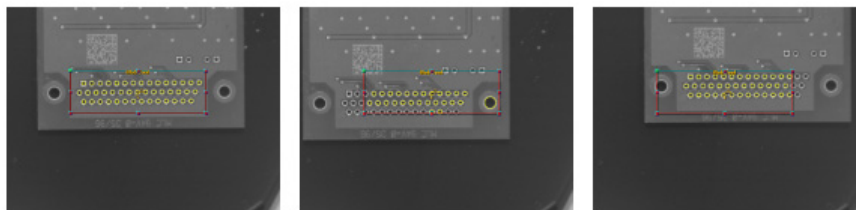
Dynamic Location

The location of an object within an image will not always repeat exactly from one image acquisition to the next. An object may move within the field of view (FOV), or rotate. When an inspection ROI needs to be precisely aligned with an object in the FOV, a dynamic locator can align the ROI with the object.

Dynamic Location is the process of moving the ROI of a tool based on other features within the image. These other features can be based on template finds or any other points that can be located or defined, such as blob filter results or intersection of two lines.

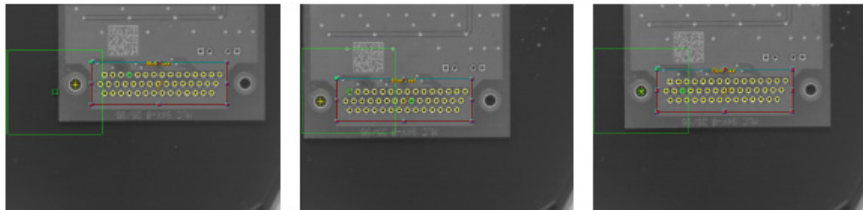
Using the examples shown in Figure 4–1, the goal is to perform a blob analysis on the connector holes of a circuit board. The position of the board within the FOV is not consistent from one image to the next. The inspection fails because the board moved but the Blob Tool ROI did not. For a successful execution, the Blob Tool ROI needs to move with the board.

FIGURE 4–1. Example of an ROI That is Not Dynamically Located



A feature of the circuit board should be selected as a reference to move the Blob Tool ROI. In Figure 4–2, the large left mounting hole is a good reference location. A Template Find locates the left hole, and the Blob Tool ROI is moved according to the position of that result. This procedure keeps the blob analysis at a fixed location on the circuit board.

FIGURE 4–2. Example of an ROI That is Dynamically Located



Point Locators

The location process can be defined to use 1, 2 or 3 reference or control points. The number of input points is determined by the tool that is inserted: OnePt Locator, TwoPt Locator or ThreePt Locator. The number of input points cannot be changed without deleting the step and inserting a new locator of a different type.

Center Point

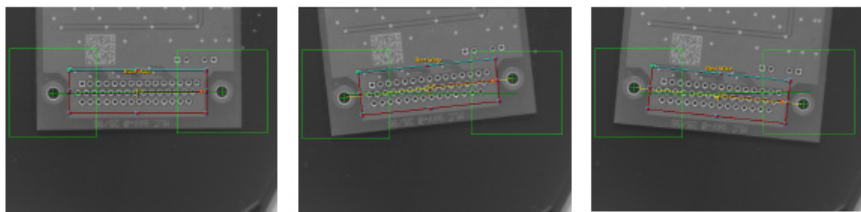
The point specified by the combination of the reference input points is called the center point. The location of the center point is the average of the input control points. In the case of a OnePt Locator, it will be equivalent to the one input control point.

Rotation

In some cases, an x,y translation is not sufficient, as the part may rotate within the image. An ROI can be rotated when two or three input control points are used.

In Figure 4–3, both mounting holes define the board position in a manner that includes rotation.

FIGURE 4–3. Rotation



Some ROIs, such as the Blob Tool, cannot be rotated. In these cases, the two input control points calculate a new x,y position for the ROI, although it is not rotated. Very few ROIs can be rotated. One that can is called the RectWarp. The OnePt, TwoPt, or ThreePt Locator can change the x, y and angle of this tool. The RectWarp Tool is used whenever it is necessary to apply rotation to any of the tools, such as the Blob Tool. It is accomplished by placing these tools inside the RectWarp. The RectWarp grabs the pixels from the original buffer and creates a new sub-buffer with the image oriented correctly. The tools that are placed inside it run normally. In effect, instead of rotating the axis, we rotate and shift the image under the ROI. For more information, see “Reference Frames” on page 4-24.

Reference Location

An NPt Locator must be trained during setup before it can be run. During training, the initial center point location is calculated. This initial location defines the reference location. The reference location is the basis for defining the relationships between the ROIs to be moved and the input control points.

Location Adjustment

During each run of an NPt Locator step, a new center point will be calculated. The difference between the location of the center point at runtime and the reference location determined at training time is called the location adjustment. This adjustment value will be applied to each shape to be moved by the NPt Locator step.

Building a Job

By default, the NPt Locator uses integrated Template Find steps to determine the reference location. When using the integrated template finds, the steps will use the image buffer determined by the placement of the NPt Locator step. When the input control points are referenced to an external input, the NPt Locator step does not depend on the input image.

In the simplest Job using an NPt Locator, the integrated Template Find steps determine the reference location and tools to be moved are inserted into the NPt Locator, as shown in Figure 4–4.

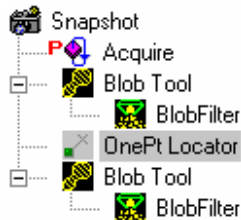
FIGURE 4–4. Simple Job



In the case of a simple Job like the one illustrated in Figure 4–4, the correct execution order is generated automatically. The OnePt Locator executes its integrated Template Find steps to determine the offset amount and adjust the ROIs of the child steps. The child step, Blob Tool, is then executed.

In Figure 4–5, a Blob Tool locates the large hole. That hole position is used by the OnePt Locator to determine the offset amount. This is done by telling the OnePt Locator to use the point returned from Blob rather than its own internal find. When you do this, it disabled the integrated template find automatically. The ROI of the following Blob Tool, which is not a child step of the OnePt Locator, is adjusted.

FIGURE 4–5. Simple Job with External Connections



In this case, you need to be aware of the execution order when constructing the Job. The NPt Locator must be placed after steps used as input connection references, but placed before any steps whose ROIs are to be moved by the NPt Locator. Tools that are to be moved by the Locator Tool must be added to the Locator's steplist unless they are inserted directly into the Locator Step.

Steps can be inserted directly into an NPt Locator. When steps that contain an ROI are inserted into the NPt Locator, they will be added to the locator's **Step List**, and dynamic location will be automatically applied. If steps are not inserted directly as children, they may be set to be moved by the locator by "Adding" them to the locator steplist.

Inserting a RectWarp

The One, Two, and ThreePt Locators can provide rotation of ROIs. However, the ROIs of most tools cannot be rotated. When these tool ROIs are moved by an NPt Locator, only the x,y location of the ROIs will be modified. To provide rotation, a RectWarp must be used between the NPt Locator and the tool.

In Figure 4–6, the ROI of the RectWarp, which supports rotation, is moved by the TwoPt Locator. The Blob Tool operates in the rotation compensated image output by the RectWarp.

FIGURE 4–6. RectWarp Inserted Between Tools



Setting Up a Job

Before the NPt Locator can be run, it must be trained to define the templates for the integrated find steps and establish the reference location.

Internal vs. External Connections

The training will behave slightly differently depending on whether the inputs are connected to the integrated find steps or referenced to external inputs. When using the integrated template finds, you position the input points by the placement of the template ROIs. The Template Finds will be trained at these locations so that the reference positions of the input connections are defined by you.

When connected to an external input, the step that is referenced will first be run to update the result of each connected input. The input control points will be moved to these result points and the reference locations will be established.

In Figure 4–7, the integrated find steps are used. You position the find step ROIs, which will also define the reference location of the NPt Locator.

FIGURE 4–7. Training with Integrated Template Finds

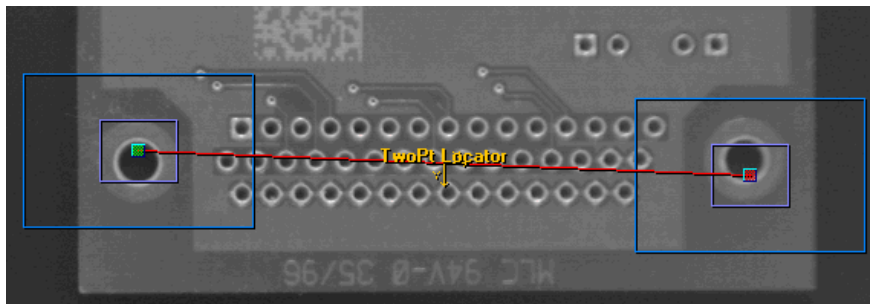
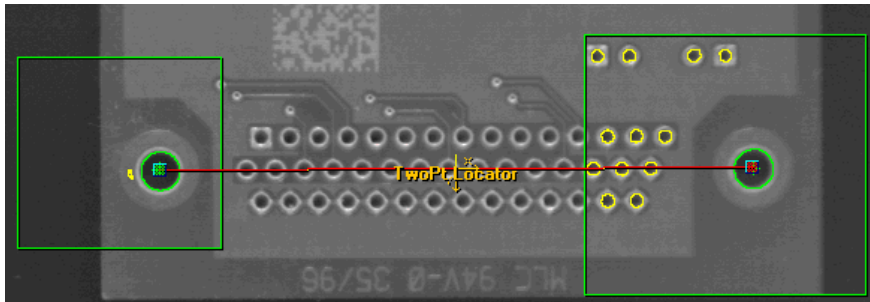


Figure 4–8 shows two Blob Filter steps used for the input connections. The result of the Blob Filter will define the input reference locations of the NPt Locator. There are no objects of the NPt Locator you have to position.

FIGURE 4–8. Training with External Connections



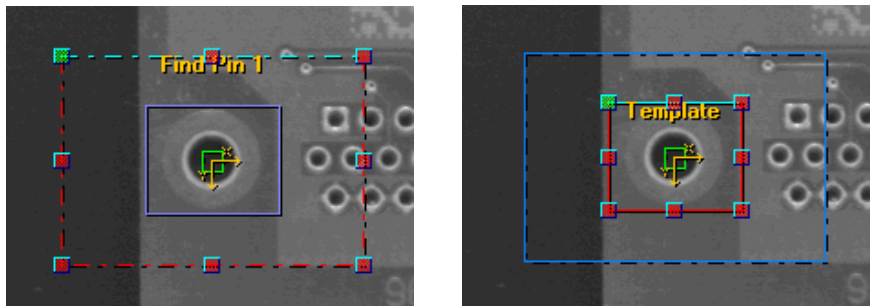
Positioning of ROIs

The NPt Locator has no ROI of its own. With the TwoPt and ThreePt locators, there are one or more lines drawn that connect the input points. These lines are for feedback only and do not need to be adjusted. However, when using the integrated find steps, the template find ROIs do need to be positioned.

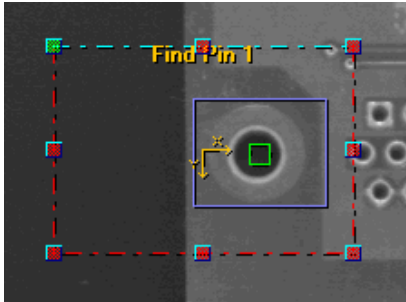
Integrated Find

When an NPt Locator is inserted into a Job, it will contain an integrated Template Find step for each input control point, as shown in Figure 4–9. These locate one to three patterns within the image.

FIGURE 4–9. Positioning of Integrated Template Find ROI



Each integrated template find will have a search area ROI and template ROI. Position the Template ROI around the pattern to be matched. The center of the Template ROI will be the input control point reference. Position the Find Pin ROI to cover the desired search area for the find step, as shown in Figure 4–10.

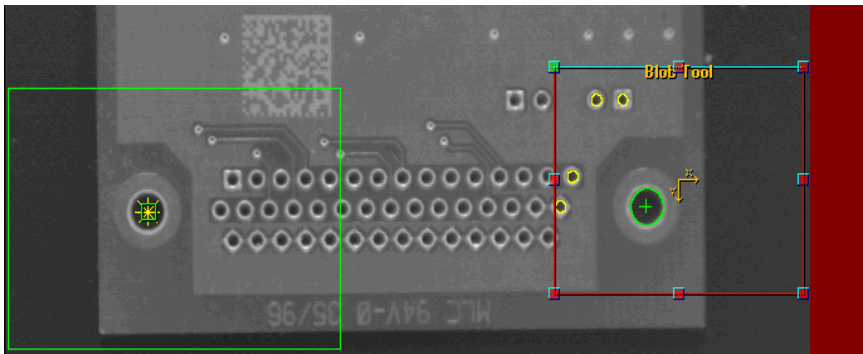
FIGURE 4–10. Disable Center Feature Search

By default, the template ROI will be centered within the search area ROI. You can turn off this feature by disabling the **Center Feature Search** checkbox in the datum display of the NPt Locator.

Controlled ROIs

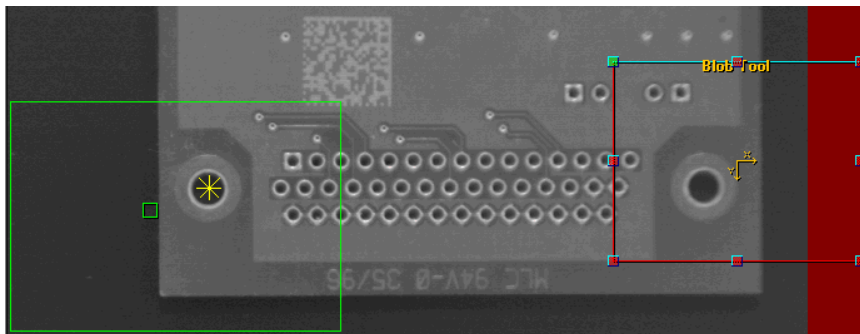
When you are setting up a Job, the location of ROIs will be based on a reference image. The ROIs of steps that are to be moved by the NPt Locator should be placed over the area of interest in the reference image. There are no restrictions on the placement of the ROIs relative to the NPt Locator input control points.

Because the ROIs will be moving, they should not be placed too close to the edge of the window. If the object moves such that it pushes an ROI off the window, that step may fail to run. In Figure 4–11, the Blob Tool ROI is placed too close to the right edge of the buffer in the reference image during setup.

FIGURE 4–11. ROI Close to Window Edge

If the NPt Locator moves the ROI to the right at runtime, it may fall outside of the window, as shown in Figure 4–12. This will cause the Blob Tool to fail to run.

FIGURE 4–12. ROI Too Close to Window Edge



The controlled step's ROIs can be repositioned after the NPt Locator is trained.

Creating Reference Positions

Training the NPt Locator establishes the reference positions of the controlled step's ROIs relative to the input control points. The location of each controlled ROI is stored for reference. During runtime, the difference between the runtime location of the inputs and reference location of the inputs will determine the location adjustment. This adjustment will be applied to the stored reference locations of each controlled ROI to determine the new position of the ROI. The reference location will be updated whenever you adjust the controlled ROI while setting up the Job.

Running a Job

When the NPt Locator executes, it will first execute any integrated Template Find step that is connected. The best match point for each Template Find step is used for the input point to the NPt Locator. The parameters for the find steps can be modified by clicking the Find Pin N tab on the NPt Locator property page.

When the integrated finds are not used by the NPt Locator, because the inputs are connected to other steps, the Template Find steps will not execute. By not running the Template Find when that result is not needed, you reduce the execution time for the Job. When the Template Finds are not used by the NPt Locator, the result datums of those steps cannot be used elsewhere in the Job, since they are not be valid.

Calculation of Location Adjustment

Each input pin point connection is referenced to one output point datum. These datums are combined to a single point result, the Center Point. The center point contains the x,y coordinates that are the center of the input points, and the rotation angle. The difference between the location of the center point at runtime and the reference location determined at setup time is calculated as the Location Adjustment.

Movement of ROIs

Once the adjustment amount has been calculated, it is applied to the location of the ROI of each controlled step. The location adjustment is added to the reference location of each shape that was stored at training time. The next time the controlled step executes, it will operate in the new area of the image specified by the moved ROI.

Advanced Configurations

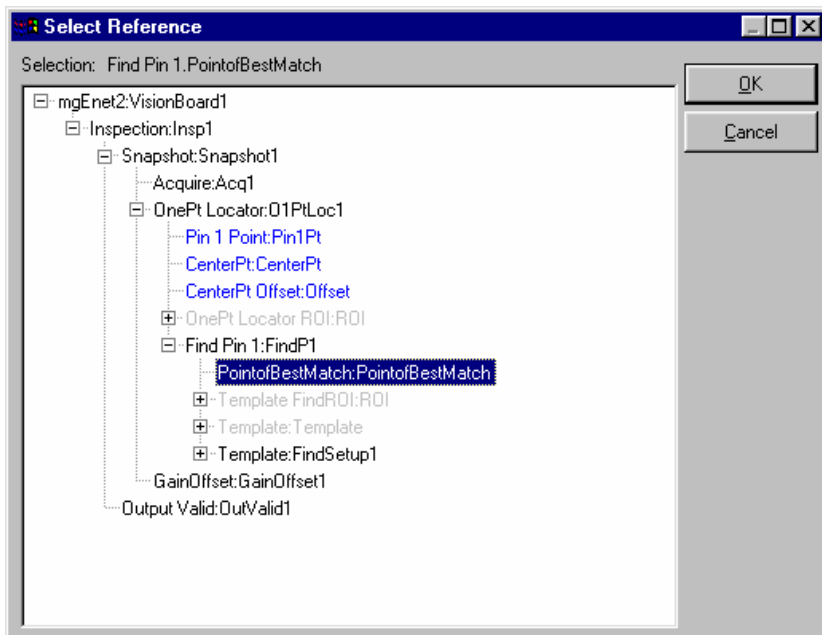
Input Point Selection

The input pin points, or control points, determine how the dynamically located ROIs should be moved. By default, the movement will be based on one or more template finds, but any point datums can be used. The datum reference selector, to the right of the datum in the datum view, can select an alternate input point.

Integrated Template Find

The default configuration of the NPt Locator uses integrated template finds to determine the location. When the locator step is inserted into a program, the input pin point datums are connected to the best match point output datum of the integrated template find steps. If the input pin point datum has been connected to an external point result, then the connection to the integrated template find can be restored by selecting Point Of Best Match of the find step with the Select Reference dialog box, as shown in Figure 4–13.

FIGURE 4–13. Select Reference Dialog Box



Connecting to Remote Finds

In cases where a template find is inserted elsewhere in the Job, the result from that find step can be used instead of the integrated find. This may be useful when the template find needs to be performed on an image different from the input image to the locator step. The input pin points can be connected to the Point Of Best Match output point datum of any template find step.

Note: Be mindful of the order of execution, so that the template find results are valid when the NPt Locator executes.

Connecting to Output PointDms

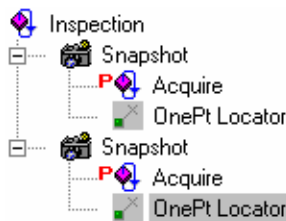
The input pin points can be connected to any step that produces an output PointDm, such as the template find, line to line point, blob filter, etc. Use the reference selector on the properties page to program the output point to use. The programmer must ensure the output point datum is valid at the time it is to be used by the NPt Locator, keeping execution order in mind.

Connecting to Another Locator

In some Jobs, NPt Locator steps may be used in more than one location. If they are both to move their controlled step's ROIs by the same adjustment amount, they can share one set of input control points.

In Figure 4–14, the first OnePt Locator will use its integrated template find for the location adjustment. Pin 1 Point of the second OnePt Locator is referenced to the Pin 1 Point of the first OnePt Locator. This will allow the second OnePt Locator to use the result of the integrated find step of the first OnePt Locator, eliminating the need to perform a second find at runtime.

FIGURE 4–14. Connecting to Another Locator



Adding and Removing Steps

The **Step List** on the NPt Locator properties page displays the steps whose ROIs are to be moved. Steps inserted directly into the NPt Locator will be added to the **Step List** automatically. To disable the dynamic location of a step's ROI, remove the step from the **Step List**.

Relocating Remote Steps

Steps that are not direct child steps of the NPt Locator can be dynamically located by adding the step to the **Step List**. Any step that has an ROI can be dynamically located; however, execution order needs to be considered. Only steps that are executed after the location should be added to the list.

Movement of Integrated Find Steps After Training

By default, the input control points of an NPt Locator are connected to the template ROI of its integrated find steps. The find step and the NPt Locator will become untrained when the find step's template ROI is moved. The step must be re-trained at the new location before it can be run. This re-training will create a new template for the find step. To re-train the NPt Locator without changing the template image of the find steps, enable the **Lock Template** parameter on the properties page of the template.

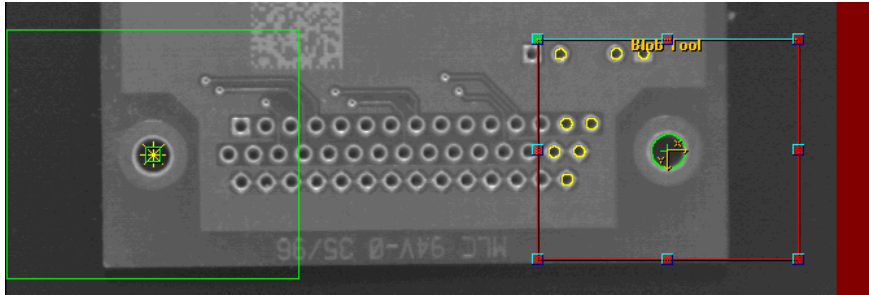
Because the find steps are integrated into the NPt Locator, the finds can only be trained by training the NPt Locator. When the NPt Locator is trained, the reference relationship between the input control points and the ROIs to be located are reestablished.

Movement of Controlled ROIs After Training

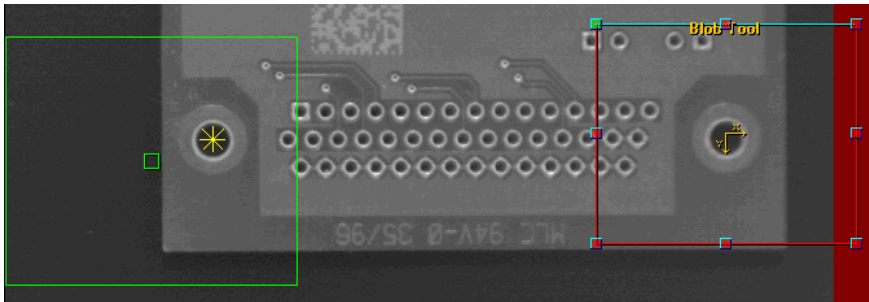
Whenever you manually move an ROI that is selected to be dynamically located, the base relationship between the locator reference location and the located shape is redefined.

Once the NPt Locator has been run, all controlled ROIs are in the adjusted positions. When a controlled ROI is moved manually, a new reference location is calculated by removing the location adjustment and storing that location as the reference.

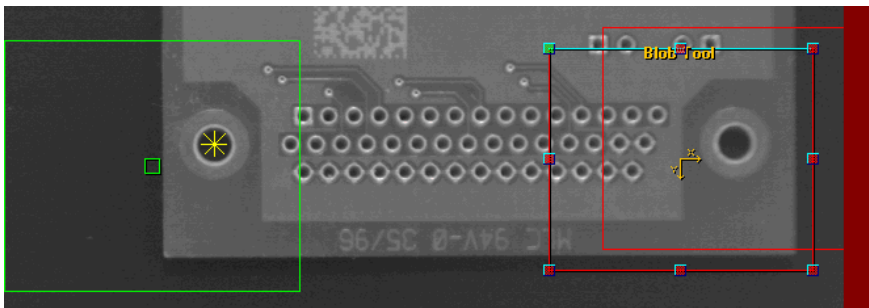
In Figure 4–15, we are performing a blob analysis on the right mounting hole of the circuit board by moving the Blob Tool ROI based on the position of the left mounting hole. A large Blob Tool ROI is centered on the right hole so that it is close to the right edge of the window.

FIGURE 4–15. Original Reference Location

When the board moves too far to the right, the Blob Tool ROI is pushed off the right edge of the buffer, causing it to fail, as shown in Figure 4–16.

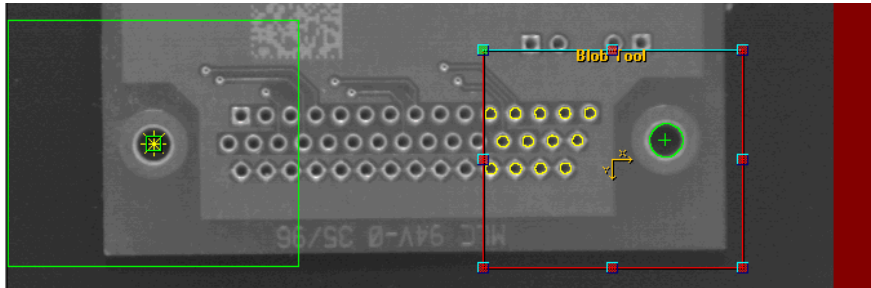
FIGURE 4–16. Run Location

Now, there is a location adjustment that is applied to the Blob Tool ROI. The ROI can be manually moved back onto the window, as shown in Figure 4–17. This will change the reference location of the Blob Tool ROI by an equal amount.

FIGURE 4–17. Adjusted ROI Position After Run

When the NPt Locator is run again on the original reference image, the reference position of the Blob Tool ROI has changed from the original position, as shown in Figure 4–18.

FIGURE 4–18. Adjusted Reference Location

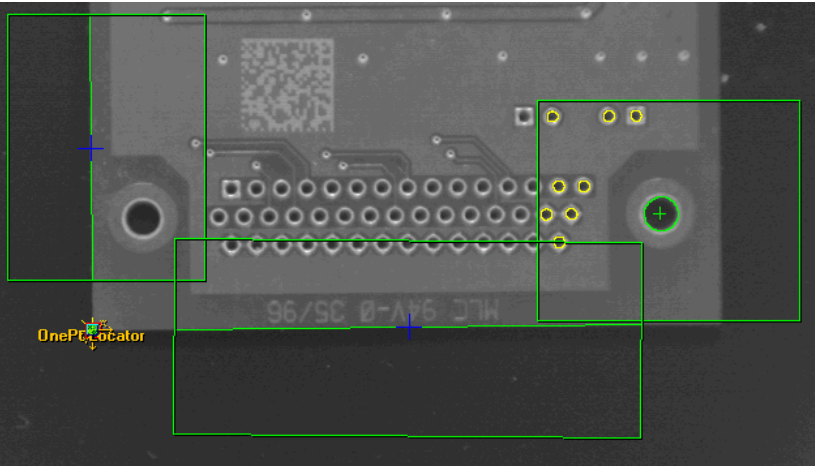


Sample Applications

Using Corner Points for Location

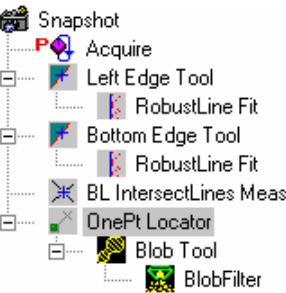
In some applications, it may not be desirable to use template finds for adjusting the location of tools, as in Figure 4–19. The ROI position is adjusted based on the corner of the board instead of a feature on the board.

FIGURE 4–19. Corner Point as Reference Location



To perform this operation, the corner point needs to be located. This can be accomplished by using two Edge Tools, each containing a RobustLine Fit to locate the left and bottom edges, as shown in Figure 4–20. This is followed by an IntersectLines Meas to find the corner point as the intersection of the edge lines.

FIGURE 4–20. Example 1



This intersection point can then be used as an input to the NPt Locator. When the NPt Locator is trained, it will establish the corner of the board as the reference location.

Connecting to OCV Autofind

In inspections involving mark inspection with an OCV Tool, it may be desirable to inspect the surface of the package around the mark for defects.

In Figure 4–21, the OCVFontless Tool inspects the mark symbols, and the runtime location of the OCVFontless Tool positions a Rect Warp around the mark. This will generate a buffer that is centered on the mark, adjusted for changes in location and rotation.

FIGURE 4–21. Connecting to OCV Autofind



By default, the OCVFontless Tool contains an integrated NPt Locator to position the OCVFontless Tool ROI, as shown in Figure 4–22.

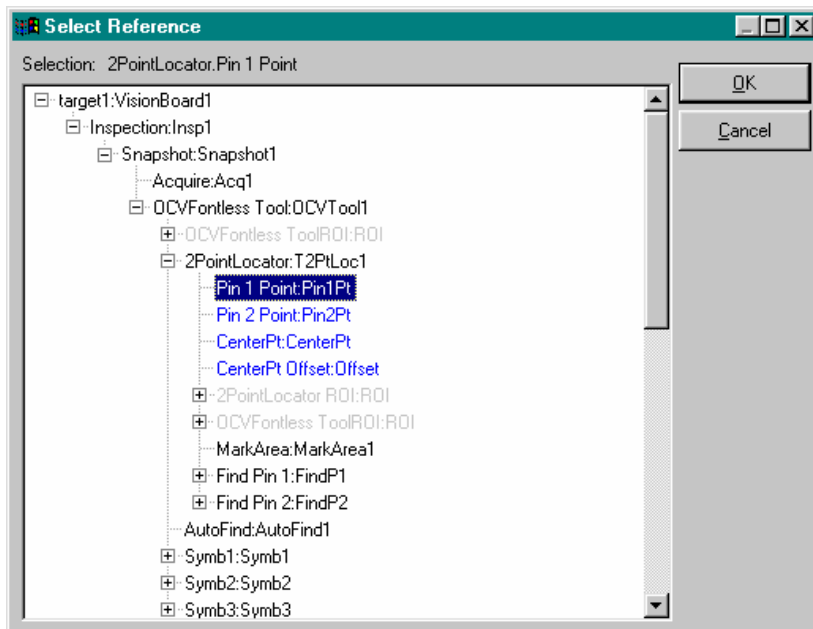
FIGURE 4–22. Integrated NPt Locator



The location result determined by the 2PointLocator of the OCVFontless Tool can be reused by the TwoPt Locator that was added after the OCVFontless Tool. This reduces overall execution time of the Job by not repeating a find that was already performed, and ensures that the positioning of the Rect Warp ROI will exactly match the orientation determined by the OCVFontless Tool.

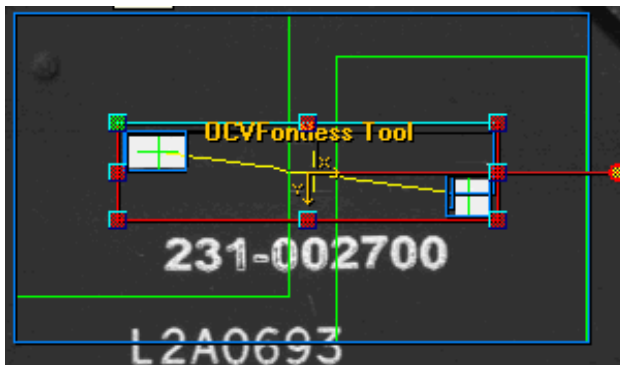
To make this connection, use the datum reference selector of the pin n point to select the Pin N Point inputs of the 2PointLocator contained within the OCVFontless Tool, as shown in Figure 4–23.

FIGURE 4-23. Select Reference Dialog Box



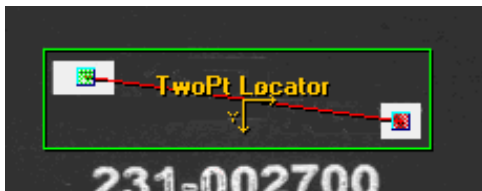
In Edit mode, position and train the OCVFontless Tool over the mark of interest. Move to the TwoPt Locator and click Train. Because it is using the results of the NPt Locator in the OCVFontless Tool, there is nothing for you to position on the image, as shown in Figure 4-24.

FIGURE 4-24. Training OCVFontless Tool



The NPt Locator will automatically move to the symbol locations determined by the OCVFontless Tool, as shown in Figure 4–25.

FIGURE 4–25. Training TwoPt Locator



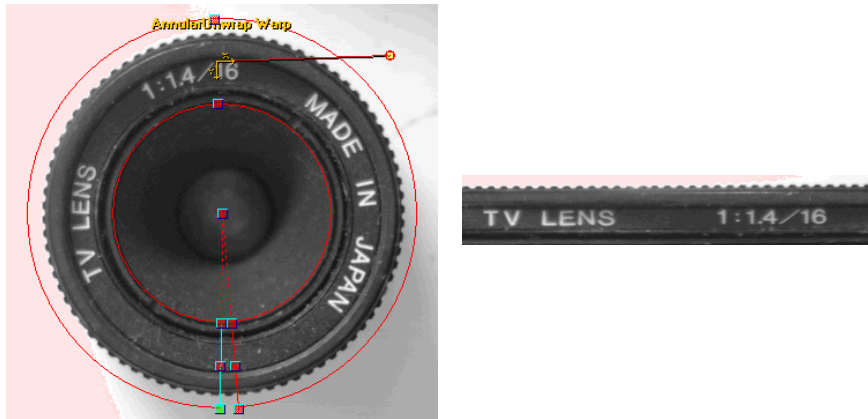
Position the Rect Warp ROI around the mark. When run, the warp ROI will be moved to match the OCV mark location, as shown in Figure 4–26.

FIGURE 4–26. Positioning Rect Warp

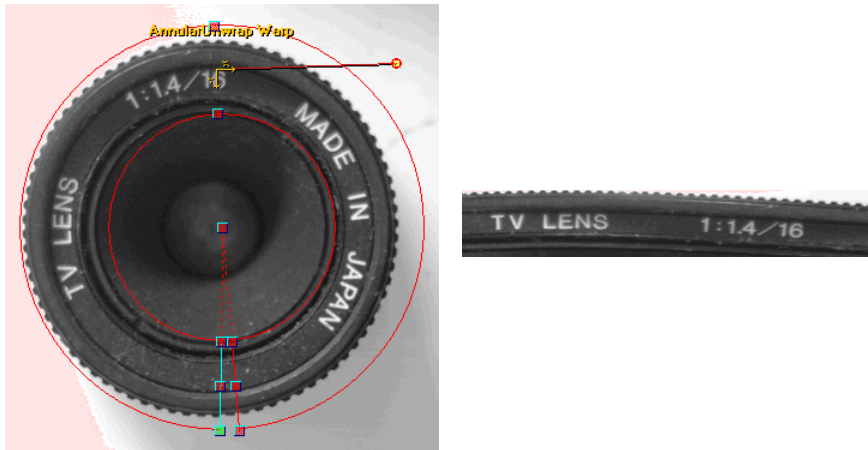


Dynamic Location of an AnnularUnwrap Warp

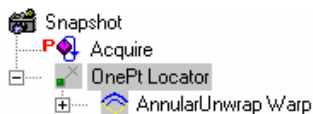
The AnnularUnwrap Warp is very sensitive to positional offsets of the image to be unwrapped. In Figure 4–27, the text around the perimeter of the lens is to be unwrapped. When the tool is correctly positioned at the center of the lens, the output is uniformly straight.

FIGURE 4-27. Aligned Annular Unwrap Warp

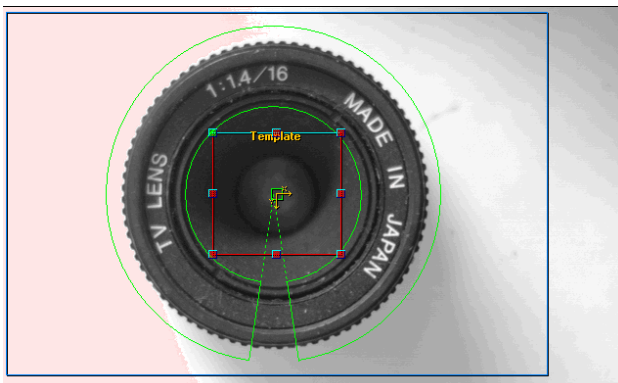
At runtime, however, a slight change in the position of the lens can cause an undesirable distortion in the output of the unwrap tool, as shown in Figure 4-28.

FIGURE 4-28. Misaligned Annular Unwrap Warp

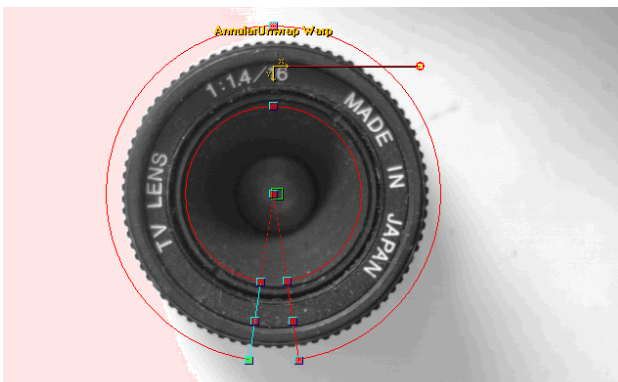
In this application, the locator can keep the AnnularUnwrap Warp tool positioned at the center of the lens. The simplest solution is to create a Job with a OnePt Locator and insert the AnnularUnwrap Warp into the locator, as shown in Figure 4-29.

FIGURE 4–29. Inserting AnnularUnwrap Warp into a Job

Position the Template ROI around the center of the lens and enlarge the search area ROI. Train the NPt Locator to establish the center portion of the lens as the template to search for at runtime, as shown in Figure 4–30.

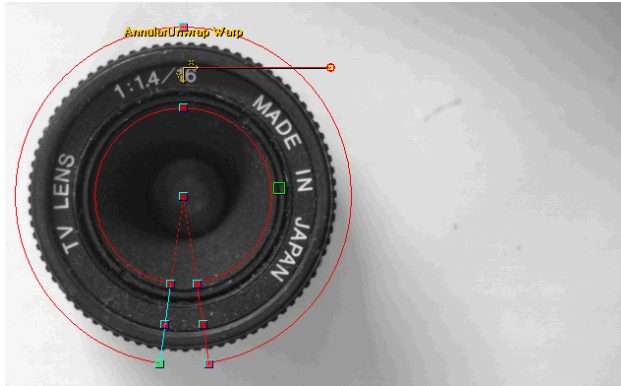
FIGURE 4–30. Positioning of Template Find ROI

Position the base of the AnnularUnwrap Warp at the center of the lens and adjust the inner and outer radii to enclose the text area of the lens surface, as shown in Figure 4–31.

FIGURE 4–31. Positioning of AnnularUnwrap Warp ROI

When the Job is run and the lens is in a different location, the center of the lens will be found by the NPt Locator and will move the AnnularUnwrap Warp to keep it properly centered, as shown in Figure 4–32.

FIGURE 4–32. AnnularUnwrap Warp ROI Relocated at Runtime

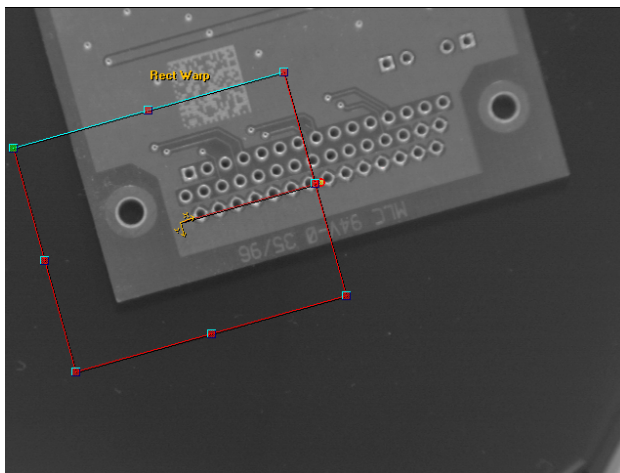


Reference Frames

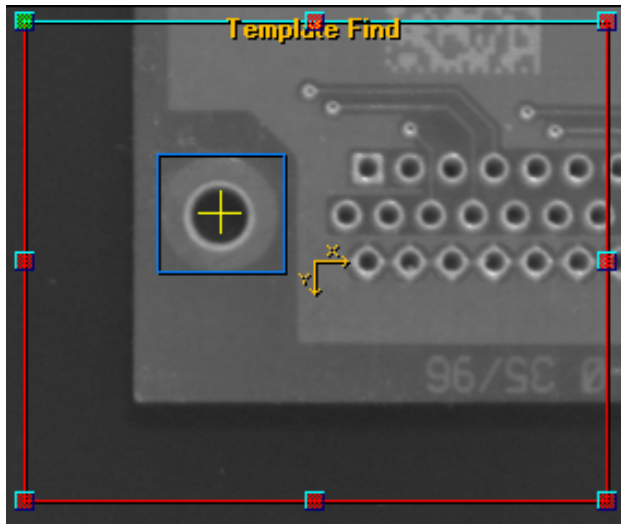
Reference Frames of Points

Every point datum has an x,y angle and scale component that specify a location, orientation and size. However, that location is specific to one coordinate system or reference frame. Some vision steps process an area of an image and produce an output image. This new image buffer will have a reference frame whose coordinate system is defined by the position of the step's ROI in the input image. In Figure 4–33, the ROI of the Rect Warp defines a new reference frame to be associated with its output image. Figure 4–34 shows the resulting output image of the RectWarp. The image is rotated to 0° and shifted to 0,0 in the upper left corner.

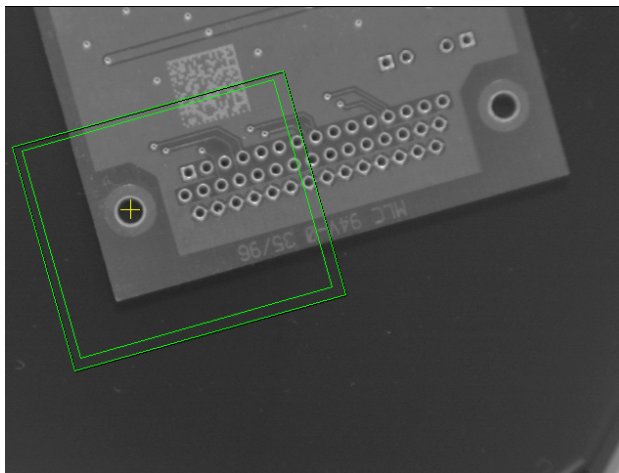
FIGURE 4–33. Rect Warp ROI Defines New Reference Frame



A point in one reference frame can be converted to a point in another reference frame with a coordinate transform. In Figure 4–34, a Template Find is run in the warp output image, which results in a match point at pixel location 99,94.

FIGURE 4–34. Point Location in RectWarp Reference Frame

That point corresponds to pixel location 129,207 in the original image, as shown in Figure 4–35.

FIGURE 4–35. Snapshot Reference Frame

The point only defines one point in space, but will have different pixel locations depending on the reference frame being used, i.e., the RectWarp image buffer of Figure 4–34 or the original Snapshot image buffer of Figure 4–35.

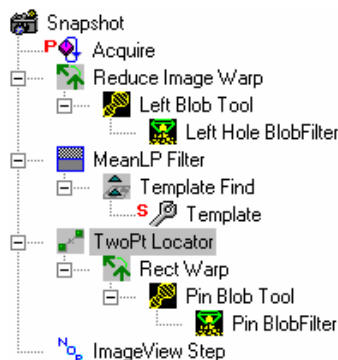
Reference Frames of the Locator

Since the NPt Locator does not produce any output image, the NPt Locator will inherit the coordinate reference frame of its parent step. When the NPt Locator is inserted into a Snapshot, the integrated template finds will also be in the Snapshot step's reference frame, and so will the output's Center Point and Center Point Offset.

However, the NPt Locator's input control points can be connected to points from other reference frames, which will affect the reference frame used by the NPt Locator. The reference frames of the center point and center offset point output datums are directly related to the reference frames of the input control points. In the case of a one point locator, the center point reference frame will always be the same as the input control point's reference frame. For a two or three point locator, the center point's reference frame will be the coordinate reference frame that is common to all inputs.

Consider the example application in Figure 4–36.

FIGURE 4–36. Example 2

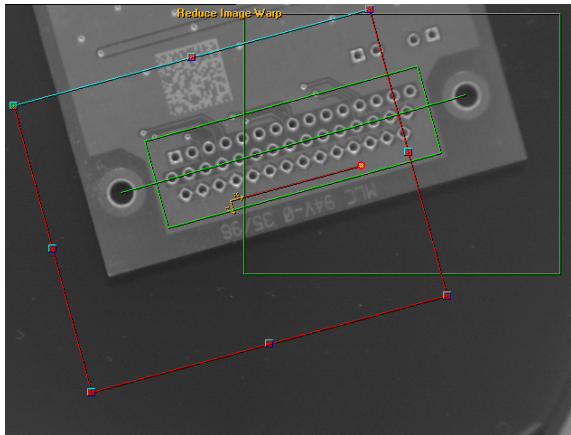


Reduce Image Warp is a RectWarp that shrinks a portion of the image by 50%. The Blob Tool is run on that image. Another area of the image is filtered by the MeanLP Filter and a Template Find is performed on that image. The results of Blob Filter and Template Find are used by the TwoPt Locator to position a Rect Warp, and another Blob Tool is run on that image.

This Job has four distinct reference frames, one associated with each of the output images from the steps: Snapshot, Reduce Image Warp, MeanLP Filter, and Rect Warp.

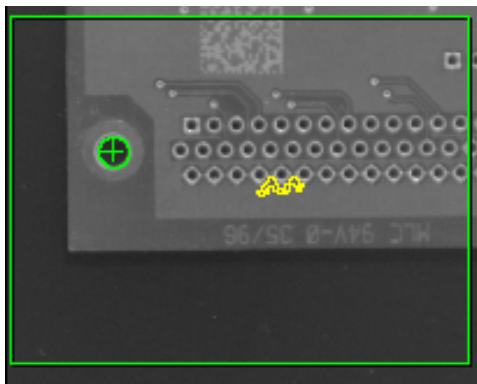
The Reduce Image Warp ROI is placed over an area of the image that will contain the large left hole in the circuit board, as shown in Figure 4–37.

FIGURE 4–37. Positioning of Reduce Image Warp ROI



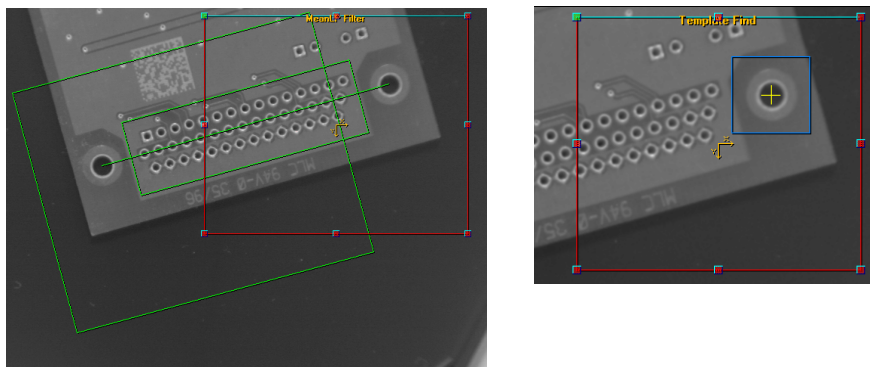
When run, the Blob Filter will locate the center of the hole in the reduced image, as shown in Figure 4–38.

FIGURE 4–38. Blob Tool Results in Reduce Image Warp Buffer



Similarly, the template find will locate a match point in the MeanLP Filter output image. Figure 4–39 shows the positioning of a MeanLP Filter in the Snapshot Buffer and Template Find ROIs in the MeanLP Filter Image Buffer.

FIGURE 4–39. Positioning of MeanLP Filter and Template Find ROIs



The Blob Filter provides the hole center result as pixel location 45,62 in the reduced image. This transforms to pixel location 129,208 in the reference frame of the snapshot. The template find result is 245, 90 in the MeanLP reference frame, and transforms to pixel location 509,100 in the snapshot reference frame. These reference frames need to be considered when looking at the results shown in the status bars. The output results of the blob filter and template find steps will be in their respective reference frames.

FIGURE 4–40. Status Bar Results

```
Inputs: Select Part (among all parts) with Largest Area. Min/Max: 0.00/307200.00
Outputs: Center:[45.1, 61.9] Area:135.00 Holes:0 Roundness:1.000
```

Blob Tool Status Bar

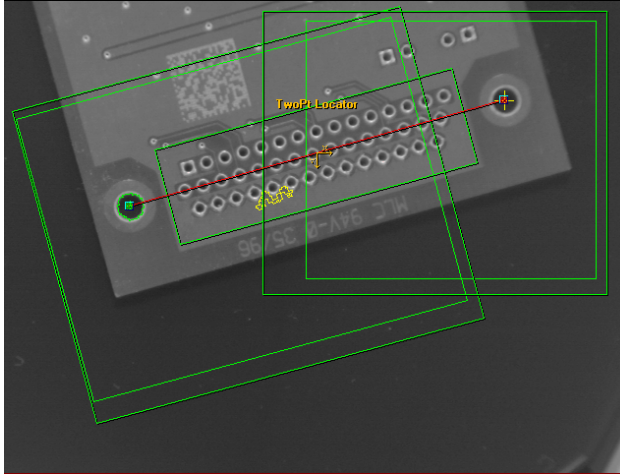
```
Inputs: Robustness: Least; AcceptThr: 0.70 CandThr: 0.70 NumCandidates: 1
Outputs: MPt: X=245.50 Y=90.50 Score=1.00 NumMatches: 1
```

Template Find Status Bar

However, the input coordinates to the NPt Locator will be displayed in the reference frame of the NPt Locator. Even though they are the same points that were generated by the Blob Filter and Template Find, the pixel coordinates will be different from the perspective of the NPt Locator.

FIGURE 4-41. Status Bar — NPt Locator

Inputs: Dist Tol: 5 TPt1: X=128.853 Y=208.356 TPt2: X=509.5 Y=100.5
Outputs: Centroid: X=319.176 Y=154.428 Angle=0 MPt1: X=128.853 Y=208.356 MPt2: X=509.5 Y=100.5



The two output datums Center Point and Center Point Offset must also be associated with a single reference frame. In the case of a OnePt Locator, the reference frame of these outputs will be the same reference frame as that of the input point. In the case of a TwoPt or ThreePt locator, the common reference frame of the inputs is used.

In Figure 4-36, “Example 2,” on page 4-26, one point is in the Reduce Image Warp reference frame and the other is in the MeanLP Filter reference frame. Each of these reference frames is derived from the Snapshot reference frame, so the snapshot is a common parent reference frame to both inputs. Therefore, the center point and center point offset outputs will be in terms of the snapshot’s coordinate system.

Results

The NPt Locator will produce two result points: the center point and offset distance. When the integrated template find steps are used, additional results, such as correlation score, can be found in the template find result list.

CenterPt

The input pin points are combined to a single point, the CenterPt. This point represents the overall position of the object being used as a location reference:

- OnePt Locator — This will be equal to the Pin 1 Point input connection point.
- TwoPt Locator — This will be the midpoint of a line connecting the two input points.
- ThreePt Locator — This will be the average center of the three input points.

Note: The training position of the reference point is available as a result as the Trained Position PointDm.

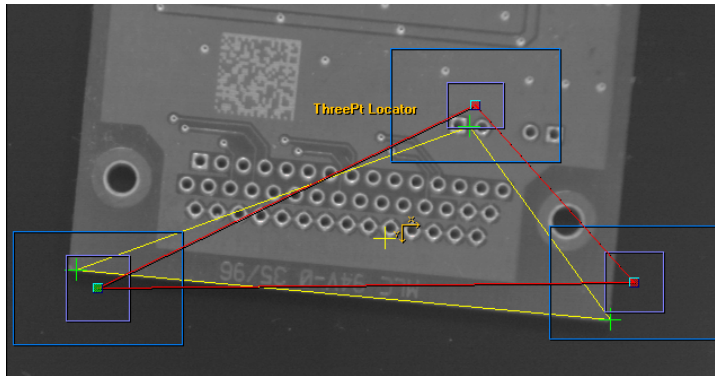
Offset Distance

The location adjustment calculated from at each run is output as an Offset Distance. This contains the distance of the reference point between its training and runtime positions.

Edit Window

In the Edit Window, a control point will be shown for each input pin point. You can drag these points, which are shown as red squares, to set the template location of the inputs when using the integrated template finds. When the integrated template finds are used, each control point will be accompanied by a template ROI and search area ROI, shown by the blue rectangles. For a TwoPt or ThreePt Locator, a red line is drawn to connect the input points, as shown in Figure 4–42.

FIGURE 4–42. ThreePt Locator



When the step is run and the **Graphics Level** is set to Show Basic Graphics or higher, additional locator graphics are drawn. Each input pin point found by the template find is marked with a green cross. For the TwoPt and ThreePt Locators, yellow lines are drawn connecting the input points. A yellow cross marks the center point result.

In Runtime, the setup positions are not shown, only the current run positions.

Status Bar

After a run, the input Status Bar will show the user-specified parameters, as shown in Figure 4–43.

FIGURE 4–43. Status Bar

Inputs: Dist Tol: 5 Rigidity: 5 TPt1: X=80 Y=287 TPt2: X=415 Y=125 TPt3: X=556 Y=282
Outputs: Centroid: X=335.464 Y=243.395 Angle=5.72705 MPt1: X=61.0608 Y=271.476 MPt2: X=410.151 Y=144.159 MPt3: X=535.181 Y=314.

The Status Bar parameter descriptions are:

Inputs: Dist Tol: d Rigidity: r TPt1: X=x1 Y=y1 TPt2: X=x2 Y=y2 TPt3: X=x3 Y=y3

Where: d — The distance tolerance.

r — The rigidity.

x1,y1 x2,y2 and x3,y3 — The reference locations of each input pin point you set at training time.

After a successful run, the output status bar will show the results in the format:

Outputs: Centroid: X=x0 Y=y0 Angle=a0 MPt1: X=x1 Y=y1 MPt2: X=x2 Y=y2 MPt3: X=x3 Y=y3

Where: x0,y0 — The position of the center point.

a0 — The amount of rotation from the training position.

x1,y1 x2,y2 and x3,y3 — The resulting input point from each input pin connection is reflected in these results. These are the template find results when the locator is using the integrated find steps.

If the step fails to run, one of the following messages may be displayed in the output status bar.

- Outputs: Failed Condition: Out of Rigidity
- Outputs: Failed Condition: Out of Tolerance
- Outputs: Failed Condition: Pins not found

Reference

NPt Locator Tool

This tool locates the position of an object in the image dynamically and repositions the inspection tools to where the object is found. In many vision applications, the part being inspected at runtime is not always perfectly fixed in the camera FOV, and can move from image to image. To inspect these moving parts, you need to use NPt Locator. NPt Locator finds features on the object using Find, or can be connected to positions results, i.e., points generated by other tools that execute before it in the cycle.

Other Steps Used

Template Find — Template Find locates a template image within a given ROI. An integrated Template Find is created for each pin when an NPt Locator is inserted into a Job. For more information, refer to “Template Find” on page 7-84.

Theory of Operation

NPt Locator calculates part rotation and translation offsets from its recorded position at training time. These calculated offsets are applied to any tool or step input ROI that are direct components of the NPt Locator. NPt Locator finds the position of the part by locating any number of pins. A pin is any step or tool that generates a Point datum as one of its output. By default, the NPt Locator creates one Template step per pin. Template uses template matching to locate the pin positions.

The user interface supports the creation of OnePt, TwoPt, or ThreePt Locator. Typically, NPt Locator is used in conjunction with any tool to adjust the position of the input ROI in x and y. They are used in conjunction with the Rect Warp so that the input ROI of Rect Warp can be adjusted dynamically to match the part rotation and translation.

Number Of Features controls which part movement the tool can compensate for:

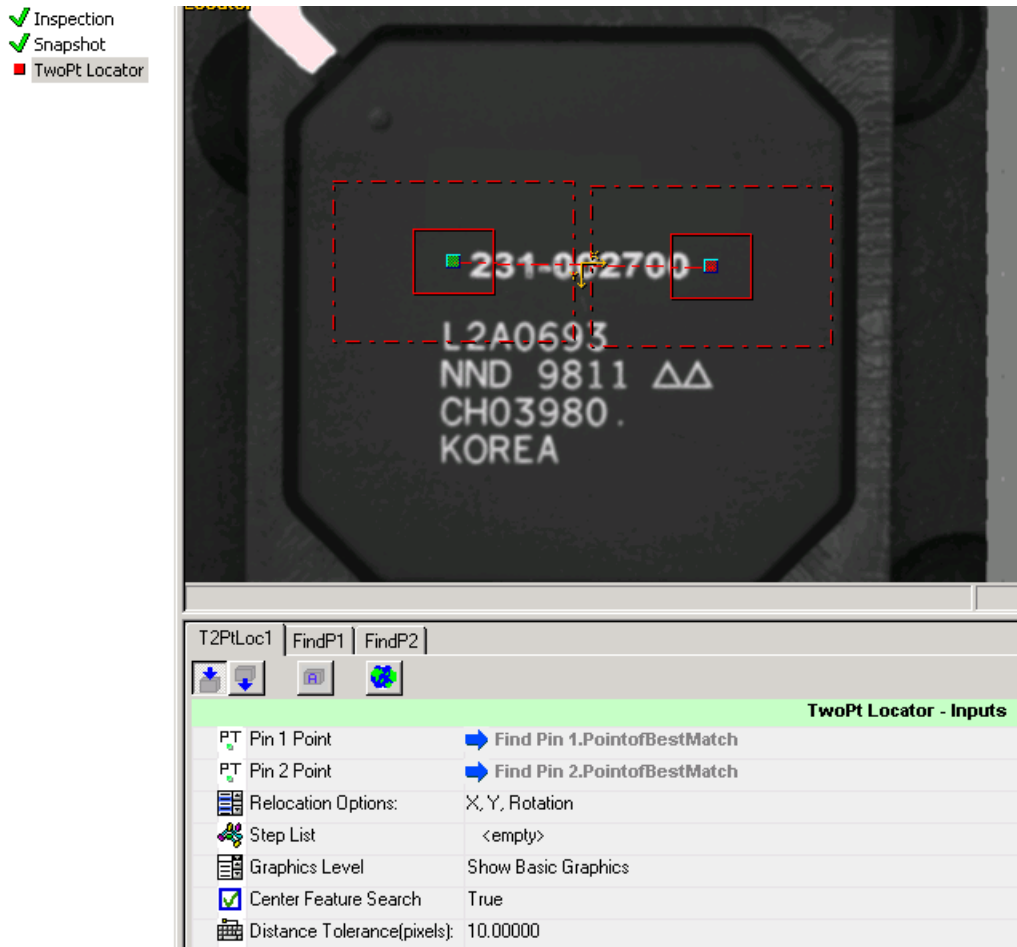
- **1Pt Locator** — A single feature finds the part translation x and y offset. If the point feature contains angle and/or scale information, then the 1PtLocator tool can also correct for part rotation and part scale changes from image to image.

- **2Pt Locator** — Two features are used. Parts rotated by up to 360° can be detected but not scale change.
- **3Pt Locator** — Three features are used. The part can be detected even if rotated by up to 360° . In addition, some amount of distortion can be tolerated. This is the most accurate method. This method does not support scale change.

Note: 2Pt Locator and 3Pt Locator tools use the distance between pins (feature points) in order to robustly find the part translation and rotation at runtime in the presence of multiple or noisy features and, as a result, do not correct for scale change.

A special ROI called a PolyShape provides the user interface for positioning the pins, as well as sizing their templates and search areas when the NPt Locator is programmed to use its internal feature location method. In Figure 4–44, the pins can be individually positioned by moving their center point with the mouse. The inner/outer rectangle defines the template/search ROI, respectively, of each pin. By default, these are always centered around the pin unless **Center Feature Templates** is checked.

FIGURE 4-44. TwoPt Locator

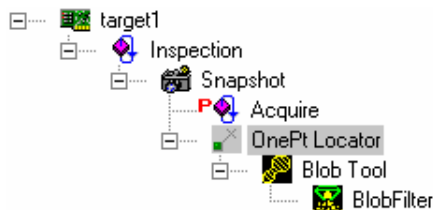


Sample Application

This example shows how to set up a locator to reposition a Blob Tool ROI as the object being inspected moves. This application will use a Blob Tool to inspect holes on a circuit board for verifying the number of holes and minimum hole size.

1. Open FrontRunner. Click **Show Job Tree Display** and create an inspection by inserting a OnePt Locator into the Snapshot.

2. Insert a Blob Tool into the OnePt Locator to construct the Job shown in Figure 4–45.

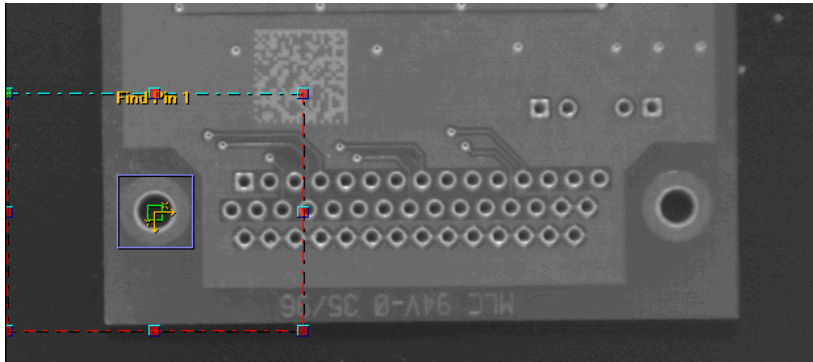
FIGURE 4–45. NPt Locator Sample Job

3. Select **Acquire** on the Job Tree to display the Acquire properties page, as shown in Figure 4–46.

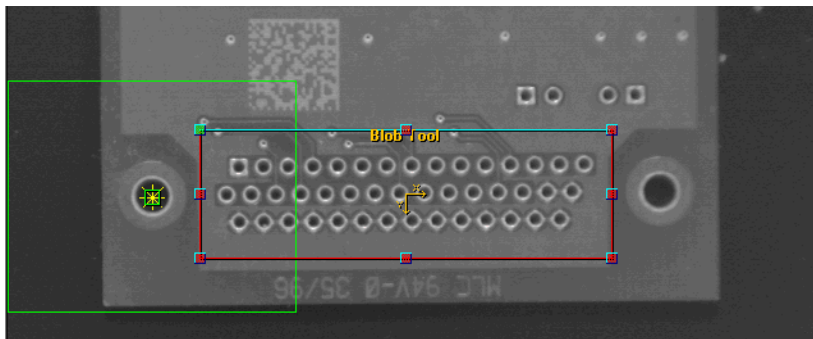
FIGURE 4–46. Acquire Properties Page

Acquire - Inputs	
Picture Mode	Load Images from File
File List	c:\vscape\tutorials & samples\sample jobs\npt locator\pcb01.tif

4. In the Picture Mode pull-down menu, select Load Images from File (Figure 4–46).
5. To File List, add pcb01.tif (Figure 4–46).
6. Select Blob Tool to display the Blob Tool properties page.
7. Enable Ignore blobs that touch the ROI.
8. Disable Discard children blobs. Leave all other properties at the default values.
9. Click Show Setup Display to position the ROIs and train the tools.
10. Select the OnePt Locator and position the Template ROI around the large left hole of the circuit board, and enlarge the Find Pin 1 ROI to cover a large area around the same hole, as shown in Figure 4–47.

FIGURE 4-47. Positioning Find Pin 1 ROI

11. Once the ROIs have been positioned, click **Train**. This will train the template find by defining the template pattern for the Template Find. This will also train the locator, by establishing the center of the template as the reference location. At runtime, the template pattern will be located in the Find Pin 1 search area, and that location compared to the reference location to determine how the child steps are moved.
12. Position the Blob Tool ROI around the smaller set of holes in the circuit board. Since the reference location of the OnePt Locator has already been defined, adjusting the position of the Blob Tool ROI establishes the relationship between the Blob Tool ROI and the reference location, as shown in Figure 4-48.

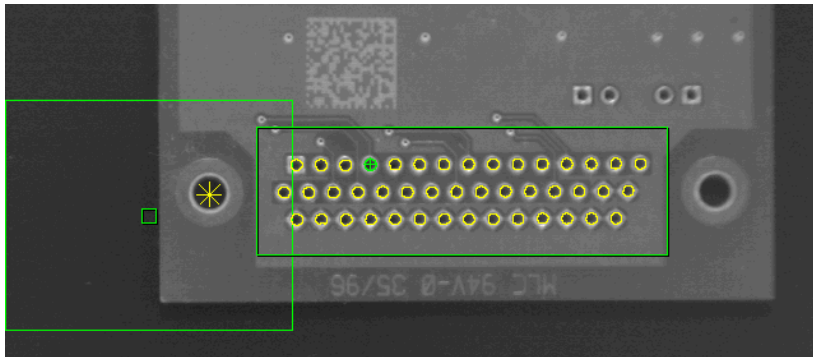
FIGURE 4-48. Positioning Blob Tool ROI

13. Additional image files can be added by returning to the Acquire properties page and adding the image files pcb02.tif and pcb03.tif to the File List.

Note: PCB02.tif and PCB03.tif are located in:
c:\vscape\tutorials & samples\sample jobs\npt locator\

14. Click **Run Inspection Once** or **Start Inspection** to run the inspection on a different image. The NPt Locator will find the large hole that was trained as a template, and move the Blob Tool ROI by the same amount the template has moved, as shown in Figure 4–49. This will keep the Blob Tool ROI located around the holes of interest.

FIGURE 4–49. Moving Blob Tool and Template ROIs

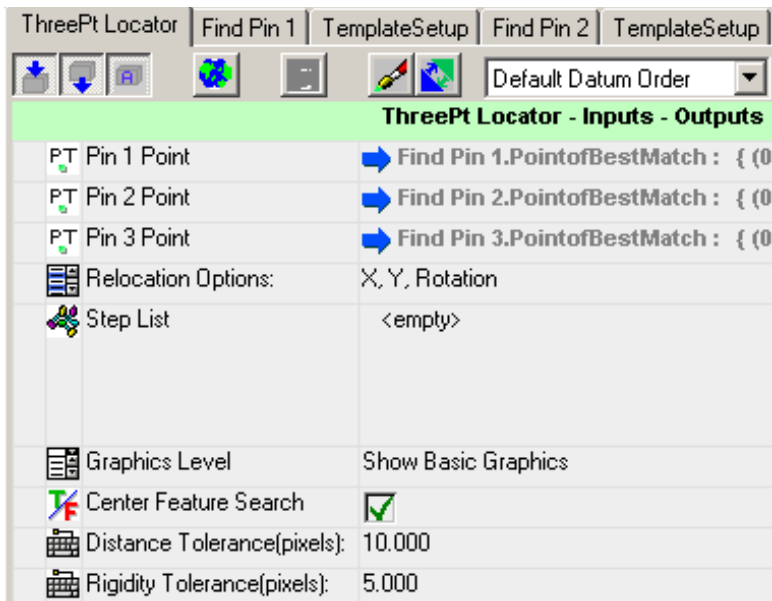


Description

NPt Locator allows editing through the NPt Locator properties pages, as shown in Figure 4–50.

Additional tabs for Find properties of each pin and the steps that are dynamically located are also shown.

FIGURE 4–50. ThreePt Locator Properties Page



Settings

- **Pin1 Point** — Identifies the Pin 1 point. Determines which input points are to be used for locating the object in the FOV.
- **Pin2 Point** — Identifies the Pin 2 point. Determines which input points are to be used for locating the object in the FOV.
- **Pin3 Point** — Identifies the Pin 3 point. Determines which input points are to be used for locating the object in the FOV.
- **Relocation Options** — Controls the type of relocation performed on the list of step's ROIs to be moved. Each degree of freedom can be enabled/disabled individually. For example, if only X is selected, then the ROIs will be moved only by the x component of the offset between the train position and the run positions of the Locator features:
 - X — Applies the X offset of the transform only.
 - Y — Applies the Y Offset of the transform only.
 - Rotation — Applies the Angle component of the transform only.
 - Scale — Applies the Scale component of the transform only.

Note: A relocation transform is defined by an offset (x, y, theta) in the System of coordinates of the Locator features, NOT necessarily the system of coordinates of the ROIs to relocate. In that case, once the offset is calculated in the Locator frame, the component (X, Y, Theta) are individually converted to each ROI's frame before being applied.

By default X, Y, Rotation offsets are enabled for the 2Pt Locator and 3Pt Locator tools, scale is not available.

For a 1Pt Locator tool, X,Y are enabled by default. Rotation and scale correction can be enabled for locator features that support it. This includes BlobFilter centroids.

- **Step List** — Contains a list of steps whose ROI is to be moved by the locator.
- **Graphics Level** — Controls the amount of graphics to be drawn on the image at runtime:
 - Show None — No runtime graphics.
 - Show ROI Only — No runtime graphics.
 - Show Basic Graphics — Draws the center point and lines connecting input points.
 - Show Details — Same as Show Basic Graphics.
 - Show Details And Mask — Same as Show Basic Graphics. Masking is not allowed for this tool.
- **Center Feature Search** — When enabled, the template find search areas will be centered around the same control point as the associated template ROI. This has no effect if the integrated template find steps are not used.
- **Distance Tolerance(pixels)** — Defines how far apart in pixels corresponding pins can deviate from one another. For example, at training time, the recorded distance between the two features of a 2Pt Locator step is 100 pixels. Then, with a distance tolerance of 10.0 (default) at runtime, the tool will pass if the features are between 90 and 110 pixels apart. This property is only available for 2Pt Locator and 3Pt Locator. This can be any positive real number 0.0 +.

This property is useful in helping the system decide between sets of pins when there may be many more than one correlation match. The system will use this to pick the pair or triplet of pins that best matches the distance tolerance. To fully enable this feature, set the **Enable Multiple Match** option of the Find Pin Step to a number greater than 1.

Default: 10.0

- **Rigidity Tolerance(pixels)** — Only valid for a ThreePt Locator and specifies the maximum rigidity allowed. The rigidity is defined as the average change in distance of the pin points from the center point between training time and runtime. This can be any positive real number 0.0+.

Default: 5.0

Training

When NPt Locator is trained, the position of all the pins is recorded. In addition, the direct child steps are scanned and their ROI positions are recorded.

Results

- **Status** — A pass/fail status. The component steps of NPt Locator are executed only when the tool passes (Pass/Fail status is 1).
- **CenterPt** — The measured, run time position of the centroid.
- **Offset Distance** — Distance between the trained and measured positions.
- **Trained Position** — Position of the centroid at training.

I/O Summary

NPt Locator provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

For 1 and 2 Pts

Inputs: Dist Tol: xxx TPt1: X=xxx Y=xxx TPt2: X=xxx Y=xxx

Outputs: Centroid: X=xxx Y=xxx Angle=aaa MPt1: X=xxx Y=xxx
MPt2: X=xxx Y=xxx

Where: xxx = coordinate of point in the image in pixels, either X or Y.
aaa = angle in degree of center of all the pins.

Note: X and Y coordinates are displayed for each pin, i.e., either 1 or 2 pins based on the variant NPinFind inserted.

For 3+ Pts

Inputs: Dist Tol: xxx Rigidity: xxx TPt1: X=xxx Y=xxx TPt2: X=xxx Y=xxx
TPt3: X=xxx Y=xxx

Outputs: Centroid: X=xxx Y=xxx Angle=aaa MPt1: X=xxx Y=xxx
MPt2: X=xxx Y=xxx MPt3: X=xxx Y=xxx

Where: xxx = coordinate of point in the image in pixels, either X or Y.
aaa = angle in degree of center of all the pins.

Note: X and Y coordinates are displayed for each pin, i.e., either 1, 2 or 3 pins based on the variant NPt Locator inserted.

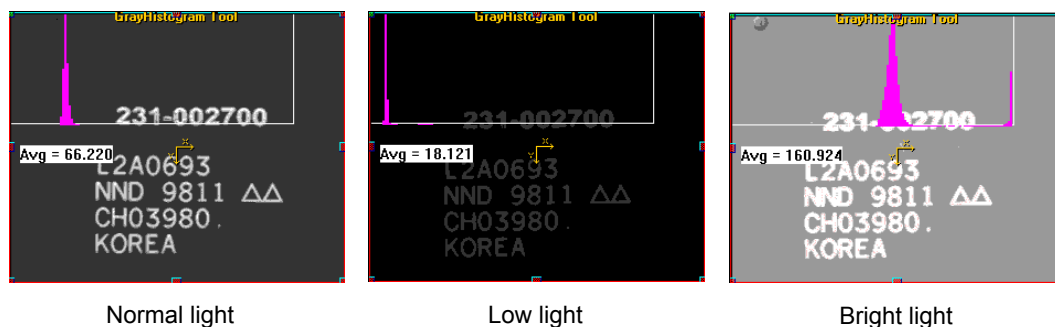
CHAPTER 5

Dynamic Thresholding

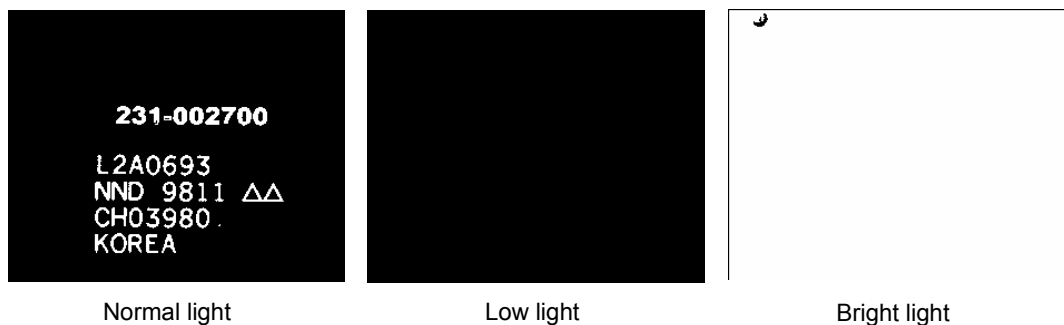
During runtime, the lighting conditions or the reflectivity of parts can vary from one image to the next. When this happens, and the threshold value remains constant, some pixels may be incorrectly categorized as foreground pixels when they should be background pixels, or vice versa.

The example images shown in Figure 5–1 are the same part taken at three different light levels. The histogram of the gray values and the average gray value are shown on each image.

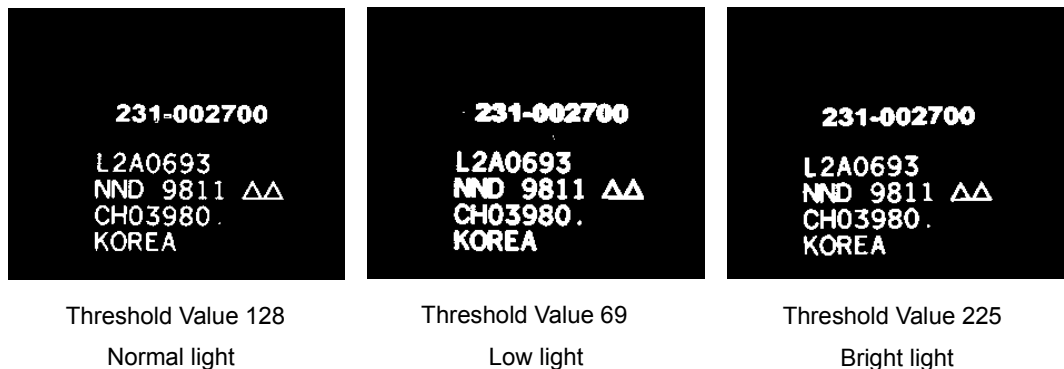
FIGURE 5–1. Example Images



The goal with the images in Figure 5–1 is to threshold them, to separate the printed characters from the background. Choosing a threshold of 128 will work well for the first image but will not produce desirable results for the other images, as shown in Figure 5–2. Because the brightness varies over a wide range, no single threshold value will work for all the images.

FIGURE 5-2. Constant Threshold Value 128

To compensate for these intensity changes at runtime, the DynamThresh Tool is used. A DynamThresh Tool adjusts the threshold values of connected steps at runtime.

FIGURE 5-3. Dynamic Threshold Values

Average Gray Value

The DynamThresh Tool will compute the average gray value of the pixels within an ROI. Then, it will adjust the thresholds of other steps based on this value. This will vary the thresholds according to changes in overall brightness of the image.

Threshold Reference Value

When deciding how much of an adjustment to make for the thresholds, the base level needs to be set. This reference is set during the setup of the tool, and remains unchanged as the tool runs. It is normally equal to the average gray value of the reference image; however, you can modify it.

Threshold Adjustment

The Threshold Adjustment is the difference between the average gray value, which changes with each new image at runtime, and the reference value. This adjustment value, also called the offset, is applied to the thresholds of connected steps.

Building a Job

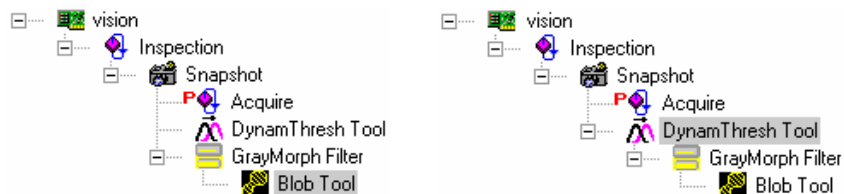
The DynamThresh Tool can be inserted into any other step that provides an image output or another tool.

The step that the DynamThresh Tool is inserted into will determine which image buffer is used for the computation of the average gray value.

The DynamThresh needs to be inserted into the Job Tree so that it gets executed before any of the steps whose thresholds it is to control. A normal Job would have controlled steps inserted as child steps of the DynamThresh Tool, which guarantees the correct execution order. If remote steps are to be controlled, you need to keep execution order in mind.

The examples in Figure 5–4 show where the execution order is to be considered.

FIGURE 5–4. DynamThresh Job Examples



The two program constructions are equivalent. The threshold of a Blob Tool inside a GrayMorph Filter is dynamically adjusted, but the unmorphed image controls the adjustment. The DynamThresh Tool needs to be inserted into the Snapshot and come before the GrayMorph Filter.

Inserting Steps With Thresholds

Any step that can normally be inserted into a tool can be inserted into a DynamThresh Tool. However, only steps that contain thresholds, such as Blob Tool and Flaw Tool, can be adjusted by the tool at runtime. This tool will not generate an output buffer. Any steps inserted into the DynamThresh Tool will be connected to the same input buffer used by the DynamThresh Tool.

Base Threshold Values

When a tool is inserted into the Job Tree, it has default threshold values. When the DynamThresh Tool is trained, the threshold values of the inserted tools are registered as the base threshold value for this tool. The adjustment value calculated by the DynamThresh Tool is applied to this base threshold value when the tool runs.

Base threshold values are also stored when a step is added to the control **Step List** and are updated whenever you change the threshold value.

For example, if the High Threshold of a Blob Tool is set to 80 during setup, this is its base threshold value. Any offset determined by the DynamThresh Tool is added to this base threshold value to determine the runtime threshold value. A runtime offset of 15 will equate to a runtime threshold of 95 for this Blob Tool.

AutoThreshold Selecting Initial Threshold Values

Some tools, such as the Blob Tool, provide a **Use Autothreshold** parameter. When a tool with this parameter is added to a DynamThresh Tool, this parameter is disabled. When left enabled, the Autothreshold overrides the DynamThresh Tool's control of the thresholds.

When setting up a new Job, it may not be desirable to leave the base threshold value at the default of 128, or to figure out a good threshold value manually. In this case, the Autothreshold option can be used. Enable the **Use Autothreshold** parameter and click **Run Inspection Once**. This runs the Blob Tool using the Autothreshold capability of the Blob Tool, and sets the threshold values appropriately for the current image. Once good threshold values are set, disable **Use Autothreshold** to return control of the threshold to the DynamThresh Tool at Runtime.

Setting Up a Job

Before training the DynamThresh Tool, you need to set up the ROI of the tool.

Dynamic Threshold ROI

Position the DynamThresh Tool ROI over the area of the image where the average gray value is to be calculated. This can cover the entire image for an average light intensity of the camera image. It can also cover a selected region, as shown in Figure 5-5, where the chip package is the main concern and not other background objects that may affect the result.

FIGURE 5-5. ROI Positioning



Controlled Step's ROIs

There are no additional requirements on the placement of the controlled step's ROIs. There does not need to be any correspondence between the DynThresh Tool's ROI and any of the controlled steps. Refer to the individual step's documentation for additional information on the setup of those steps.

Normal Training of Reference Value

Before the tool can be run, it must be trained. After positioning the ROI, click Train. The average gray value of the pixels in the ROI will be calculated and that value stored as the reference value. This reference value can be viewed on the DynamThresh Tool properties page.

Manually Setting Reference Value

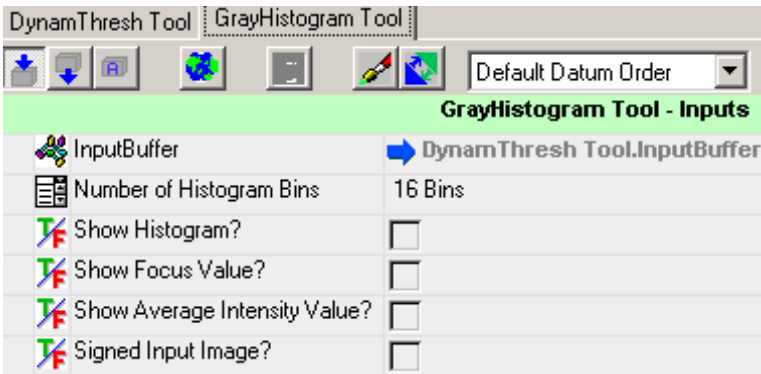
After training, the reference value shown on the properties page can be manually adjusted, if desired. The tool can be re-trained at any time.

Running a Job

Integrated GrayHistogram Tool

Each DynamThresh Tool has an integrated GrayHistogram Tool tab, as shown in Figure 5–6. When the DynamThresh Tool executes, it will first execute the GrayHistogram Tool. This tool will calculate the average gray value of the pixels in the ROI and pass the result to the DynamThresh Tool.

FIGURE 5–6. GrayHistogram Tool Properties Page



You can modify the properties of the GrayHistogram Tool, but it should not be necessary.

Note: If **Input Control Value** of the DynamThresh Tool is not left at its default connection to the GrayHistogram Tool, this tool will not execute. This reduces runtime by avoiding the computation of a result that is not used by the DynamThresh Tool, so other steps should not use the result of the GrayHistogram Tool either.

Calculation of the Threshold Adjustment

The DynamThresh Tool retrieves the control value from the datum determined by its Input Control Value connection. By default, this is the average gray value computed by the GrayHistogram Tool. The reference gray value is then subtracted from this control value to compute the Threshold Adjustment.

Modification of Threshold Values

The new Threshold Adjustment value is added to the base threshold value of each controlled step to determine its new threshold value. This new threshold value is then pushed into the threshold datum of the controlled step; the step will then use this value when it executes. Because it is pushed into the controlled step, the threshold is directly available to the step when it executes. When viewing the controlled step's properties page, the value shown in the threshold box is always the adjusted value.

Exception for Thresholds 0 and 255

Some steps have two threshold levels, a High Threshold and Low Threshold. Often, either the low threshold is set to 0 or the high threshold is set to 255. When the base value of a threshold has been set to one of these two values, the DynamThresh Tool does not adjust it at runtime. For example, if a blob tool has a low threshold set at 0 and a high threshold set to 100, then when the DynamThresh tool applies a threshold adjust value of 5 to the blob tool, the high threshold is set to 105, but the low threshold will remain at 0.

Clamping

Saturation

The thresholds of controlled steps will not be set outside the range of 0 to 255. When the addition of the Threshold Adjustment to a base threshold value would cause the threshold to be set outside this range, the threshold is set to the closest limit. For example, if a controlled threshold's base value is 200 and the DynamThresh Tool calculates a Threshold Adjustment value of 60, the controlled threshold is set at 255, not 260.

If any controlled threshold is limited in this way, the Threshold Clamped output status is set to TRUE. This can notify the operator that the threshold values being used by a step are not necessarily in a valid range. This can happen in cases when the image becomes completely black due to a lighting failure, the part is not in the FOV, etc. The Threshold Clamped output can be used for adjusting the program execution flow at runtime.

Clamping does not cause an execution failure, but you can add this datum to the inspection pass criteria.

0 and 255 Gray Values

Since the DynamThresh Tool does not adjust thresholds with a base value of 0 or 255, they cannot be considered clamped, regardless of the threshold adjustment value.

Advanced Configurations

Reconnecting Input Reference

Connecting to Double or Int Datum Outputs

The Input Control Value is the connection between the DynamThresh Tool and the numeric value that is to control the threshold adjustments. This datum can be connected to any numeric output datum (DoubleDm or IntDm). By default, this will connect to the integrated GrayHistogram Tool, but it can reference another GrayHistogram Tool or any other output. It can also be connected to an expression step output for creating custom offset conditions.

GrayHistogram Tool

When the Input Control Value is connected to a different step, and that step is deleted, the input datum will have no connection. The connection needs to be set to another output value. If you want to reconnect it to the integrated GrayHistogram Tool, choose the GrayHistogram Tool.Avg Histogram Value (symb AutoFocus1.Avg), as shown in Figure 5–7.

FIGURE 5–7. Input Control Value



Changing Thresholds of Controlled Steps

The thresholds of controlled steps can be modified when you are setting up a Job. However, the properties page will always show the current adjusted threshold value, and any change to this value will also be considered adjusted.

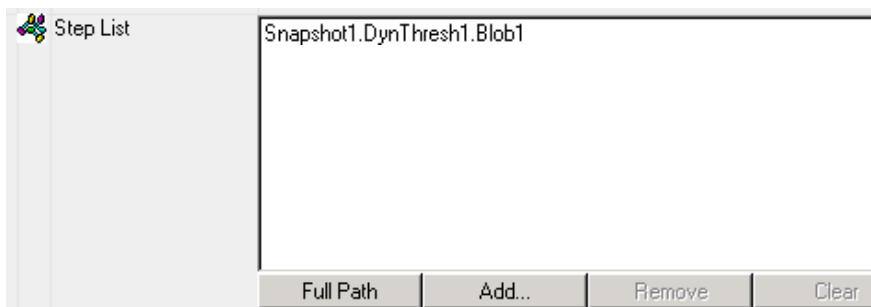
For example, assume a threshold is initially set to 40 during setup, which is stored as the base threshold. Then, after a test run, the threshold adjustment is 10, which makes the new threshold value 50. If the threshold is manually adjusted on the properties page back to 40, the threshold adjustment will be taken back out to store the new base threshold value as 30. Normally, this is of little concern, except when examining cases of clamping.

Connecting Steps Using Step List

Usually, programs are constructed by inserting tools whose thresholds are to be controlled by the DynamThresh Tool directly into it. This will automatically connect up the threshold values to the DynamThresh tool for dynamic adjustment.

However, it is not required to insert steps into the DynamThresh Tool to be controlled by it. Any step with adjustable thresholds can be connected. The step to be controlled should come after the DynamThresh Tool in execution order. Add the step to the control list by adding it to the **Step List** of the DynamThresh Tool, as shown in Figure 5–8.

FIGURE 5–8. Step List



Also, you can remove child steps that were automatically added to the step list.

Sample Applications

Adding Dynamic Thresholding to an Existing Step

The thresholds of the Blob Tool can be controlled dynamically without having to delete the tool and reinsert it inside of a DynamThresh Tool.

This is performed by inserting a DynamThresh Tool before the tool to be controlled, as was done with the Blob Tool in the example shown in Figure 5–9.

FIGURE 5–9. Inserting DynamThresh Tool into a Job

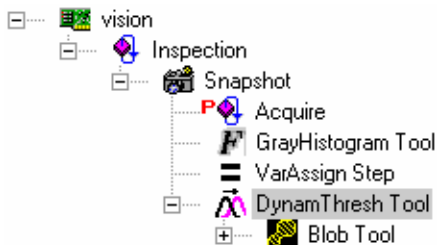


From the Step List, select the step you want to control. It is important to add it before so that execution order is correct.

Customizing Threshold Adjustment With an Expression

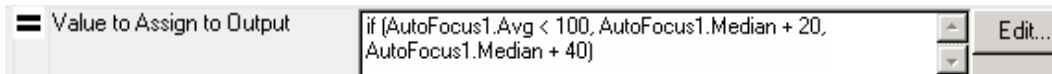
Custom dynamic threshold adjustments can be created by using the VarAssign Step. The GrayHistogram Tool can be inserted to gather data on the image characteristics. These output values can be combined in a VarAssign Step; the output of that can control the dynamic threshold, as shown in Figure 5–10.

FIGURE 5–10. DynamThresh Tool with VarAssign Job



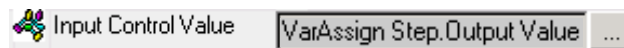
Since the GrayHistogram Tool calculates the median, high, low and standard deviation in addition to the average, more options are available for threshold selection. In Figure 5–11, the median value is used, but it will offset the median depending on the average brightness.

FIGURE 5–11. Value to Assign to Output



The output of this VarAssign Step is then selected as the input control value for the DynamThresh Tool, as shown in Figure 5–12.

FIGURE 5–12. Input Control Value



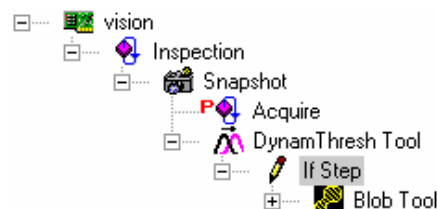
This method is also useful to limit the threshold from becoming too low and thresholding background noise. For example, in a particular inspection, the background pixel values are normally below 100 and foreground pixels are above 150 when an object is present. However, if the object is missing from the FOV, autothreshold and dynamic threshold may choose a threshold below 100, which will mistakenly consider background pixels as foreground pixels. This can be avoided by using an expression in the VarAssign Step such as:

```
if (AutoFocus1.Avg < 100, 100, AutoFocus1.Avg)
```

Controlling Execution When Thresholds are Clamped

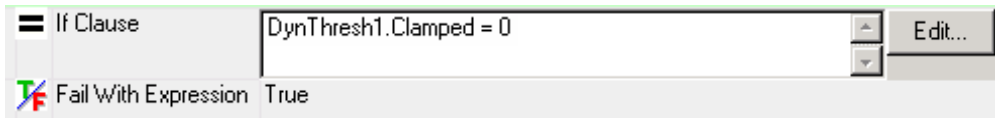
When a threshold is clamped, it may be desirable to halt execution of that inspection. This can be performed by using the Clamped output status in an If Step, as shown in Figure 5–13.

FIGURE 5–13. DynamThresh Tool with If Step



The execution of any steps within the If Step will depend on the status of the Clamped output datum, as shown in Figure 5–14.

FIGURE 5–14. If Clause



When a threshold is clamped at runtime, the if would fail and not continue with the inspection, as shown in Figure 5–15.

FIGURE 5–15. Failed Job

- ✗ Inspection
- ✓ Snapshot
- ✓ DynamThresh Tool
- ✗ If Step
 - Blob Tool
 - BlobFilter

Results

Average Gray Value

This is a copy of the Input Control Value retrieved when this step runs. When the Input Control Value is left at the default connection to the internal GrayHistogram Tool, this value will be the average gray value of the pixels in the DynamThresh Tool ROI.

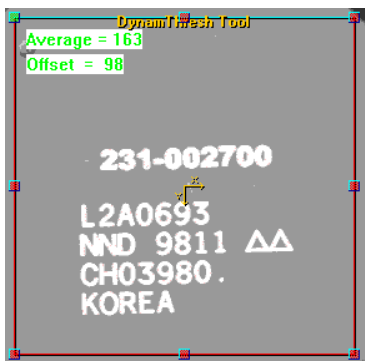
Threshold Adjustment

This is the difference between the runtime input control value and the trained reference gray value. This is the adjustment value that is applied to the thresholds of the controlled steps. The threshold adjustment is added to the base threshold value of each step to determine the step's runtime threshold values.

Edit Window

By default, the Average Gray Value and Offset will be displayed inside the upper left corner of the ROI, as shown in Figure 5–16.

FIGURE 5–16. Average Gray Value and Offset Displayed

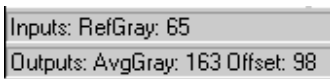


These graphics can be disabled with the **Graphics Level** parameter. The display will show red if a threshold is clamped; otherwise, it will show green.

Status Bar

The Status Bar shows the results as, shown in Figure 5–17.

FIGURE 5–17. Status Bar



Inputs: RefGray: xxx

Where xxx is the average gray value of the pixels in the ROI. This value will be stored into the Reference Gray Value.

Outputs: AvgGray: xxx Offset: yyy

Where xxx is the Average Gray Value and yyy is the Threshold Adjustment.

Reference

DynamThresh Tool

This tool adjusts the threshold values of tools at runtime. This allows threshold values to be compensated for changes in lighting conditions.

Other Steps Used

GrayHistogram Tool — Verifies whether an image is in focus, and measures how well an image is in focus by calculating a focus number. An integrated GrayHistogram Tool is included when a DynamThresh Tool is inserted into a Job. For more information, refer to “GrayHistogram Tool” on page 7-52.

Theory of Operation

This section will walk through a simple use of the DynamThresh Tool to adjust the threshold value of a Blob Tool.

1. Using FrontRunner™, create a new Job with the structure shown in Figure 5–18.

Note: For information about FrontRunner™, see the Visionscape FrontRunner User Manual.

FIGURE 5–18. Example Job



2. Configure the **Acquire** step to Load Images from File and add the image file text02.tif to the File List. This will be our reference image, the reference threshold value will based on the average gray value of this image.

Note: Text02.tif is located in:
c:\vscape\tutorials & samples\tutorials\framegrabber boards\defect detection with flaw tool

3. Click **Show Setup Display** to position the ROIs and define the reference value. Step through the Job to the DynamThresh Tool with the file image loaded into the window.
4. With the DynamThresh Tool selected, position its ROI so that it covers most of the surface of the chip. Click **Train**. This will compute the average gray value of the pixels in the ROI and store this as the reference gray value.
5. Step forward to the Blob Tool and position its ROI around the markings on the chip, as shown in Figure 5–19.

FIGURE 5–19. Blob Tool



6. Set the parameter **High Threshold** to 90 and **Blob Polarity** to **Light Parts**.
7. Click **Run Inspection Once**. Since the image is unchanged, the average gray value will be the same as before and the adjustment value will be zero, as shown in Figure 5–20.

FIGURE 5–20. Average Gray Value and Offset Displayed

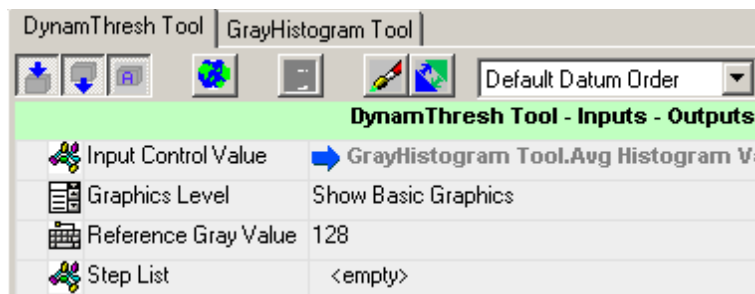
8. Add some additional images (text04.tif and text05.tif) with varying brightness levels to the Acquire step's File List and run in a Loop. The average gray value and adjustment amount will be displayed in the top left corner of the DynamThresh Tool ROI.

Note: Text04.tif and Text05.tif are located in:

c:\vscape\tutorials & samples\sample jobs\dynamthresh tool\

Description

DynamThresh allows editing through the DynamThresh properties page, as shown in Figure 5–21.

FIGURE 5–21. DynamThresh Tool Properties Page

Settings

- **Input Control Value** — Determines which datum value will calculate the offset adjustment. Any numeric output datum can calculate the adjustment values. For example, this can be connected to another GrayHistogram Tool in another image buffer or to the output of an VarAssign Step.
- **Graphics Level** — Controls the amount of graphics to be drawn on the image at runtime:
 - Show None — Disables the output graphics.
 - Show ROI Only — Draws the outline of the step's ROI.
 - Show Basic Graphics (Default) — Displays the average gray value, offset value and ROI.
 - Show Details — Displays the average gray value, offset value and ROI.
 - Show Details And Mask — Displays the average gray value, offset value and ROI. Masking is not allowed for this tool.
- **Reference Gray Value** — The average gray value computed at training time is stored here as the reference. The runtime gray value is compared against this reference value to determine the offset adjustment. This value will usually be determined automatically at training time.

Default: 128

Range: -255 to 255

- **Step List** — Contains the list of steps whose thresholds are to be adjusted dynamically by this step. Steps can be added and removed using the buttons provided. Any step that contains adjustable thresholds can be added to this list. Any child step added to the DynamThresh Tool will be added to this list automatically.

Training

Training involves placing the ROI around an area of the image to create a template and initiate the train. The average gray value of the pixels in the ROI will be calculated and that value stored as the reference value.

Results

- **Status** — Set to true after a successful execution of the step.
- **Average Gray Value** — The runtime average gray value computed by the GrayHistogram Tool.
- **Threshold Adjustment** — The difference between the Average Gray Value and the Reference Gray Value.
- **Threshold Clamped** — True when a threshold has been clamped to 0 or 255.

I/O Summary

DynamThresh Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: RefGray: xxx

Where: xxx = The average gray value of the pixels in the ROI.

Outputs: AvgGray: xxx Offset: yyy

Where: xxx = The average gray value.
 yyy = The threshold adjustment.

This chapter describes the use of I/O in Visionscape systems.

Introduction

Visionscape hardware provides general purpose physical I/O points (GPIO) and sensor and strobe points on the PCIe I/O card or smart camera. GPIO consist of optically-isolated discrete I/O points that can be written or read under software control. This I/O can be used by the software for control and communication purposes.

In addition to the GPIO points, sensors and strobes, Visionscape provides 2048 points of software I/O called Virtual I/O. Virtual I/O are 32-bit longwords that can simulate physical I/O, or as data registers. These I/O are written directly through software. No physical control lines are necessary. The data is carried over Ethernet or the PCI bus.

All I/O is accessible to clients on the same PC or on the network using TCP/IP. For information on programmatic access to I/O using COM, refer to Chapter 8 of the Visionscape Programmers Kit (VSKit) Manual for comprehensive information on using the AvpIOClient COM Object

I/O may be employed either through various steps in your Job, such as the DigitalInput and DigitalOutput Steps, which can accept I/O points as inputs or outputs, or Output Valid, which generates data valid signals, or in Inspection or Sequence steps, which generate pass/fail signals.

General Purpose I/O

The Visionscape PCIe provides a complement of optically-isolated I/O communications points called general purpose I/O, or GPIO. Each PCIe has 32 GPIO points 16 inputs and 16 outputs. Refer to the GigE Getting Started manual for more information.

PCIe points are mapped to Virtual Software I/O points, starting at Virtual Software I/O point 160. These points can be selected by choosing the Virtual Software I/O type and choosing an I/O point in the range of the PCIe.

Note: When selecting an I/O Board input point for triggering, only the first 8 points can be used for low-to-high trigger events and only the second 8 input points can be used for high-to-low trigger events.

You can configure the smart camera GPIO in the Vision System Step properties page by setting GPIO Input Mask. This mask defines which I/O points are programmed as inputs and which are defined as outputs.

By default:

TABLE 6–1. GPIO

GPIO Points	Input/Output
1 - 8	Input (Selected)
7 - 16	Output (Not Selected)

If you are using a smart camera:

- GPIO Point 1 is Input (Selected)
- GPIO Points 2-4 are Outputs (Not Selected)
- GPIO Points 5-8 can be either Input or Output (Outputs by default)

Sensors And Strobes

The camera I/O card supports four sensor points and four strobe points. These can also be used as general purpose binary I/O. The sensor points are always inputs and the strobe points are always outputs.

Virtual Software I/O

Each Vision System supports 2048 points of software I/O. Virtual I/O are 32-bit longword registers that can be used as either inputs or outputs at any time. You can use a software I/O point as a binary value (zero versus non-zero), or you can program the DigitalOutputs step to write longword values to the software I/O point.

All Vision System types (Vision Processor, Vision Accelerator, smart cameras, and software-based systems) support Virtual I/O. Software-based systems support I/O when the system resides on the AvpBackplane. See Chapter 2 of the Visionscape FrontRunner User Manual for more details about creating software-based systems on the AvpBackplane.

Important Notes

When using Virtual I/O points to generate triggers at regular interval, as programmed in the Visionscape IO Display or the FrontRunner RunView IO bar or from within the AVP itself with an IO Inspection, the accuracy of the timing between triggers will depend on the Operating System the AVP runs on and also on the number of Virtual I/O points programmed to be triggers. The following information provides guidelines based on Device/OS configurations:

- Smart Cameras — Accuracy < 1 msec, independent of the number of virtual triggers generated by the AVP. Trigger to trigger time depends on relative priority of Inspection.
- Software Systems — Triggers generated by the avp, IO display and FrontRunner IO bar.
 - Windows 10 or 7 — Accuracy +/- 5 msec typical, trigger to trigger time increases as more triggers are generated.

Note: The values above are typical and may vary from PC to PC. Accuracy should be measured/tested first for the particular avp for the particular PC. When greater accuracy than the Device allows is required, use physical triggers.

Network Access

Visionscape I/O is accessible from the PC using TCP/IP. You can create COM-based programs using Visual Basic or Visual C++ that utilize the AvpIOClient COM Object to communicate seamlessly to a Vision System either on the local AvpBackplane or over the Ethernet network. Accessing a Vision System's I/O entails referencing the system using UNC, for example “\\pc_system\1300.1”.

For more information on COM access to Visionscape I/O, refer to Chapter 8 of the Visionscape Programmers Kit (VSKit) Manual.

I/O In Visionscape Steps and Tools

Several Visionscape steps, or tools, support I/O. In each of these steps, an I/O point is selected as an input or an output, and data is read from/written to the I/O point when the step executes.

Digital Inputs Step

The Digital Inputs Step is the general purpose step for reading I/O points and using them within the Inspection. The step can read the 16 GPIO points and a programmable range of Virtual I/O points. The step can also act as a general purpose trigger signal (or Data Valid Input signal).

Digital Outputs Step

The Digital Outputs Step is the general purpose step for writing I/O points from within the Inspection. Each I/O point is set to an evaluated value. Digital Outputs support any I/O point (except hardware-based inputs).

Output Valid Step

The Output Valid Step is a private step that provides an output “data valid” signal for a specified duration. The signal can be written to any output point for a custom duration. Typically, steps that generate an I/O signal (Digital Outputs, Sequence, Inspection) have a private Output Valid step for this purpose. When the data valid signal is asserted, the Inspection is blocked for the specified time period.

Inspection Step

The Inspection Step uses six I/O points:

- **Busy Signal IO** — I/O point that is set/cleared when the Inspection is busy. If the Inspection has a triggered acquisition, then the Inspection is considered busy when the trigger fires. If the Inspection is not triggered, then the busy signal is asserted when the first step within the Inspection is executed. The point is then cleared when the iteration completes. You can also set the polarity of the signal in the Inspection settings under **Busy Signal Polarity**.

- **Part Queue Almost Full IO** — I/O point that is used only when the Part Image Queue is enabled. It is asserted when the queue is nearly full. Use this I/O point as a signal to know when to stop the Inspection and clear the Part Image Queue.
- **Part Queue Full IO** — Specifies an IO point to signal when the Part Queue is full.
- **Ready To Run Output** — Specifies an IO point to signal while an inspection is running.
- **Record Entered Into Queue IO** — I/O point that is used when the Part Image Queue is enabled. Images can be stored in the Part Image Queue during execution and uploaded for review at a later time. When an image is added to the queue, this I/O point is asserted.
- **Status Output** — I/O point that is set when the Inspection iteration passes. It is cleared when the Inspection iteration fails. This signal is updated at the end of the iteration before the Output Valid Step of the Inspection is executed.

Sequence Step

The Sequence Step uses an I/O point to signal the pass/fail status of the Sequence. To determine its status, Sequence uses the set of tools contained within it. If they all pass, the Sequence passes. If one fails, the Sequence fails. After the set of tools is executed, the Sequence updates its I/O status point.

Acquire Step

The Acquire Step uses up to five I/O points in its acquisition definition.

- **Pic Done IO** — This I/O point is cleared when the acquisition is started and set when the acquisition completes and the image is in memory.
- **Pic Expose IO** — This I/O point is cleared when the acquisition has started and is set when the camera has successfully exposed the image. The image has not yet been transferred to memory when this point is set.
- **Trigger** — This I/O point is the trigger signal for the acquisition. The polarity of the signal is defined by **Trigger Polarity**.

- **Trigger Overrun IO** — This I/O point is asserted when a trigger overrun occurs. A trigger overrun occurs when a trigger event occurs while the camera I/O card is processing the previous trigger.
- **Process Overrun IO** — This I/O point is asserted when a process overrun occurs. A process overrun occurs when there are no buffers left in the bufferpool for an acquisition event to use. This usually occurs when the Inspection itself is not processing the buffers fast enough and, therefore, is not returning the buffers to the bufferpool when they are released. This overrun can also occur if there are not enough buffers in the bufferpool to handle all the functionality you have active in your Inspection (PC image displays, Keep Last Failed Data, and so on).

Tolerance Meas and PointTolerance Meas Tools

The Tolerance Meas and PointTolerance Meas tools each allow you to select an output point through **Output to**.

- **Output To** — This I/O point indicates the pass/fail status of the Tolerance Meas or PointTolerance Meas tool.

References

The remainder of this chapter describes the following steps:

- “Digital Inputs Step” on page 6-8
- “Digital Output Step” on page 6-12
- “Output Valid” on page 6-14

Digital Inputs Step

This step reads the state of all GPIO and a range of software I/O points at a specified time. Then, the state of the inputs are used in steps that employ expressions for a wide variety of purposes. The DigitalInput Step supports an optional Data Valid Signal, which is an input trigger to indicate when to sample the state of the input I/O points. The Signal Polarity indicates whether the I/O points are sampled on a Low-to-High or a High-to-Low transition. A trigger overrun I/O point is set when a trigger overrun occurs. You can also change the style of the trigger to use it as a “state” machine. That is, when the state mode is used, the step checks the I/O point for the correct state. As long as it is in the right state, the Step does not block execution.

The DigitalInputs Step provides a means to input data to the Inspection. The inputs originate either from general purpose physical inputs or from virtual I/O. There is no specific output of DigitalInputs Step other than the state of the received inputs.

Other Steps Used

None.

Theory of Operation

DigitalInputs provides you with a set of integer input datums. You use the datums to access the input values within expressions in the Inspection. Also, you can select an Input Data Valid point and its polarity. When the point is used, the Inspection can block execution until the correct transition is received, or it can block execution until the correct I/O state is encountered. Input data is then sampled when the trigger or state is received.

The Input trigger can be any I/O point listed in the Data Valid Signal list.

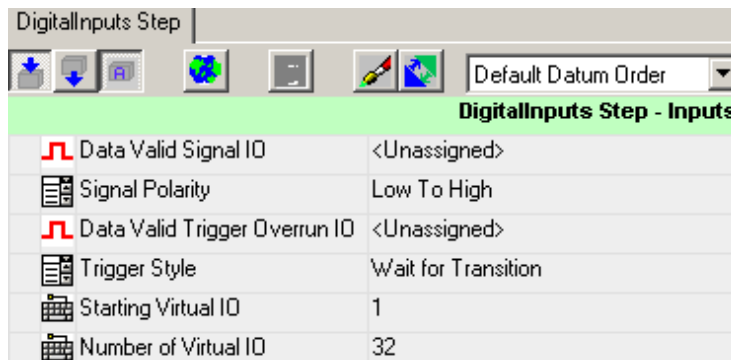
The I/O is not pipelined on the system and the inputs are sampled when the Step is executed, not necessarily when the Data Valid Signal occurs. If the data valid signal occurs during the execution of the rest of the tree, this is considered an overrun condition, and the overrun I/O point is asserted.

DigitalInputs has output datums for each of the inputs, named Digital1 to Digital8. In addition, a range of Virtual input has output datums named “Virtual” suffixed by the IO number. The default range has 32 outputs named Virtual1 to Virtual32, to represent the first 32 points of software I/O. The range of Virtual inputs is selectable by setting the **Starting Virtual IO** and **Number of Virtual IO** datum values. These datums are accessible to an expression, typically in DigitalOutputs or IF steps.

Description

DigitalInputs Step allows editing through the DigitalInputs Step properties page, as shown in Figure 6–1.

FIGURE 6–1. DigitalInputs Step Properties Page



Settings

- **Data Valid Signal IO** — Input point used as the Data Valid trigger signal. When set to none, the Inspection is not blocked at all and the inputs are sampled at the Step's execution time. When it is set to an input point, execution of the Inspection blocks at the Step until the trigger signal of the appropriate polarity is received. The inputs are then sampled and execution continues. If **Trigger Style** is set to "Wait for State", then the I/O point is tested for the correct state according to the polarity. If it is in the correct state, the Inspection continues normally. If it is not, the Inspection is blocked until the I/O changes state. State changes of the I/O point outside of this tool's execution do not necessarily block the Inspection, only the state of the I/O point at the time of the Step's execution.

For serial input triggers, select the device's RS-232 or TCP port from the drop-down list and enter the trigger string in the edit box, as shown in Figure 6–2. The acquisition will be triggered each time the trigger string is received on the port.

FIGURE 6–2. Serial Trigger for Digital Inputs



Enter non-printing characters using the "C" language escape sequences, as shown in Table 6–2.

TABLE 6–2. Supported Escape Sequences

Sequence	Output Character
\a	Bell (alert)
\b	Backspace
\f	Formfeed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\	'Single quotation mark

TABLE 6–2. Supported Escape Sequences (continued)

Sequence	Output Character
\"	Double quotation mark
\\	Backslash
\ooo	ASCII character in octal notation
\xhhh	ASCII character in hexadecimal notation

The device’s available TCP ports and RS-232 settings are configured through the Configure Device dialog box in the File menu.

To use the RS-232 port of the smart camera for serial triggering, set the ConsoleTTY to zero in the boot parameters of the camera (factory default) using the Bootloader menus.

- **Signal Polarity** — Determines the polarity of the trigger signal, either Low to High or High to Low. If the Wait for State mode is used, Low to High means High and High to Low means Low.
- **Data Valid Trigger Overrun IO** — Output point that signals a trigger overrun condition. Overruns can occur only in Runtime and only in “Wait for Transition” mode when the input point is triggered and the DigitalInputs Step cannot react to the signal.
- **Trigger Style** — Determines the style of trigger. When using a serial trigger, set to Wait for Transition.
 - If Wait for Transition is used, the Step triggers on a transition signal and trigger overruns can occur.
 - If Wait for State is used, the Step triggers when an I/O point reaches the desired state but continues to trigger as long as the I/O point remains in that state.
 - If Wait for Transition, Ignore Overruns is used, the Step triggers on a transition signal and overruns are ignored. No overrun warning message will be displayed, and the Data Valid Trigger Overrun IO will not be set when an overrun occurs.
- **Starting Virtual IO** — Specifies the first IO number of the range of virtual IOs to be read and made available as output datums.

Default: 1

Range: 1 to 2048

- **Number of Virtual IO** — Specifies the number of sequential Virtual IOs to be read and made available as output datums starting with and including the Starting Virtual IO.

Default: 32

Range: 0 to 2048

Training

None.

Results

The results of DigitalInputs Step are dictated in the input D1 to D8 on the camera and V1 to V32, by default. These datums store the current value of the sampled input and maintain the value until the next time the inputs are sampled.

I/O Summary

The DigitalInputs Step provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Input: Indicates which I/O point is being used as a Data Valid signal and the desired polarity, or OK if not used.

Output: Waiting for trigger signal..., Triggered, or OK.

Digital Output Step

This step sets the state of one or more I/O points at a specified time. You specify the number of output expressions and assign each expression an I/O point. The DigitalOutput Step can synchronize inspections with each other, Visual Basic code, or external hardware.

The DigitalOutput Step provides a means to output expressions from the Inspection optionally using a data valid signal. The outputs can be any I/O points listed in the associated I/O List datum with the expression.

The DigitalOutputs Step accepts no input.

Other Steps Used

Output Valid Step — Generates a data valid output signal after execution of DigitalOutputs Step for a user-specified duration.

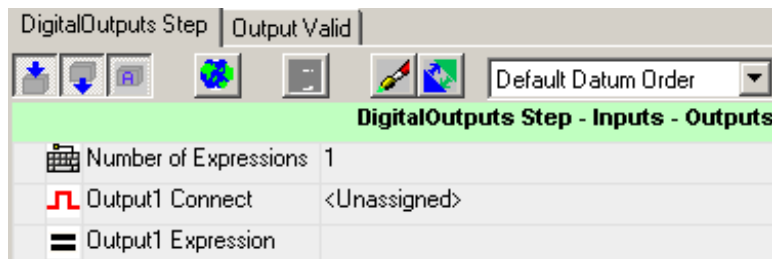
Theory of Operation

The DigitalOutput Step enables you to enter expressions for any I/O point. You create a set of output expressions and assign each expression to an I/O point. You can also use the embedded Output Valid Step to specify a data valid signal and duration.

Description

The DigitalOutputs Step allows editing through the DigitalOutputs Step properties page, as shown in Figure 6–3.

FIGURE 6–3. DigitalOutputs Step Properties Page



Settings

- **Number of Expressions** — The number of expressions contained in this step. Each expression is tied to an output. You can alter this number at any time to add more expressions to the step.
- **Output1 Connect** — An I/O List datum that displays the output to which the nth expression is tied. When the expression is calculated, the value of the expression is written to the I/O point. If the I/O point is only a bit, then it is asserted or cleared based on the expression evaluating to zero or non-zero. If the I/O point is a byte or larger, the value of the expression is written to the I/O point.
- **Output1 Expression** — The expression for the output. Each expression is calculated at runtime and written to the I/O point listed in the corresponding Outputn Connect datum. If no expression is entered, then the I/O point is not altered in any way.

Training

None.

Results

The results of DigitalOutputs are stored in the output datums until the next time DigitalOutputs is executed.

I/O Summary

The DigitalOutputs Step provides a I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Compile Error (yy): xxx

Outputs: Compile Error (yy): xxx

Where: xxx = description of compilation error or OK
 yy = error code

Output Valid

This step provides a way to assert a data valid signal using I/O for a specified duration in milliseconds. When set, the current Inspection is blocked for the amount of time given.

Other Steps Used

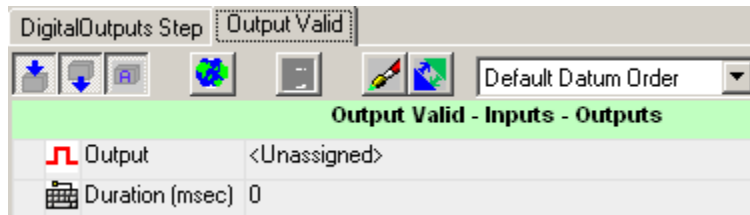
None.

Theory of Operation

Output Valid provides a way to assert a data valid signal using I/O. It cannot be inserted into the Step tree through the interface. It is used primarily as a component of a step that uses I/O. The parent step sets its I/O, and then runs Output Valid to signal data valid.

Description

Output Valid allows editing through the Output Valid properties page, as shown in Figure 6–4.

FIGURE 6–4. Output Valid Properties Page

Settings

- **Output** — Specifies an I/O point to use for data valid. Any output point listed can be selected.
Default: <none>, indicating that no data valid signal is to be used
- **Duration (msec)** — Duration of the signal in milliseconds. When set to a value higher than 0, the data valid signal is set high for the specified duration followed by a low setting for the same duration.

Default: 0

Training

None.

Results

None.

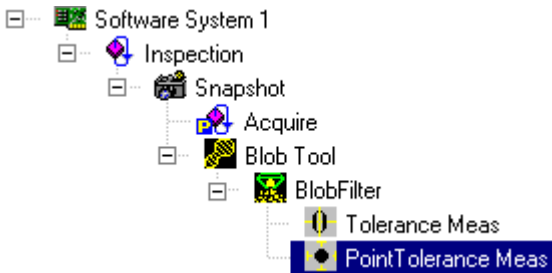
I/O Summary

None.

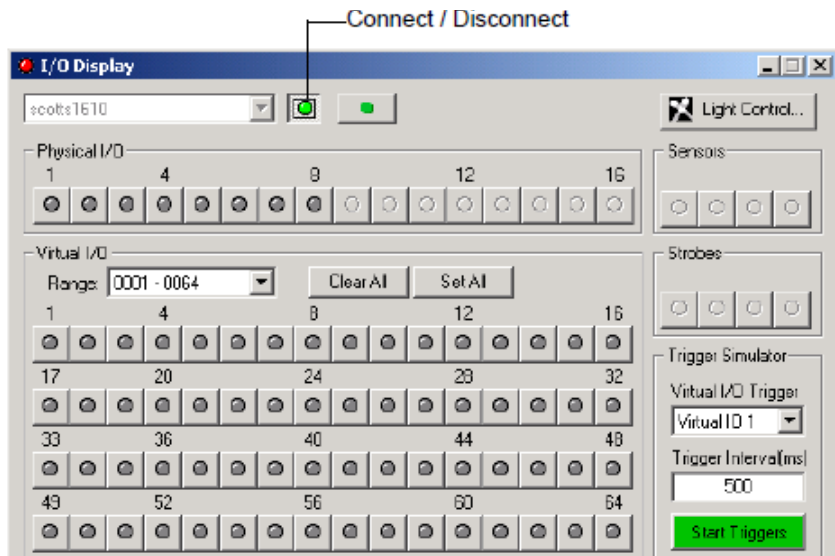
I/O Example

The following example illustrates the use of Digital I/O in Visionscape.

1. Build a Job, as shown in Figure 6–5.

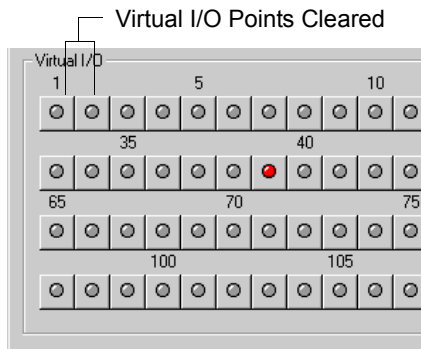
FIGURE 6-5. Building I/O Job

2. Click **Tolerance Meas** in the Job Tree. This will display the **Tolerance Meas** properties page. Set the **Output To** parameter to **Virtual Point 0001**.
3. Click **PointTolerance Meas** in the Job Tree. This will display the **PointTolerance Meas** properties page. Set the **Output To** parameter to **Virtual Point 0002**.
4. Train the **Tolerance Meas** and the **PointTolerance Meas** tools.
5. Open the **View** menu and select **I/O Display**. This displays the **I/O Display** window, as shown in Figure 6-6.

FIGURE 6-6. I/O Display Window Showing Disconnected I/O Points

6. Select the appropriate Vision System name to view from the drop-down list located in the upper left corner of the window. This drop-down list contains all the Vision Systems in your PC. In our example, we have selected scottsl610.
7. Click **Connect/Disconnect**. This allows you to connect to the Vision System.
8. Click on an I/O point button to toggle its state between cleared to set. In our example, virtual I/O points 1 and 2 have been cleared Off, as shown in Figure 6–7.

FIGURE 6–7. Virtual I/O Points 1 and 2 Cleared



9. Tryout the Inspection in FrontRunner™ to generate a trial run of the Job. The virtual I/O points 1 and 2 are set On (appear red) when the tolerance tools pass, as shown in Figure 6–8.

FIGURE 6–8. Virtual I/O Points 1 and 2 Set

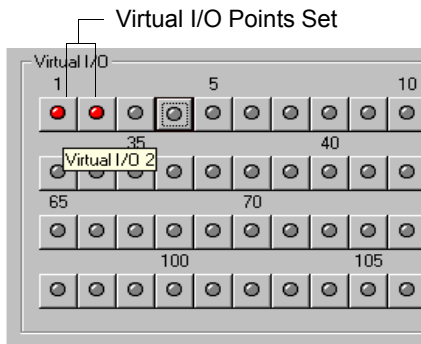


Image Analysis Tools

The Visionscape Image Analysis tools extract feature data from an image without modifying the gray levels of the image. These tools have a number of features in common, such as an image buffer and an ROI as input. The ROI designates the area in the image within which the analysis is to be performed. This ROI can be moved, sized, and rotated when setting up the tool, and can also be offset by the output of another feature-finding tool at runtime. Typically, the output of an Image Analysis tool is one or more values describing features found within the ROI. Examples of these outputs are point locations, numbers of edges, or gray level values within a given range.

Connectivity Analysis

Blob Tool

This tool performs connectivity analysis on the image, finding groups of connected pixels with gray levels above or below a threshold. Each group is called a blob. The Blob Tool records all the detected blobs into a blob tree. For each blob, the tool calculates many geometric features including the length, width, area, etc. In addition, the number of blobs and number of parts in the ROI are calculated. A part is a blob that has the opposite color of the background. A hole is a blob with the same color as the background.

Other Steps Used

- **AutoThreshold** — Automatically calculates the thresholds (low and high) used to binarize the ROI. AutoThreshold calculates the high threshold when Blob Tool is looking for dark parts (low threshold = 0), or calculates the low threshold when the tool is looking for light parts (high threshold = 255).
- **BlobFilter** — Filters out the blob that satisfies the filtering criteria specified in the BlobFilter properties page (see “Blob Filter” on page 7-14). BlobFilter can only serve as a child step of a Blob Tool. When the Blob Tool is inserted into a Job Tree, a BlobFilter step is automatically inserted into the Blob Tool. Multiple BlobFilter steps can also be inserted into the Blob Tool to filter out blobs according to different criteria.

Theory of Operation

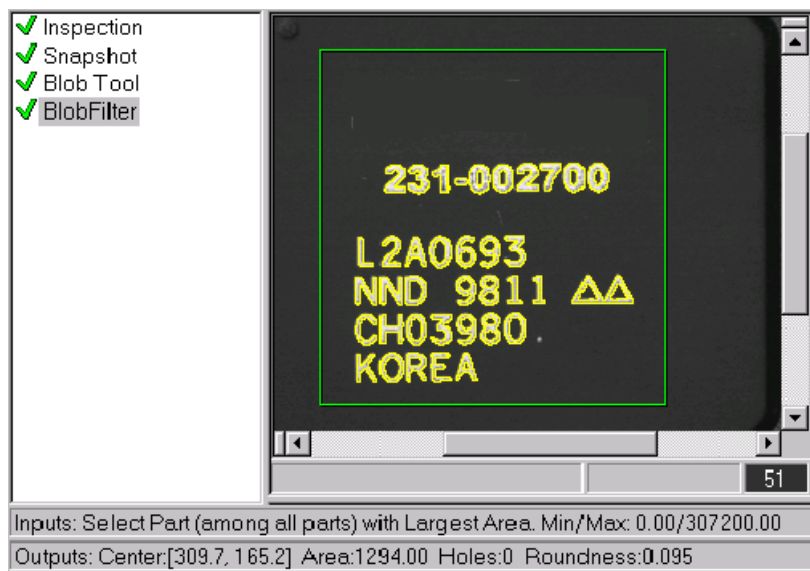
A blob is either a part or a hole. A part is a blob (e.g., connected region) of foreground. A hole is a blob of background that is completely surrounded by a connected region of foreground. In other words, blobs inside the background are called parts. Blobs inside parts are called holes of that part.

The Blob Tool locates all parts and holes within the ROI. It separates the image into light and dark pixel regions using two grayscale thresholds, determines which regions are connected (connectivity), or touch one another, and then records many useful features about each blob. Fifty different features, including area, centroid position, major and minor axes, and orientation angle, are available. All blobs that are found are organized into a blob tree. The blob tree contains information about the geometric relationship of the blobs.

Note: The Blob Tool does not return information about any given blob. This operation is performed by BlobFilter steps.

After a run, all blobs found are highlighted in yellow when the Graphics Level is set to Show Basic Graphics or better, as shown in Figure 7-1. The BlobFilter's resultant blob is highlighted in green.

FIGURE 7-1. Blobs Found Highlighted Around Logo and Font Areas



Description

The Blob Tool allows editing through the Blob Tool properties page, as shown in Figure 7-2.

FIGURE 7-2. Blob Tool Properties Page

Blob Tool | BlobFilter | AutoThreshold

Default Datum Order

Blob Tool - Inputs

InputBuffer	Snapshot.SnapOutputBuffer
Use Autothreshold	<input checked="" type="checkbox"/>
Low Threshold	0
High Threshold	0
Blob Polarity	Dark Parts
Minimum Blob size	10
Maximum Blob size	1000000
Apply minblob to parts	<input type="checkbox"/>
Apply maxblob to parts	<input type="checkbox"/>
Filter blobs by Area	<input type="checkbox"/>
Ignore blobs that touch the ROI	<input type="checkbox"/>
Discard children blobs	<input checked="" type="checkbox"/>
Keep all blobs	<input checked="" type="checkbox"/>
Min Total Area	0.000
Max Total Area	307200.000
Min Number of Blobs	0
Max Number of Blobs	5000
Pass On No Data	<input type="checkbox"/>
Calculate gray features	<input type="checkbox"/>
Min asymmetry length Thr	0.250
Min asymmetry width Thr	0.250
Calculate only hole moments	<input type="checkbox"/>
Maximum ASIC Rle Width	504
Maximum Rle Segments	2048
Process Every Nth pixel	4
Process Every Nth line	2
Graphics Level	Show Details
Use Input Mask	<input type="checkbox"/>
Calculations To Perform	Default

TABLE 7-1. Links to Property Descriptions

For Information About...	Go To...
Apply maxblob to parts	page 7-6
Apply minblob to parts	page 7-6
Blob Polarity	page 7-6
Calculate gray features	page 7-7
Calculate only hole moments	page 7-7
Calculations To Perform	page 7-7
Discard children blobs	page 7-6
Filter blobs by Area	page 7-6
Graphics Level	page 7-9
High Threshold	page 7-6
Ignore blobs that touch the ROI	page 7-6
Keep all blobs	page 7-6
Low Threshold	page 7-6
Max Number of Blobs	page 7-7
Max Total Area	page 7-7
Maximum ASIC Rle Width	page 7-9
Maximum Blob Size	page 7-6
Maximum Rle Segments	page 7-9
Min asymmetry length Thr	page 7-8
Min asymmetry width Thr	page 7-8
Min Number of Blobs	page 7-7
Min Total Area	page 7-7
Minimum Blob Size	page 7-7
Pass On No Data	page 7-7
Process Every Nth line	page 7-8
Process Every Nth pixel	page 7-8
Use Autothreshold	page 7-6
Use Input Mask	page 7-10

- **InputBuffer** — Displays the name of the image buffer that the tool will process. The default buffer is the output buffer of its parent. This property is read-only.

Binarization Properties

These properties control how an image is binarized by the Blob Tool.

- **Blob Polarity** — Defines whether light or dark pixels comprise blobs according to the rule, as described in “Theory of Operation” on page 7-2.
- **Low/High Thresholds** — Any pixel gray value that falls between low and high when looking for dark parts or outside low and high when looking for light parts become the foreground pixels (binary value 1). Other pixels make up the background pixels (binary value 0).
- **Use Autothreshold** — Calculates either low threshold (light parts) or high threshold (dark parts).

Filtering Properties

These properties control how each blob is filtered out from the resulting blob tree:

- **Apply maxblob to parts** — When enabled (checked), applies the **Maximum Blob Size** to parts. When disabled, applies the **Maximum Blob Size** to parts and holes.
- **Apply minblob to parts** — When enabled (checked), applies the **Minimum Blob Size** to parts. When disabled, applies the **Minimum Blob Size** to parts and holes.
- **Discard children blobs** — Discards child blobs contained within a discarded originator blob.
- **Filter blobs by Area** — When enabled (checked), applies **Minimum Blob Size** and **Maximum Blob Size** to the blob’s **Area**. The **Area** is the sum of the pixels that form the part of the blob. When disabled, applies **Minimum Blob Size** and **Maximum Blob Size** to the blob’s **Total Area**. The **Total Area** is the sum of the pixels that form the part and the holes of the blob.
- **Ignore blobs that touch the ROI** — Discards blobs that touch the boundary of the ROI.
- **Keep all blobs** — Records all blobs that satisfy the filtering conditions into the blob tree. Otherwise, keeps the one with the largest area in pixels.
- **Maximum Blob Size** — Maximum blob size in pixels. Any blob whose area is greater than this value will not be recorded in the blob tree.

Default: 1000000 (1 million)

- **Minimum Blob Size** — Minimum blob size in pixels. Any blob whose area is less than this value will not be recorded in the blob tree.

Default: 10

Criteria Properties

These properties control whether the Blob Tool should pass or fail.

- **Min Number of Blobs/Max Number of Blobs** — When the number of found blobs falls out of this range, the Blob Tool fails.
- **Min Total Area/Max Total Area** — When the total area, in pixels, of all found blobs falls out of this range, the Blob Tool fails.
- **Pass On No Data** — When no blobs are found, this property determines whether the tool should pass or fail.

Processing Properties

These properties control how much information is calculated or kept for each blob.

- **Accumulate boundary pts** — Stores the perimeter information. This is required to compute the perimeter length and roundness features. This option is only visible when **Calculations To Perform** is set to Custom.
- **Calculate gray features** — Calculates the average grayscale of a blob.
- **Calculate only hole moments** — Calculates hole moments so that additional features can be returned for holes (e.g., area, center).
- **Calculate perimeter features** — Calculates features based on the perimeter points. These include average radius and radius ratio. This option is only visible when **Calculations To Perform** is set to Custom.
- **Calculate symmetry feature** — Calculates length and width (in rotated space) of the blob, as well as asymmetry (offset between Center of Area and center of bounding box at given blob orientation). This option is only visible when **Calculations To Perform** is set to Custom.
- **Calculations To Perform** — Five levels of calculations are provided for indicating which blob results are to be reported by the Blob Tree result.
 - Default — Xcent, Ycent, Area, Color.

- Basic — Default results plus Angle, Nholes, Xmin, YatXmin, Xmax, YatXMax, YatYmin, Ymin, XatYmax, Ymax, Xdiff, Ydiff, Major, Minor, Areatatio, Minora, Minorb, Minorc, Majora, Majorb, Majorc.
- Area — Basic results plus Totarea, Holearea, Holeratio, Boxarea, Boxarearatio, Axratio.
- All — Area results plus PEround, Length, Width, Lenratio, Avgrad, Rmin, Rmax, Radratio, Rminang, Rmaxang, X3sign, Y3sign, Perimeter, Ppda, Rminx, Rminy, Rmaxx, Rmaxy.
- Custom — Allows a custom application result list to be specified by Visual Basic.
- **Compute first moments** — Calculates first moment of area. This is required for computing the center of a blob. This option is only visible when **Calculations To Perform** is set to Custom.
- **Compute second moments** — Calculates second moments of area. This is required to compute a blob orientation. This option is only visible when **Calculations To Perform** is set to Custom.
- **Min asymmetry length Thr** — Minimum length differential to measure the asymmetry of the blob along the major axis.
- **Min asymmetry width Thr** — Minimum length differential to measure the asymmetry of the blob along the minor axis.
- **Process Every Nth line / Process Every Nth pixel** — These properties can enhance significantly the tool performance at a small cost in accuracy by processing less boundary pixels.

Range: 1 to 32

Expert Properties

These properties control internal parameters of the ASIC implementation of Blob.

- **Hardware Acceleration** — Enables ASIC for blob processing. When disabled, the blobs processed on the host processor.
- **Maximum ASIC Rle width** — Defines the width of the ROI processed internally by the ASIC. The maximum width is 512. For images that are very cluttered, the ASIC may overflow at that width. This parameter affects speed as the ASIC performs additional passes through the image when the ROI width is greater than this setting.

Default: 504 (50Hz format) for most realistically complex images

- **Maximum Rle Segments** — Controls how much storage to allocated per line for the ASIC hardware. This value should be set to the buffer width on which the tool is operating.

Default: 2048 (50Hz format)

Range: 640 to 16384

Graphic Properties

These properties control graphic and mask parameters.

- **Graphics Level** — Five graphics levels are provided for drawing the result of this tool:
 - **Show None** — No graphics are displayed.
 - **Show ROI Only** — Draws the ROI only. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red.
 - **Show Basic Graphics** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The outlines of all blobs are drawn in yellow.
 - **Show Details** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The outlines of all blobs are drawn in yellow.

- Show Details and Mask — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The outlines of all blobs are drawn in yellow. The mask pixels are drawn in red.
- Use Input Mask — This property is applicable only when the Blob Tool has a component step that produces a mask buffer as an output. When such a child step is inserted into the Blob Tool, **Use Input Mask** is automatically enabled:
 - When **enabled**, the Blob Tool must be trained. After it is trained, the mask pixels are highlighted in red.
 - When **disabled**, this property will allow the Blob Tool to retain the child mask-generating step but does not apply the mask at runtime. When there is no child mask-generating step, the Blob Tool does not require training other than setting the size and position of the input ROI.

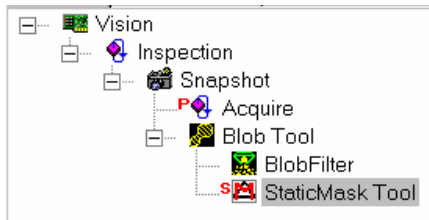
When a mask-generating step is inserted into the Blob Tool, the tool must be trained when **Use Input Mask** is enabled.

Training

No special training is required for the Blob Tool other than setting the size and position of the input ROI.

When a mask-generating step is inserted into a Blob Tool, the tool becomes trainable and has to be trained when **Use Input Mask** is enabled.

Create a Job, as shown in Figure 7–3.

FIGURE 7-3. Job - Example

1. Adjust settings in the properties page of the StaticMask Tool.
2. Disable the Align Rgn at Training setting.

FrontRunner will show two ROIs: the Blob Tool and the StaticMask Tool.

3. Size and position the Blob Tool ROI.
4. Size and position the StaticMask ROI to where the foreground is to be masked out while Blob Tool runs.
5. Move the StaticMask ROI.

You should see the ROI bounded by the Blob Tool ROI. The region to be masked should be within the region to be processed by the Blob Tool.

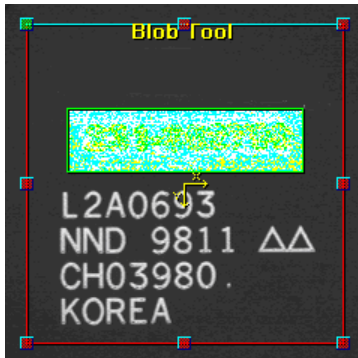
6. Click Train to train the Blob Tool.

The masked pixels are highlighted, as shown in Figure 7-4.

FIGURE 7-4. Highlighted Masked Pixels

7. Once the Blob Tool is trained, sizing the ROI will center the mask.
8. If you want to mask out an entire sub-region inside the Blob Tool ROI, change **Mask-Generation** to Fill. You should see the whole StaticMask Tool ROI is masked out as shown Figure 7–5.

FIGURE 7–5. StaticMask Tool



Results

- Status — Set to true after a successful execution of the step.
- Blob Tree — The blob tree to be used as input to the BlobFilter.
- Number of blobs — The number of blobs found in the ROI excluding the masked region.
- Number of parts — The number of parts found in the ROI excluding the masked region.
- Total Area — The total area of all blobs found in the ROI excluding the masked region.

I/O Summary

The Blob Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Lowthr: xxx, HighThr: xxx or AutoThr, MinBlob: xxx, MaxBlob: xxx, Polarity: xxx, Filter flags:[TCAPB], Results Flags:[M1M2PRLHK]

Outputs: LowThr: xxx, HighThr: xxx, NumParts: xxx, NumBlobs: xxx or Error message, Min/Max Num Blobs: xxx/xxx, Min/Max TotArea: xxx/xxx

Where: xxx = actual value

Filter Flags: uppercase letter for flag ON, lowercase for flag OFF for T (t) or C (c) only,

- T or t — Filter blobs that touch ROI
- C or c — Discard children blobs
- A or a — Filter blobs by total area or area
- P or B — Apply minblob to parts or blobs
- P or B — Apply maxblob to parts or blobs

Result Flags: uppercase letter for flag ON; lowercase for flag OFF

- M1 or m1 — First moments
- M2 or m2 — Second moments
- P or p — Keep boundary points
- R or r — Perimeter features
- L or l — Symmetry features
- H or h — Calculate hole moments
- K or k — Keep blobs

Error Messages:

- Blob agent execution failure
- Blob Vision Agent could not be created
- Not enough Temp memory in ASIC bank to run rle
- Number of blobs is out of specified range
- Ran out of memory (image too complex)
- Too many blobs (image too complex); more than: 4096 blobs
- Too many segments (image too complex) caused ASIC low/high overflow
- Total area is out of specified range

Blob Filter

BlobFilter filters out the blob that satisfies the filtering criteria specified in the BlobFilter properties page. BlobFilter is a child step of the Blob Tool.

Other Steps Used

Blob Tool — Performs connectivity analysis on the image to find groups of connected pixels with gray levels above a low threshold and below a high threshold. Since BlobFilter is a child of the Blob Tool, the Blob Tool must be inserted into your Job before you have access to BlobFilter.

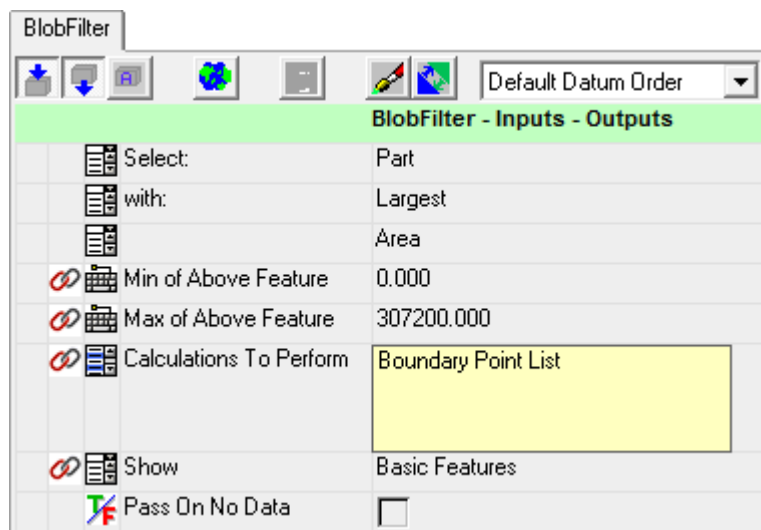
Theory of Operation

The input to the BlobFilter is a Blob Tree from the output of a Blob Tool. BlobFilter locates a particular blob in the blob tree that satisfies a specific condition such as the largest or the smallest radius. By default, Blob Tool has a BlobFilter component step. You can add or delete BlobFilter steps as necessary.

Description

BlobFilter allows the editing through the BlobFilter properties page, as shown in Figure 7–6.

FIGURE 7–6. BlobFilter Properties Page



Settings

- **Filter Parts based on Num Holes** — Specifies whether the filtering process should be applied to parts with a certain number of holes. When enabled, the number of holes is specified by **Num of Holes In Part**. By default, all parts and holes are subject to the filtering process.
- **Num of Holes In Part** — Indicates the number of holes in conjunction with **Filter Parts based on Num Holes**. When **Filter Parts based on Num Holes** is disabled, this setting has no effect.

Default: 0

- **Select** — Specifies whether the filtering target blob is a Part (default) or a Hole.
- **with** — Specifies whether the filtering target blob should have Largest (default), Smallest, Left Median, or Right Median feature.

Note: If you have an even number of blobs:

Left Median corresponds to the smaller of the two median blobs.

Right Median corresponds to the larger of the two median blobs.

You can specify the feature to be one of the following:

- **Area** — Area of the blob excluding holes in the blob.
- **Total Area** — Area of the blob including holes in the blob.
- **Average Radius** — Average radius from the center of the area to the perimeter.
- **Perimeter Length** — Distance around the periphery of the blob.
- **Axis Ratio** — Ratio of the length of the minor axis to the length of the major axis.
- **Roundness** — A measure of roundness in the range from 0.0 to 1.0. For a circle, the roundness is 1.0.
- **Length** — Length of the smallest rectangle enclosing the blob and oriented along the major and minor axes.

- Width — Width of a rectangle circumscribing the blob and oriented along the major and minor axes.
- Horiz Extent — Length of the blob along the major axis.
- Vert Extent — Width of the blob perpendicular to the major axis.
- Gray Average — Average of all the pixels that make up the blob.
- **Min of Above Feature** — Range of selected features acceptable by the BlobFilter. When the filtered resultant feature falls into this range (inclusive), the blob tree passes. When the filtered resultant feature falls outside this range (exclusive), the blob tree fails.

Default: 0.0

- **Max of Above Feature** — Range of selected features acceptable by the BlobFilter. When the filtered resultant feature falls into this range (inclusive), the blob tree passes. When the filtered resultant feature falls outside this range (inclusive), the blob tree fails.

Default: 307200.0

- **Calculations to Perform** — “Calculations to Perform” must first be enabled as shown in Figure 7–6 for point list functionality. The point list is an ordered list of boundary point [x,y] coordinate pairs following the perimeter of the filtered blob, starting from the topmost point on the boundary and then following the perimeter counterclockwise.
- **Show** — Controls the amount of information to be graphically displayed in the Blob Tool’s input buffer:
 - None — The filtered blob is not graphically indicated at all.
 - Basic Features (Default) — Highlight the filtered blob with a green boundary and a cross at the geometric center of the blob.
 - Basic+Points Features — In addition to the green boundary and cross as in the Basic Features, four points (left, top, right, and bottom), are also highlighted. These are the left-most, highest, right-most, and lowest points on the boundary, respectively.
 - All Features — In addition to the above features, the major and minor axis lines are also displayed.

- **Create Calibration Dot Datum** — Creates a calibration dot datum, Pixel CalDot Positions. This fills all the parts in the blob tree.
Default: Disabled
- **Pass On No Data** — When the blob tree is empty, the runtime status of the BlobFilter will be pass or fail depending on the setting of this property.

Training

None.

Results

Table 7–2 describes a set of individual features.

TABLE 7–2. Features

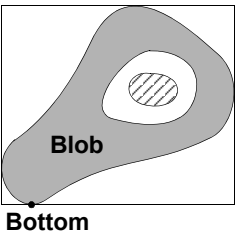
Feature	Description	Measurement Type
Area to Box Ratio	Ratio of blob area to its enclosing box area.	Double
Area to ROI Ratio	Ratio of area to area of search region (ROI).	Double
Average Radius	Average of all radii from the center of the area to all perimeter points of the blob. Blob Tool's setting, <i>Calculate Perimeter Features</i> , should be enabled to calculate the radius features.	Distance
Axes Ratio	Ratio of minor axis length to major axis length.	Double
Bottom Point	The lowest point on the boundary. 	Point
Box Area	Area of the box that exactly encloses the blob.	Area
Center Point	The geometric center of the blob.	Point
Center to Origin Distance	Distance between center point to buffer coordinate origin (usually top-left corner of buffer).	Distance
EnclRect	The smallest rectangle that encloses the blob.	Rect
Exclude-Holes Area	Count of pixels excluding holes.	Area
Hole Ratio	Ratio of hole area to total area (≤ 1.0).	Double
Holes Area	Hole area.	Area

TABLE 7-2. Features (continued)

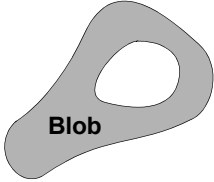
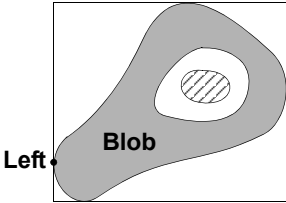
Feature	Description	Measurement Type
Include Holes-Area	Count of pixels including holes. 	Area
Left Point	The left-most point on the boundary. 	Point
Length Ratio	Ratio of width to length.	Double

TABLE 7-2. Features (continued)

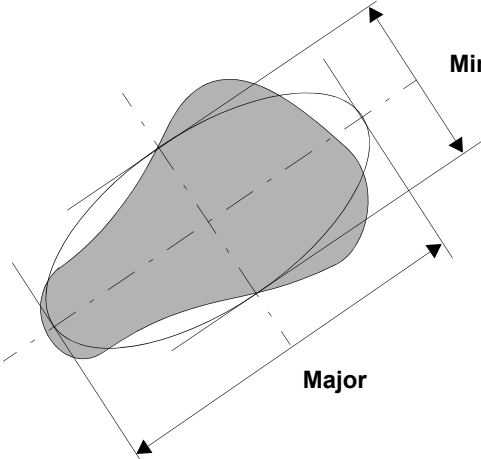
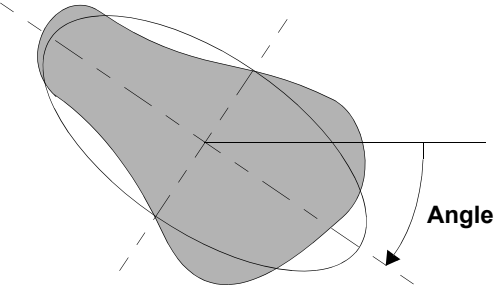
Feature	Description	Measurement Type
Length-Ellips, Width-Ellips	<p>The length and width of the major and minor axes of an equivalent ellipse, e.g., an ellipse that has the same second moments of area as the blob. Blob's processing property Calculate symmetry features, Compute first moments, and Compute second moments should be enabled.</p> 	Distance
Length-Y Perimeter	Width of perimeter along y direction in pixels.	Distance
Major Axis Angle	<p>The angle (in radians) between the major axis of the blob and the x-axis of the original coordinate system, which is a left-handed coordinate system.</p> 	Angle

TABLE 7-2. Features (continued)

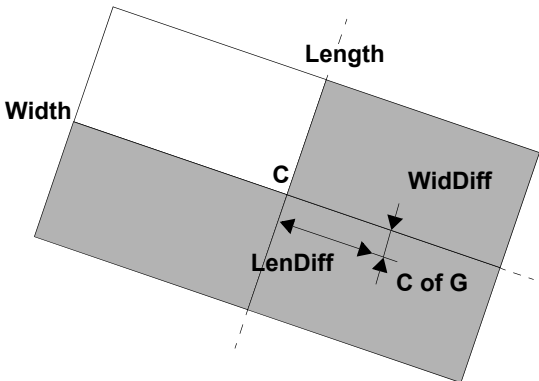
Feature	Description	Measurement Type
Major Axis Asymmetry, Minor Axis Asymmetry	<p>Measure the asymmetry of an object along its major and minor axes. These features enable you to:</p> <ul style="list-style-type: none"> — Determine the orientation angle of the object in the full range of -180° to $+180^\circ$ — Distinguish between right-handed and left-handed objects <p>The two asymmetry measures are the displacements of the center of the rotated enclosing rectangle with respect to the object's center of area, along the major and minor axes. The rotated enclosing rectangle is the smallest rectangle enclosing the blob and is oriented along the major and minor axes of the blob.</p>  <p>Blob Tool's setting, <i>Calculate Symmetry Features</i>, should be enabled.</p>	Distance
Major Axis Line	Major axis line represented by three parameters: A, B, and C that satisfy $Ax + By + C = 0$ in the original coordinate system.	Line

TABLE 7-2. Features (continued)

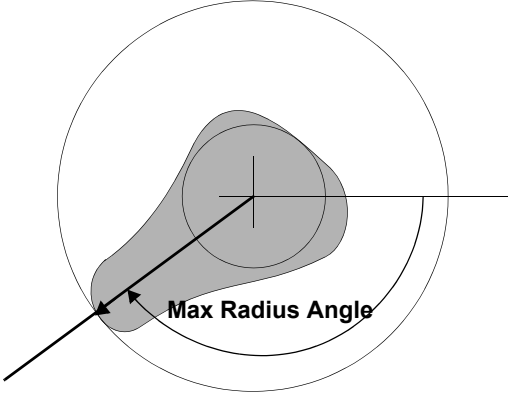
Feature	Description	Measurement Type
Max Radius Angle	<p>The Max Radius Angle is the angle (in radians) between the maximum-radius line of the blob and the x-axis of the original coordinate system. The angles are measured with the left-handed rule.</p> 	Angle
Max/Min Angle Range	Angle range in radians ($\pm \pi$).	Angle
Maximum Radius	Radius of the circle that encloses the blob and intersects with the blob at only one perimeter point. Blob Tool's setting, <i>Calculate Perimeter Features</i> , should be enabled to calculate the radius features.	Distance

TABLE 7-2. Features (continued)

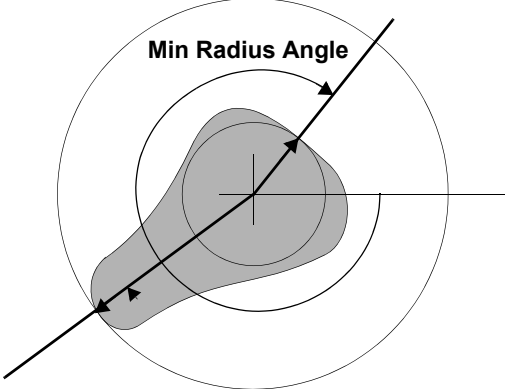
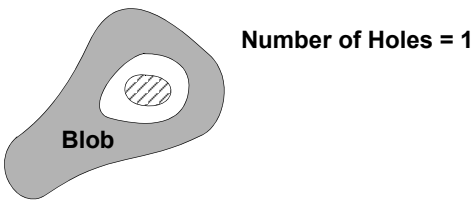
Feature	Description	Measurement Type
Min Radius Angle	<p>The Min Radius Angle is the angle (in radians) between the minimum-radius line and the x-axis of the original coordinate system. The angles are measured with the left-handed rule.</p> 	Angle
Minimum Radius	<p>Radius of the circle that is enclosed by the blob and intersects with the blob at only one perimeter point. Blob Tool's setting, <i>Calculate Perimeter Features</i>, should be enabled to calculate the radius features.</p>	Distance
Minor Axis Line	<p>Minor axis line represented by three parameters: A, B, and C that satisfy $Ax + By + C = 0$ in the original coordinate system.</p>	Line
Number of Holes	<p>The number of holes in the blob.</p> 	Int
Perimeter	<p>Blob perimeter length. Blob Tool's processing property Calculate Perimeter Features should be enabled.</p>	Distance

TABLE 7-2. Features (continued)

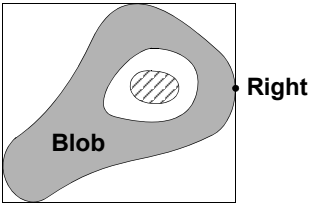
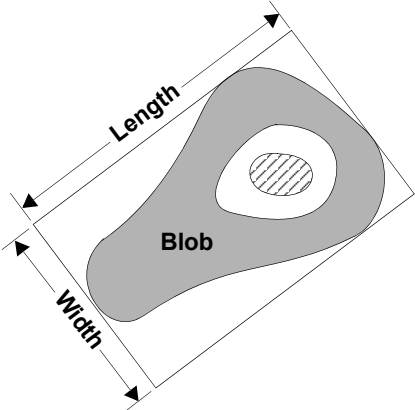
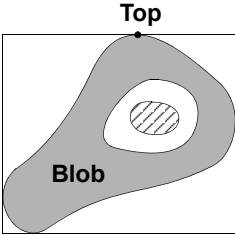
Feature	Description	Measurement Type
Perimeter Invariant	Calculated as perimeter squared over total area.	Double
Radius Ratio	Ratio of minimum radius to maximum radius.	Double
Right Point	The right-most point on the boundary. 	Point
Rotated Length, Rotated Width	Length and width of the rectangular box enclosing the blob, and oriented along the major and minor axes. Blob's processing properties: Calculate symmetry features, Compute first moments, and Compute second moments, should be enabled. 	Distance

TABLE 7-2. Features (continued)

Feature	Description	Measurement Type
Roundness	Measure of roundness calculated as $(4 \text{ Pi Area})/\text{Perimeter}^2$. (For a circle, roundness is 1.0).	Double
Top Point	The highest point on the boundary.  A diagram showing a gray irregular blob with a white circular hole in the center. The hole has diagonal hatching. A black dot marks the highest point on the top boundary of the blob, labeled 'Top'. The word 'Blob' is written in bold black text inside the gray area.	Point
Unrotated Length, Unrotated Width	Length and width of the rectangular box circumscribing the blob and oriented along the x and y axes.	Distance
Width-X Perimeter	Length of perimeter along x direction in pixels.	Distance

I/O Summary

The BlobFilter provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Select ppp (among all parts) with zzz uuu Min/Max: nnn/xxx

Where: ppp = Either Part or Hole
 zzz = Either Largest or Smallest
 uuu = One of the following: Area, Total Area, Average Radius, Perimeter Length, Axis Ratio, Roundness, Rotated Length, Rotated Width, Horizontal Extent, Vertical Extent
 nnn = Minimum feature size
 xxx = Maximum feature size

Outputs: Center: [x,y] Area: aaa Holes: n Roundness: rrr, YYY

Fail to find the required blob

Where: [x,y] = Coordinate of the found blob
n = Number of holes in the blob found
aaa = Area in pixel count excluding holes
rrr = Roundness of the bound blob
YYY = Selected features other than Area and Roundness.

Possible selections: Total Area, Average Radius,
Perimeter Length, Axis Ratio, Rotated Length,
Rotated Width, Horiz Extent, or Vert Extent

Circle Analysis

Circle Find

This step finds the best circle (x,y,r) in the input buffer matching the specified parameters.

Other Steps Used

CustomVision Tool or Custom Step — Configured for Circle Find.

Theory of Operation

The Circle Find is performed in one of two ways:

- When **Nominal Radius** is specified, the Circle Find projects along the gradient vector at each data point and generates a possible circle center point.
- When **Nominal Radius** is set to 0.0 (the default), the Circle Find pairs the data points that are approximately normal to each other and intersects the gradient vectors to generate a possible center point. It ensures that the distances from this center point to each of the data points are approximately equal.

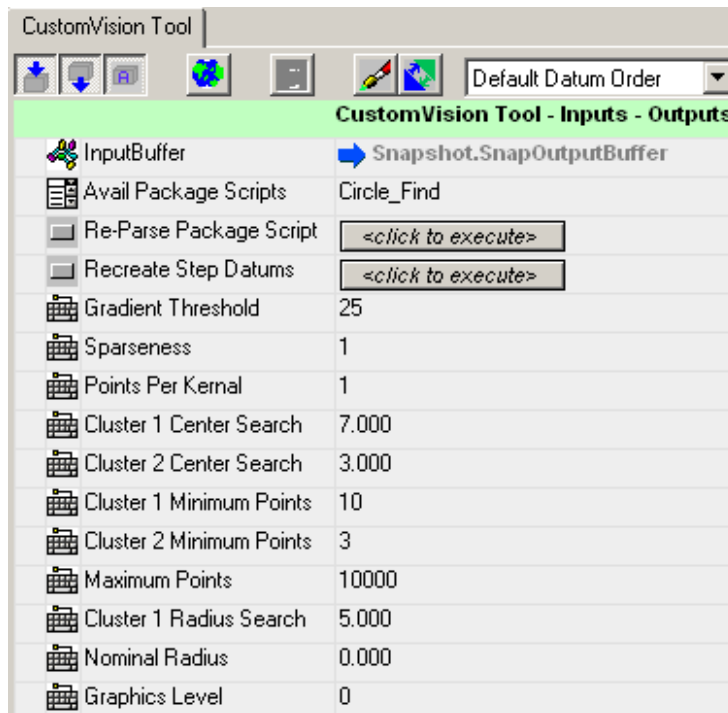
After generating a group of possible center points, the center positions are clustered to search for good circles. Each center point cluster becomes a starting point for the next step in the algorithm.

For each center cluster, data points are collected within a donut around the median radius. These points are split into sets, and a circle is fit through each set of data points.

The more accurate center positions obtained from these circles are clustered to obtain a final circle.

Description

The CustomVision Tool properties page pertaining to Circle Find is shown in Figure 7–7.

FIGURE 7-7. CustomVision Tool Properties Page — Circle Find**TABLE 7-3. Links to Property Descriptions**

For Information About...	Go To...
Avail Package Scripts	page 7-28
Cluster 1 Center Search	page 7-29
Cluster 1 Minimum Points	page 7-29
Cluster 1 Radius Search	page 7-29
Cluster 2 Center Search	page 7-29
Cluster 2 Minimum Points	page 7-29
Gradient Threshold	page 7-28
Graphics Level	page 7-30
Maximum Points	page 7-29
Nominal Radius	page 7-30

TABLE 7-3. Links to Property Descriptions (continued)

For Information About...	Go To...
Points Per Kernal	page 7-29
Recreate Step Datums	page 7-28
Re-Parse Package Script	page 7-28
Sparseness	page 7-29

Settings

The CustomVision Tool properties page, pertaining to Circle Find, allows the modification of the following properties:

- **Avail Package Scripts** — Selects the appropriate Perl script package. The script package that implements the functionality of this step is Circle_Find.
- **Re-Parse Package Script** — This button causes the package script to be parsed when clicked. Whenever any changes to the script are made, this button needs to be clicked to make these changes take effect.
- **Recreate Step Datums** — This button causes the input and output datum lists in the step to be recreated. This button only needs to be clicked when a datum is either added, removed or changed in the script. If the script is changed, but no input or output datums are changed, then this button does not need to be clicked. Clicking this button also causes all input datums to be set to their default values and lose their connections to other step results or parameters.

Datums created by the package script are added to the user interface. Input datums are shown as a box with a drop-down list button. The input datums can be linked to other similar type datums in the Job. Clicking the drop-down list button causes the Job Tree to be displayed, allowing you to select the datum to link to the input datum. Resource datums are shown as user-editable boxes that can be set to a value directly. Output datums are not shown in the user interface for this step, but can be seen in the Job Tree that comes up when linking an input datum.

- **Gradient Threshold** — Refers to the minimum magnitude of gradient required to acquire a data point. This gradient is equivalent to the sum of the X and Y gradients at that point — $ABS(Gx)+ABS(Gy)$.

Default 25

Range: 1 to 255

- **Sparseness** — Refers to the distance in pixels that determines the size of the kernel. The kernel varies \pm sparseness from the center pixel:
 - Sparseness of 1 yields 3x3 kernel (every pixel sampled) (default)
 - Sparseness of 2 yields 5x5 kernel (every other pixel sampled)
 - Sparseness of 3 yields 7x7 kernel (every third pixel sampled)
- **Points Per Kernel** — Refers to how many data points are used per kernel position. The sparseness determines how many gradient data points may be taken at each kernel position. The maximum number is equal to twice the sparseness. When the maximum possible data points are taken, there should be a data point for every pixel.

Default: 1

Range: 1 to 6

- **Cluster 1 Center Search** — Refers to the size of the ROI where center points resulting from the initial pass may be collected into a possible circle.

Default: 7.0

Minimum Value: 0.0

- **Cluster 2 Center Search** — Refers to the size of the ROI where center points from the best-fit circles may be collected into a final circle.

Default: 3.0

Minimum Value: 0.0

- **Cluster 1 Minimum Points** — Refers to the minimum number of initial data points required to make a possible circle.

Default: 10

Minimum Value: 1

- **Cluster 2 Minimum Points** — Refers to the minimum number of fitted circles required to make a final circle.

Default: 3.0

Minimum Value: 1

- **Maximum Points** — Refers to the maximum number of data points that may be included in a circle.

Default: 10000

Minimum Value: 1

- **Cluster 1 Radius Search** — Refers to the width of the donut that collects data points for the fine search.

Default: 5.0

Minimum Value: 0.0

- **Nominal Radius** — Refers to the nominal radius of the desired circle. When specified (non-zero), it generates the initial batch of possible center points. This number should be negative for a dark circle or a light background.

Default: 0.0

- **Graphics Level** — Any combination of the following bit patterns may be used:
 - 0x00000000 — None
 - 0x00000001 — Show best circle only
 - 0x00000010 — Show all circles found
 - 0x00000100 — Show all gradients found

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.
- **Center Point of Best Circle** — The (x,y) position of the best circle found (a point datum).
- **Radius of Best Circle** — The radius of the best circle found (a distance datum).

Edge Analysis

Edge Tool

This tool finds straight, circular, or elliptical edges, or line segments in the ROI. Edge and line features include line equation, a midpoint of the line, and angle.

The Edge Tool searches the ROI for places where gray level changes occur and uses these places to compute a line, circle, or ellipse. These transition places are found using a threshold or gradient analysis.

Edge extracts edge points from an image. Edge points are pixel locations in the image where there is a large change in grayscale value around its location. The inputs to Edge are the buffer that contains the image, the ROI that determines which part of the image is processed, and the direction this area is scanned. The output of the Edge is a list of points found.

Other Steps Used

Rect Warp — Used internally to Edge. Rect Warp searches for edges in different directions and at different rotations.

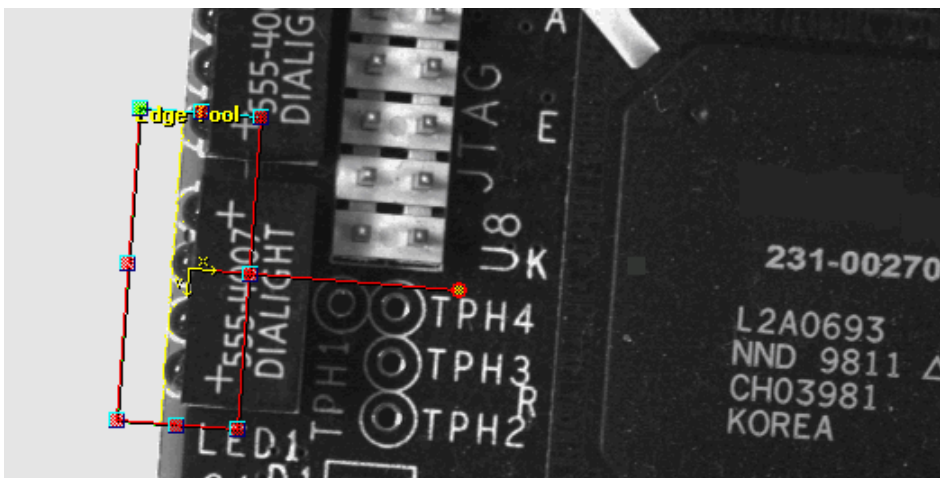
Theory of Operation

The image is scanned for edge points as determined by the size and orientation of the ROI. Each row of the ROI (scan line) is scanned left to right. The rows are scanned from top to bottom. Edge creates a Rect Warp internally that is required for Edge to operate.

The ROI, which defines the pixels to scan, can be adjusted by moving, sizing and rotating the search area associated with Edge. The default size of the ROI is 150x100 with no rotation.

In Figure 7–8, the ROI, shown in red, is scanned for the first positive edge on each scan line. The scan direction, as defined by the top of the ROI, is approximately 280° clockwise from vertical running along the ROI rotation handle.

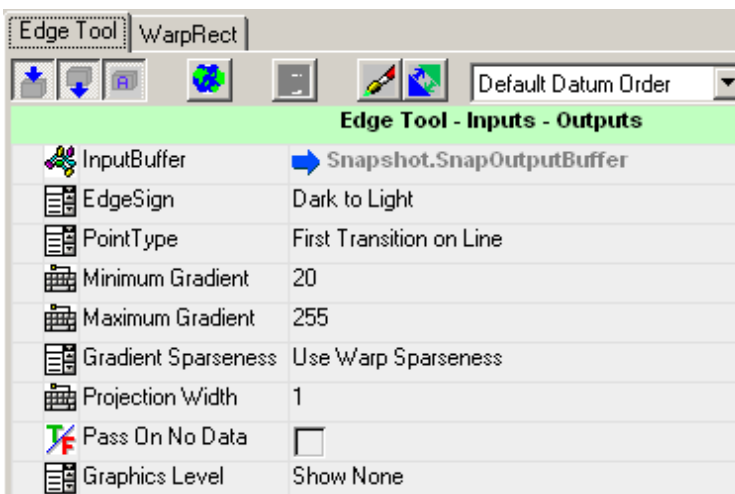
FIGURE 7-8. Locating an Edge



Description

Edge Tool allows editing through the Edge Tool properties page, as shown in Figure 7-9.

FIGURE 7-9. Edge Tool Properties Page



Settings

- **EdgeSign** — Selects the polarity of detected edges:
 - Any — Directs the step to ignore the sign of the gradients.
 - Dark to Light (Default) — Directs the step to find only edge points with the specified dark to light gradient polarity.
 - Light to Dark — Directs the step to find only edge points with the specified light to dark gradient polarity.
- **PointType** — Selects which edge points are returned by the step. Only one edge point is returned on each scan line:
 - First Transition on Line (Default) — Directs the step to find the first edge point found on each scan line.
 - Maximum Transitions on Line — Directs the step to find the point on each scan line at which the gradient is the largest.
- **Minimum Gradient** — Sets the minimum gradient value that the gradient must exceed for a point to qualify as an edge.
Default: 20
- **Maximum Gradient** — Sets the maximum gradient value that the gradient cannot exceed for a point to qualify as an edge.
Default: 255
- **Gradient Sparseness** — Tunes the gradient detection based on weak or strong edge strength in the image:
 - Sharp Edges (Default) — Takes the gradient at every point.
 - Weak Edges — Takes the gradient at 60% of the points in the x-direction and all points in the y-direction.
 - Use Warp Sparseness — Takes the gradient at the percentage of points in the x and y directions, based on the internal RectWarp properties page settings for X ScaleFactor and Y ScaleFactor.

- **Projection Width** — Before gradient detection, scan lines may be grouped together or binned by performing a projection. This produces a single scan line for each group that is the average of the group. The gradient detection becomes more resistant to clutter as these points are averaged away by the projection. Refer to “Projection Tool” on page 12-13 for more information. To scan the ROI as a single fat vector, the projection width should be set to the ROI height.

Default: 1, which denotes no binning

- **Pass On No Data** — Allows the step to pass when no edge points are found.

Default: Disabled

- **Graphics Level** — Enables various graphics options at runtime:
 - **Show None (Default)** — No graphics are displayed.
 - **Show ROI Only** — Draws the ROI only. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red.
 - **Show Basic Graphics** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The outlines of all edge points are drawn in yellow.
 - **Show Details** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The outlines of all edge points are drawn in yellow.
 - **Show Details and Mask** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The outlines of all edge points are drawn in yellow. The mask pixels are drawn in red.

Training

No special training is required for Edge Tool other than setting the size and position of the input ROI.

Results

The results from the Edge Tool are listed in Table 7–4. The main result is the Point List, which contains the edge points found. When Graphics Level is set to Show Details, the edge points are drawn. As with all steps, a pass/fail status is also output from the step.

TABLE 7–4. Results Datums

Symbolic Name	Datum Type	Description
AvgGrad	Double	Average gradient of the edge points found
AvgInt	Double	Average intensity of the edge points found
NPoints	Integer	Number of edge points found
PtList	PointList	Edge points found

I/O Summary

Edge Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: MinGrad: aaa, MaxGrad: bbb, Trans: ccc (ddd), Sparse ± eee,
Project Hgt: fff

Outputs: Trans: ggg, Avg Grad: hhh, Avg Intens: iii

Where: aaa = Minimum gradient value between 0 and 255
 bbb = Maximum gradient value between 0 and 255
 ccc = First or max
 ddd = Dark -> Light or Light -> Dark
 eee = 1 for strong edges, 2 for weak edges
 fff = Projection height
 ggg = Value between 0 - 9999
 hhh = Average gradient value between 0 and 255
 iii = Average intensity value

Fast Edge Tool

This tool finds straight edges or line segments in the ROI. The step calculates a line equation, a midpoint of the line and angle, as well as statistics for the edges found as they relate to the line that is fit.

The Fast Edge Tool searches the ROI for places where gray level changes occur and uses these places to compute a line based on one of three types of line fit algorithms. These transition places are found using a threshold or gradient analysis.

The edge points extracted from the image by the Fast Edge Tool are pixel locations in the image where there is a large change in grayscale value around its location. The inputs to Fast Edge Step are the buffer that contains the image, the ROI that determines which part of the image is processed, the direction in which this area is scanned, and the line fit algorithm parameters. The outputs of the Fast Edge Step include a line equation and midpoint, the number of points found, the number of points used in the line fit, the fit error, the furthest points from the line, and the statistical deviations of all points from the fit line.

Other Steps Used

Fast Edge Group — Multiple Fast Edge Tools can be inserted into a Fast Edge Group for faster runtimes. By grouping the Edge tools in this way, the edge processing can be pipelined better. The step is responsible for running the gradient analyses of all of its children Fast Edge Steps.

Theory of Operation

The image is scanned for edge points as determined by the size of the ROI and the direction you specified. The ROI, which defines the pixels to scan, can be adjusted by moving and sizing the search area associated with the Fast Edge Tool. The default size of the ROI is 150x100. Once all edge points are found, a line is fit using one of three algorithms: a least squares fit, robust classic line fit, or robust line fit.

Description

Fast Edge Tool allows editing through the Fast Edge Tool properties page, shown in Figure 7–10.

FIGURE 7-10. Fast Edge Tool Properties Page

Blade1_fe

Default Datum Order

Blade1_fe - Inputs - Outputs

InputBuffer	Snapshot GEOM.SnapOutputBuffer
Scan Direction	Top to Bottom
Transition color	Dark to Light
Scan Mode	Line
Scan Option	First edge
Transition Type	Max Gradient
Gradient Threshold	50
Minimum Points	5
Minimum Line Width	3.000
Maximum Line Width	16.000
Sharpness	+/- 1 Very Sharp
Row Scan Mode	Process every Nth Row
N Rows (Row Scan Mode)	1
Average Rows(+/-)	0
Line Fit	No Line Fit
Straightness	+/- 0.75 Straight
Minimum % of Points on Line	0.300
Graphics Level	ShowDetails
Calculations To Perform	Edge Point List
Pass On No Data	<input type="checkbox"/>

FIGURE 7–11. Example of Fast Edge Tool with No Line Fit, No Averaging

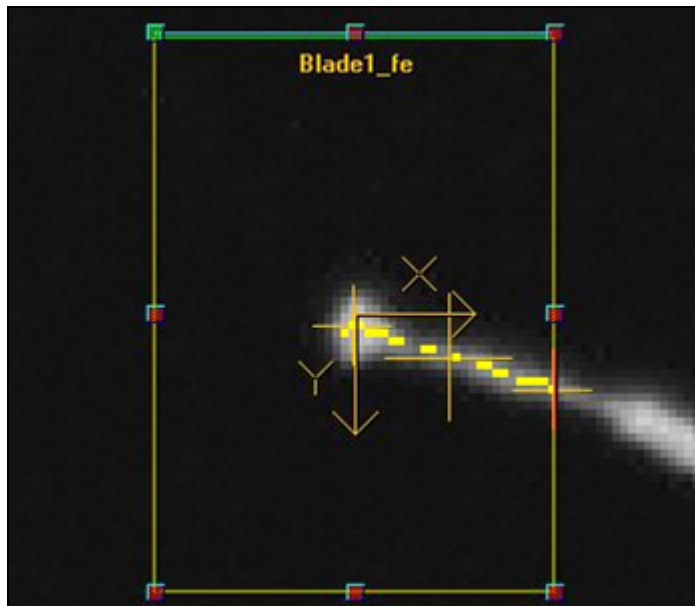
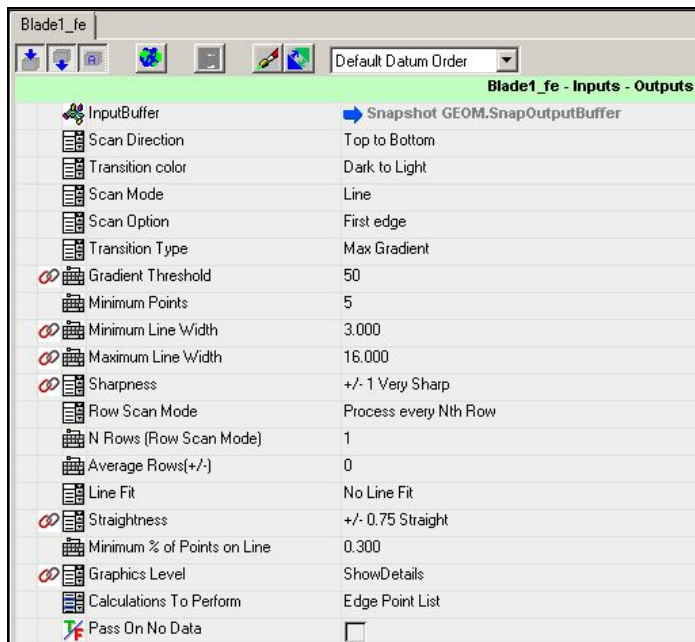
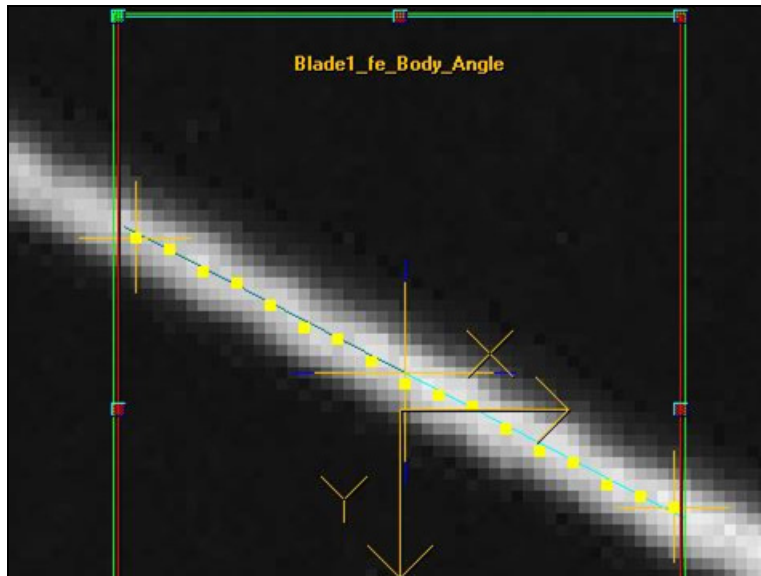
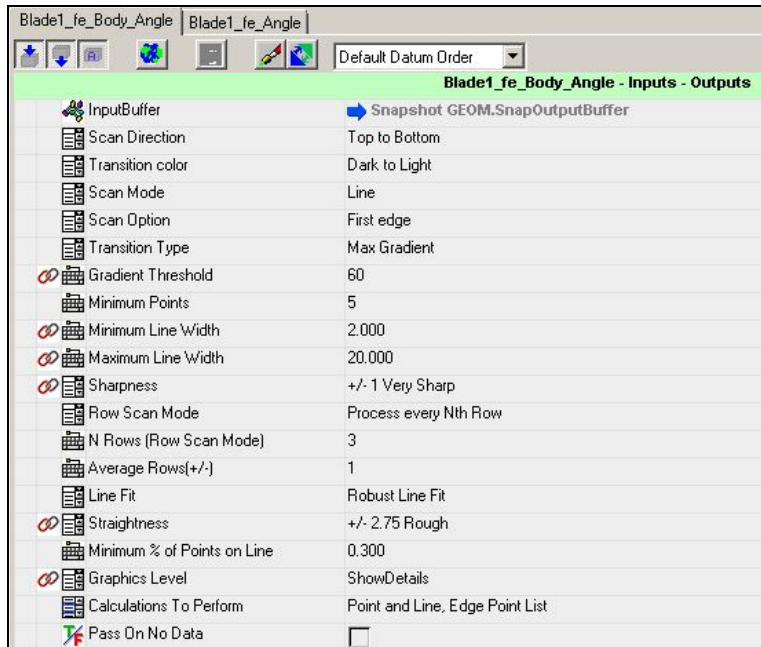


FIGURE 7–12. Example of Fast Edge Tool with Robust Line Fit and Row Averaging

Settings

- **Scan Direction** — Selects in which direction the ROI is scanned for edges:
 - Left to Right (Default)
 - Right to Left
 - Top to Bottom
 - Bottom to Top
- **Transition Color** — Selects the polarity of detected edges:
 - Absolute Gradient — Ignores the sign of the gradients.
 - Dark to Light — Finds only edge points with the specified dark to light gradient polarity.
 - Light to Dark (Default) — Finds only edge points with the specified light to dark gradient polarity.
- **Scan Mode** — Gives you a choice between Edge Line.
- **Scan Option** — Selects which edge along the scan direction should be chosen as the contributing point to the line at each point across the scanned pixels:
 - Best Edge — Selects the edge with the strongest gradient.
 - First Edge (Default) — Selects the first edge found.
 - Last Edge — Selects the last edge found.
- **Transition Type** — Selects the algorithm to be used when finding edge points:
 - Max Gradient (Default) — The edge point is determined by finding a local maximum gradient along the scan direction. When Scan Option is set to Best Edge, the point with the highest gradient is returned, otherwise the First or Last point with the highest gradient compared to its neighboring gradients is returned. The gradient must be greater than the Gradient Threshold to be considered an edge.

- **Gradient Crossing** — The edge point is determined by finding the point at which the gradient along the scan direction crosses the Gradient Threshold. This is used when looking for a gradient (first derivative of the gray values) with a specific value.
- **Gradient Threshold** — Sets the minimum gradient value that the gradient must exceed for a point to qualify as an edge.

Default: 20

Range: 0 to 255

- **Minimum Points** — Specifies the minimum number of edge points that must be found before attempting to fit a line to the points.

Default: 5

Range: 3 to 2147483647

- **Minimum Line Width** — Measured in pixels in the direction of the scan.
- **Maximum Line Width** — Measured in pixels in the direction of the scan.
- **Sharpness** — Tunes the gradient detection based on blurred or sharp edges in the image. The number of pixels over which the gradient is computed changes based on this setting:
 - ± 1 Very Sharp (Default)
 - ± 2 Sharp
 - ± 3 Blurred
 - ± 4 Very Blurred
- **Row Scan Mode** — Together with **N Rows**, this property species how many rows of pixels will be scanned for edge transitions. The default setting of “Process Every Nth Row,” combined with the default value of 1 for **N Rows**, means process every row. Use this parameter to speed up the Fast Edge tool by cutting down on the amount of processing it needs to do:
 - **Process every Nth Row** — If you select Process every Nth Row and set N Rows to 2, the Fast Edge tool processes every other row of pixels.

If you select Process every Nth Row and set N Rows to 3, the Fast Edge tool processes every third row, and so on.

- Process exactly N Rows — If you select Process exactly N Rows, the Fast Edge tool processes exactly the number of rows specified in N Rows. For example, if N Rows is 20, then the Fast Edge tool will process exactly 20 rows of pixels, regardless of the size of the ROI.
- N Rows (Row Scan Mode) — Specifies the number of rows that will be processed, based on the setting in Row Scan Mode.

Default: 1

Range: 1 to 100

- Average Rows (+/–) — Performs row averaging.
 - If set to 0, no row averaging is performed.
 - If set to 1, lines are grouped into threes (the center line and the ones on either side).
 - If set to 2, lines are grouped into fives.
- Line Fit — Provides choices for the type of line fit performed.

Note: If only points are required, set Line fit to “No Line Fit” to save processing time.

- Average Horizontal Line
 - Average Vertical Line
 - Least Squares Line Fit
 - No Line Fit
 - Robust Classic Line Fit
 - Robust Line Fit (Default)
- Straightness — Specifies how straight the robust line is. If a line matching this straightness parameter cannot be fit given the points found, the step fails:
 - ± 0.10 Extra Straight
 - ± 0.25 Very Straight
 - ± 0.75 Straight (Default)

- ± 2.75 Rough
- ± 7.75 Very Rough
- **Minimum % of Points on Line** — This property only applies when you have **Line Fit** set to **Robust Line Fit**. When the Robust Line Fit algorithm has calculated its best line, it will attempt to verify that it has calculated a good line by running through all of the edge points, and looking to see how many of the points are actually on the line (as specified by **Straightness**). The final % of points on the line must be larger than this value in order for the line to be accepted.

Range: 0.05 to 1

- **Graphics Level** — Enables various graphics options at runtime:
 - **Show None** — No graphics are displayed.
 - **Show ROI Only** — Draws the ROI only. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red.
 - **Show Basic Graphics (Default)** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The line that was fit to the points is also drawn in green and its midpoint is drawn in blue.
 - **Show Details** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The line that was fit to the points is also drawn in green and its midpoint is drawn in blue. All qualifying edges in the ROI are drawn in yellow. A blue cross is drawn at the edge furthest from the line on either side of the line, and a yellow line perpendicular to the line is drawn from each of these points.
- **Calculations to Perform** — Allows selection from the following list of additional calculations to be done at runtime. When any of these calculations is done, the tool will execute more slowly. By default, all of these calculations are enabled.

Note: If a line is not required, do not select “Point and Line”.

- **Point and Line** — Generates the midpoint and line equation for the edge found.

- Points Used in Fit — Generates the number of edge points found that were used in the line fit.
- Mean, Min, Max, Abs Deviations — Generates this group of deviations based on the edge point distances from the line.
- Furthest Points — Finds the points furthest from the line on both sides of the line.
- RMS Deviation — Computes the RMS deviation based on the edge point distances from the line.
- Edge Point List — Generates a list of all edge points found.
- Pass On No Data — Determines whether the tool should pass or fail (default) when no edges are found.

Training

No special training is required for Fast Edge Tool other than setting the size and position of the input ROI.

Results

The results from the Fast Edge Tool are listed in Table 7–5. The main result is the Line that was fit to the edge points found. When **Graphics Level** is set to Show, Details, the edge points are drawn. As with all steps, a pass/fail status is also output from the step.

TABLE 7–5. Results Datums

Symbolic Name	Datum Type	Description
EdgeAngle	Angle	Angle of the fitted line
EdgeLine	Line	Line fit to the points found
EdgePt	Point	Midpoint of the line found
FitErr	Double	Line Fit Error
MaxAbs	Double	The maximum distance from the line (either side)
MaxDev	Double	The largest distance of an edge above the line
MaxFitAbs	Distance	Fit point furthest from the line
MaxFitDev	Distance	The largest distance of an edge in the line fitting set above the line

TABLE 7-5. Results Datums (continued)

Symbolic Name	Datum Type	Description
MaxFitPt	Point	The edge point in the line fit set furthest away above the line
MaxPt	Point	The edge point furthest away above the line
MeanDev	Double	The average deviation of the edge pts from the line
MinDev	Double	The largest distance of an edge below the line
MinFitDev	Distance	The largest distance of an edge in the line fitting set below the line
MinFitPt	Point	The edge point in the line fit set furthest away below the line
MinPt	Point	The edge point furthest away below the line
NPoints	Int	The number of total edges found in the ROI
PtList	PointList	The edge points found
RMS	Int	The RMS error of the points from the line
UsesPts	Integer	Number of edge points found that were used in the fit

I/O Summary

Fast Edge Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: GradThr: ttt, ScanDir: ddd, Trans: nnn (ppp), Sharp: \pm sss, Fit: fff, Straightness: zzz

Outputs: Line: A=aaa, B=bbb, C=ccc PtsFound: xxx, PtsUsed: yyy

Where: aaa, bbb, ccc = The coefficients of the line fit to the points found
 ddd = Left->Rt, Rt->Left, Top->Bot, or Bot->Top
 fff = None, RobCls, Robust, or LstSq
 nnn = First, Best, or Last
 ppp = Dk->Lt, Lt->Dk, or AbsGrad
 sss = 1, 2, 3, or 4
 ttt = Gradient value between 0 and 255
 xxx = Number of edges found in the ROI
 yyy = Number of points used in the line fit
 zzz = Very Straight, Straight, Rough, or Very Rough

Flaw Analysis

Flaw Tool

This tool detects the presence or absence of a feature. For example, it can determine whether a package has any defects in appearance, such as cracks or scratches. The input to the Flaw Tool is an image buffer on which to process. The output of the step at runtime is the pass/fail status of the inspection. An output containing the resulting sum or count is also available.

Other Steps Used

None.

Theory of Operation

The following methods can perform flaw inspection:

- **Count Edge Pixels** — Calculates the number of qualifying edge pixels within the ROI.
- **Count Pixels Within Range of Intensities** — Calculates the number of qualifying pixels within the ROI.
- **Sum All Intensity Values** — Calculates the sum of all pixel values in the ROI.
- **Sum All Edge Energy Values** — Calculates the sum of the edge energy values for all qualifying edge pixels within the ROI.

For all methods, the value calculated is compared to a set of user-selectable limits to determine the pass/fail status of the tool. The ROI within which the Flaw Tool operates can be adjusted by moving and sizing the search area shape associated with the Flaw Tool. The Flaw Tool's minimum size is 9x9.

Description

Flaw Tool allows editing through the Flaw Tool properties pages, as shown Figure 7-13 and Figure 7-14.

FIGURE 7-13. FlawTool Properties Pages

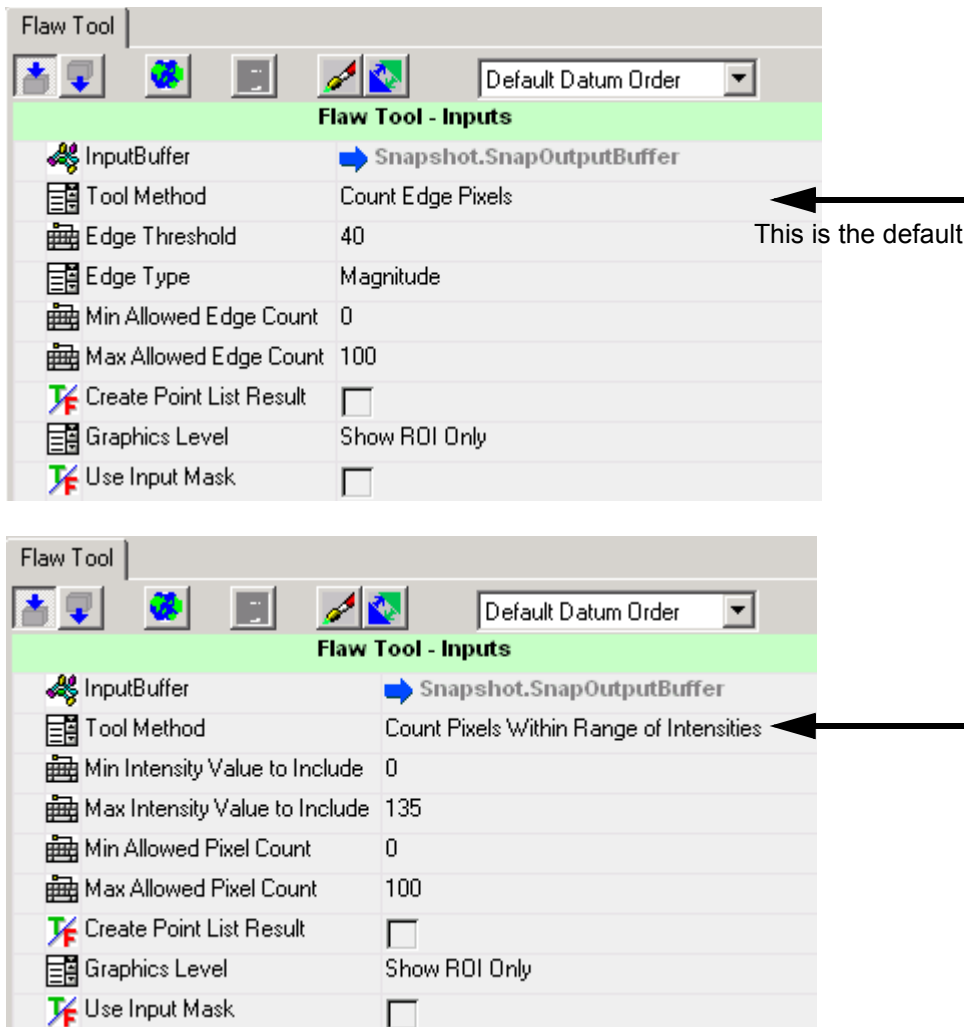
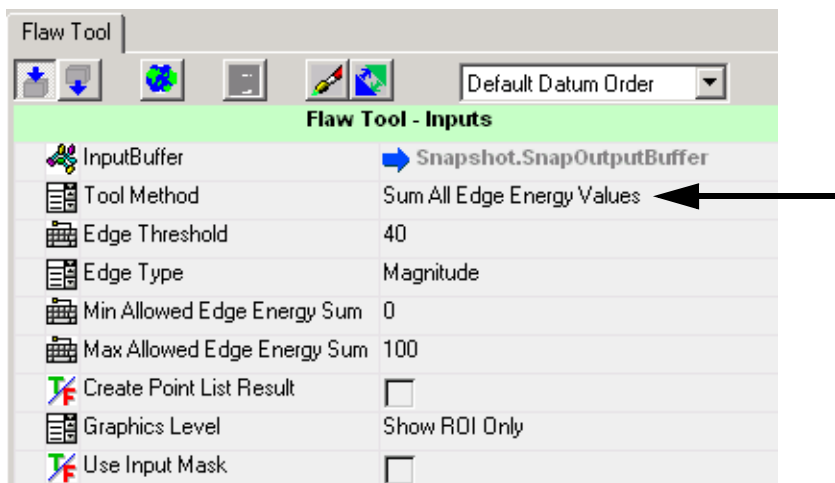
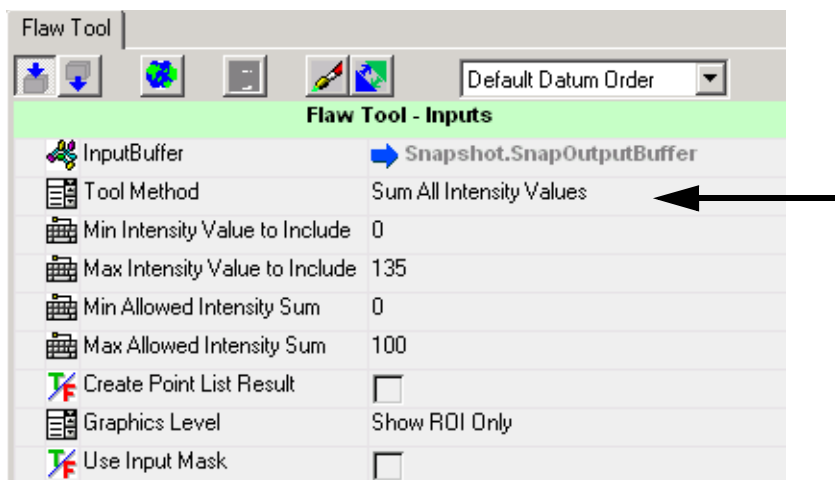


FIGURE 7-14. FlawTool Properties Pages



Settings

- **Tool Method** — Allows the selection of the type of inspection to perform. The FlawTool can be programmed to perform one of the following methods:
 - **Count Edge Pixels (Default)** — For each pixel in the ROI, if the edge energy value is greater than or equal to a user-specified threshold, that pixel is included in the count. When the total count of qualifying pixels is within a user-specified range of values, the inspection passes. When the total count of qualifying pixels is outside the user-specified range of values, the inspection fails.
 - **Count Pixels Within Range of Intensities** — For each pixel in the ROI, if the intensity value is greater than or equal to the user-specified lower threshold and the intensity value is less than or equal to the user-specified upper threshold, that pixel is included in the count. When the total count of qualifying pixels is within a user-specified range of values, the inspection passes. When the total count of qualifying pixels is outside the user-specified range of values, the inspection fails.
 - **Sum All Intensity Values** — For each pixel in the ROI, the gray level value at that pixel is added to the sum. When the total sum of gray level values is within a user-specified range of values, the inspection passes. When the total sum of gray level values is outside the user-specified range of values, the inspection fails.
 - **Sum All Edge Energy Values** — For each pixel in the ROI, the edge energy value is added to the sum. When the total sum of edge energy values is within a user-specified range of values, the inspection passes. When the total sum of edge energy values is outside the user-specified range of values, the inspection fails.
- **Edge Threshold** — Sets the minimum value that an edge energy value must be in order to consider that pixel to be an edge pixel.

Default: 40
Range: 3 to 255
- **Edge Type** — Selects one from this list to be the pixel value of the output image:
 - **Magnitude (Default)** — Magnitude of the gradient.
 - **Absolute Gradient-x** — Absolute value of the gradient along the x-axis.

- Absolute Gradient-y — Absolute value of the gradient along the y-axis.
- Sum of Absolute Gradients — Sum of Absolute Gradient-x and Absolute Gradient-y.
- **Min Allowed Edge Count** — Sets the minimum number of edge pixels that must be present in order for the inspection to pass. This property is only used when **Tool Method** is set to Count Edge Pixels.

Default: 0

Range: 0 through 1048576

- **Max Allowed Edge Count** — Sets the maximum number of edge pixels that may be present in order for the inspection to pass. This property is only used when **Tool Method** is set to Count Edge Pixels.

Default: 100

Range: 0 through 1048576

- **Min Intensity Value to Include** — Sets the minimum intensity value that a pixel must have in order to be included in the count.

Default: 135

Range: 0 - 255

- **Max Intensity Value to Include** — Sets the maximum intensity value that a pixel may have in order to be included in the count.

Default: 135

Range: 0 - 255

- **Min Allowed Pixel Count** — Sets the minimum number of qualified pixels that must be present in order for the inspection to pass.

Default: 0

Range 0 through 10478576

- **Max Allowed Pixel Count** — Sets the maximum number of qualified pixels that may be present in order for the inspection to pass.

Default: 100

Range: 0 through 10478576

- **Min Allowed Intensity Sum** — Sets the minimum sum of intensity values that must be calculated in order for the inspection to pass.

Default: 0

Range: 0 through 268435456

- **Max Allowed Intensity Sum** — Sets the maximum sum of intensity values that may be calculated in order for the inspection to pass.

Default: 100

Range: 0 through 268435456

- **Min Allowed Edge Energy Sum** — Sets the minimum sum of edge energy values that must be calculated in order for the inspection to pass.

Default: 0

Range: 0 through 268435456

- **Max Allowed Edge Energy Sum** — Sets the maximum sum of edge energy values that may be calculated in order for the inspection to pass.

Default: 100

Range: 0 through 268435456

- **Create Point List Result** — When checked, this property creates a PointList output datum that contains all of the pixels that were counted by the Flaw tool.
- **Graphics Level** — Allows the selection of the level of detailed graphics to be displayed when the Flaw Tool performs an inspection:
 - **Show Basic Graphics** — The count or sum resulting from the inspection is displayed in the center of the ROI along with the ROI graphics.
 - **Show Details** — When enabled and Tool Method is set to Count Edge Pixels or Count Pixels within Range of Intensities, each pixel that was included in the count is drawn in yellow. This selection also incorporates the same graphics as Show Basic Graphics.
 - **Show Details and Mask** — In addition to the graphics drawn with Show Details, the masked pixels (if any) will be drawn in red.
 - **Show None** — No graphics are displayed.
 - **Show ROI Only (Default)** — The box defining the ROI is drawn, green when the tool passes, and red when the tool fails.

- **Use Input Mask** — This property is applicable only when there is a mask associated with the Edge Tool. When such a child step is inserted into the Edge Tool, **Use Input Mask** is automatically enabled. When enabled, the Edge Tool must be trained. After it is trained, the mask pixels are highlighted in red.

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.
- **Point List Datum** — When the “Create Point List Result” option is enabled, this datum will contain a list of all pixels that were counted by the Flaw tool.
- **Output Value** — Contains the sum or count that was computed based on the selected **Tool Method**.

I/O Summary

The Flaw Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Method: xxx, **Mask:** yyy

Where: xxx =

- Count Edge Pixels when selected
- Counting Pixels Within Range of Intensity when selected
- Sum All Intensity Values when selected
- Sum All Edge Energy Values when selected

Where: yyy =

- Enabled when masking is enabled
- Disabled when masking is disabled

Outputs: Tool aaa, bbb: ccc, Min Allowed: ddd, Max Allowed: eee

Where: aaa = Passed when the tool passed inspection, or
 aaa = Failed when the tool failed inspection

If selected method is Count Edge Pixels and the tool passed:

bbb = Edge Count
ccc = Number of edges found during inspection
ddd = Value of the Min Allowed Edge Count
eee = Value of the Max Allowed Edge Count

If selected method is Count Pixels Within Range of Intensities values:

bbb = Pixel Count
ccc = Number of qualified pixels found during inspection
ddd = Value of the Min Allowed Pixel Count
eee = Value of the Max Allowed Pixel Count

If selected method is Sum All Intensity Values:

bbb = Gray Sum
ccc = Sum of all intensity values within the ROI
ddd = Value of the Min Allowed Intensity Sum
eee = Value of the Max Allowed Intensity Sum

If selected method is Sum All Edge Energy Values:

bbb = Edge Energy Sum
ccc = Sum of all edge energy values within the ROI
ddd = Value of the Min Allowed Edge Energy Sum
eee = Value of the Max Allowed Edge Energy Sum

Histogram Analysis

GrayHistogram Tool

This tool verifies whether an image is in focus. It measures how well an image is in focus by calculating a focus number. It also provides a histogram of pixel intensity values along with basic statistics about the histogram (minimum value, maximum value, average value, and standard deviation). You specify the input image and the number of bins in the histogram.

Other Steps Used

None.

Theory of Operation

The technique that measures focus is based on measuring the contrast in the image. Contrast is the difference in image intensity between nearby pixels. For every pixel in the ROI, the GrayHistogram Tool calculates the difference between the pixel value at that location, $I(x,y)$, and the pixel value at a location nearby, $I(x + x_distance, y + y_distance)$. The $x_distance$ and $y_distance$ parameters determine how close together the pixels are. These are the parameters called horizontal sparseness and vertical sparseness. A distance of 1 would be used for images that have great detail and contrast. A distance of 10 might be used for an image with less detail and less contrast. The $x_distance$ and $y_distance$ are usually set to be the same, but if an image has more contrast horizontally than vertically (e.g., the vertical lines of a UPC barcode), the $x_distance$ should be smaller than the $y_distance$ in order to achieve maximum sensitivity.

The sum of all the pixel differences is divided by the total number of pixels in the ROI to calculate the focus number. For a specific image and specific ROI at a specific illumination intensity, the focus number will be greatest when the image is in proper focus.

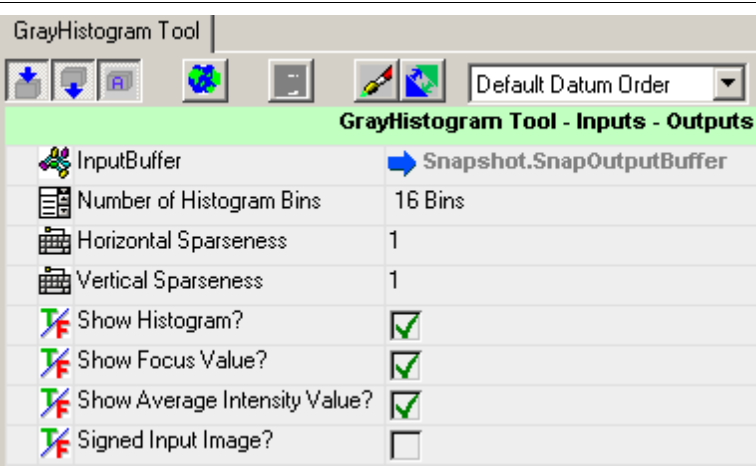
Note: Changes in illumination, image content, and location of the ROI will result in a change in focus number. Changes in intensity of the illumination can be monitored by looking at the average pixel intensity of the ROI and the sum of pixel values.

The GrayHistogram Tool adjusts the camera lens until the focus number is greatest. It can also be used by an operator to verify that the focus set up by a supervisor is still valid at the time of inspection, or to verify that the camera’s focus has not changed over time or been affected by foreign matter on the lens.

Description

The GrayHistogram Tool allows editing through the GrayHistogram Tool properties page, as shown in Figure 7–15.

FIGURE 7–15. GrayHistogram Tool Properties Page



Settings

- Number of Histogram Bins** — Number of bins to be used by the histogramming feature. It can only be set to 16, 32, 64, 128 or 256. It does not affect the calculated focus number, but it will affect the accuracy of the histogram statistics that are calculated. A histogram with 256 bins takes longer to generate than one with 16 bins.
 Default: 16 bins
- Horizontal Sparseness** — Selects the horizontal distance between pixels that computes the image contrast measurement as described in “Theory of Operation” on page 7-54. Use a low value, such as 1, for images with great detail and contrast in the horizontal direction and a larger number, such as 10 or 20, for images with less detail and less contrast in the horizontal direction.
 Default: 1
 Range: 1 to 100

- **Vertical Sparseness** — Selects the vertical distance between pixels that computes the image contrast measurement as described in “Theory of Operation” on page 7-54. Use a low value, such as 1, for images with a lot of detail and contrast in the vertical direction and a larger number, such as 10 or 20, for images with less detail and less contrast in the vertical direction.

Default: 1

Range: 1 to 100

- **Show Histogram?** — The display of the histogram in the upper left-hand corner of the ROI on the input image can be enabled (checked) or disabled. Disabling the display will yield an increase in speed.

Default: Enabled

- **Show Focus Value?** — The display of the focus value in the upper left-hand corner of the ROI on the input image can be enabled (checked) or disabled. Disabling the display will yield an increase in speed.

Default: Enabled

- **Show Average Intensity Value?** — The display of the average intensity value in the upper left-hand corner of the ROI on the input image can be enabled (checked) or disabled. Disabling the display will yield an increase in speed.

Default: Enabled

- **Signed Input Image?** — If checked, then it is assumed that the gray values of the input image are signed values.
- **Discard Black Pixels in Average?** — If checked, pixels with a gray value of 0 will not be included when computing the average gray value of the ROI. When disabled, the average includes all pixels in the ROI.

Training

None.

Results

- **Status** — Set to true after a successful execution of the step.
- **Focus Value** — The measurement of how well the image is focused.

- Sum Pixel Diffs — The sum of pixel differences.
- Histogram Values — A histogram of the pixels in the ROI.
- Pixel Count — The number of pixels.
- Pixel Sum — The sum of pixel intensity.
- Avg Histogram Value — The average histogram value.
- Std Dev of Histogram Values — The standard deviation of the histogram values.
- Min Histogram Value — The minimum histogram value.
- Max Histogram Value — The maximum histogram value.
- Median Histogram Value — The median histogram value.

I/O Summary

The GrayHistogram Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Nbins: aaa Hsparse: bbb Vsparse: ccc

Where: aaa = Number of bins in the histogram
 bbb = Horizontal sparseness parameter
 ccc = Vertical sparseness parameter

Output: Focus: ddd Hmin: eee Hmed: fff Hmax: ggg Havg: hhh
 HStdDev: kkk PixCnt: mmm SumPix: nnn SumPixDif: ppp

Where: ddd = Focus value
 eee = Minimum of the histogram
 fff = Median of the histogram
 ggg = Maximum of the histogram
 hhh = Average of the histogram
 kkk = Standard deviation of the histogram
 mmm = Number of pixels in the source area
 nnn = Sum of the gray values of all the pixels in the source area
 ppp = Sum of the differences between the pixels in the source and their neighbor pixel located at (x + hsparse, y + vsparse).

IntelliFind Tool

The IntelliFind[®] tool finds and locates objects based on the geometry of their contour edges, or contours, for short. Scale factor, orientation, and position are provided for each located match or instance. Models are built from synthetic images or directly from the image using the integrated model editor interface.

IntelliFind[®] finds objects in the image with full 360° rotation and scale invariance with a position accuracy up to 1/30th of a pixel and an angle accuracy of 1/50th of a degree.

The tool is robust to occlusion, image clutter and reverse contrast. IntelliFind[®] can be trained by showing synthetic images. Then, it can find multiple instances of the object in the same image.

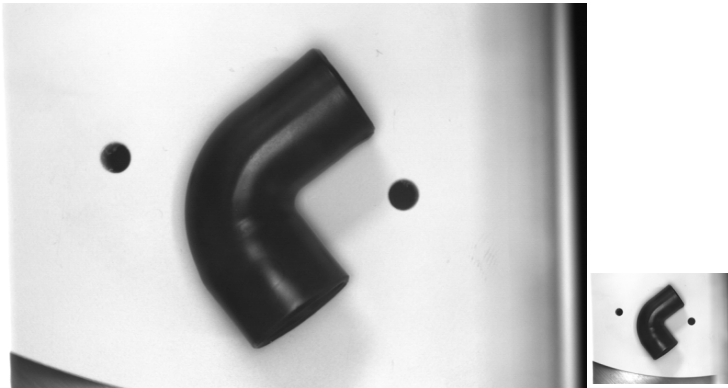
Theory of Operation

IntelliFind's search process operates on two levels of coarseness or image resolution:

- Outline Level — Object features are identified at the coarsest level.
- Detail Level — At this level, focusing the search and refining the position of the object occur.

Note: IntelliFind[®] generates hypotheses of potential instances, based on the Outline level edge contours only. IntelliFind[®] then confirms the hypotheses and refines the location of the instances based on the Detail level edge contours. Therefore, there is no searching for matches performed ever in the detail image.

Figure 7–16 contains an example of Outline and Detail level images, where the Outline level is 4 and the Detail level is 1.

FIGURE 7–16. Detail Level 1 (Original Image) and Outline Level 4 Images

An Outline level image at a value of **n** is a reduced resolution image of the original image where the width and height of the image is **n** times smaller than the original (i.e., divided by **n**).

Typically, Detail level images have a resolution of 1, the same as the original image, unless the original image is somewhat blurry, which IntelliFind® detects and compensates at teach time by reducing the resolution of the image where the fine positioning is performed.

Outline level and Detail level edge contours do not need to be the same contours, and are typically different for most Models. Outline level contours are selected such that they are few (4 or less) and discriminating with respect to the background clutter for reliable pose estimation (i.e., a unique x,y, Theta, Scale transform from the Model to the image object instance). Detail level edge contours are selected such that they are relatively many (up to 10 or so) and are rigid with respect to the Model for good positional accuracy.

Search constraints and Recognition parameters further restrict the search, for example, to find objects within a specific range of scale or orientation.

IntelliFind® will then output the position, orientation and scale of each instance found as standard Datum results that can be used by other tools in the Job. The position returned is the position of the model reference point or hotspot which, by default, is placed in the center of the IntelliFind® Setup ROI at 0°.

IntelliFind® uses Model(s) to locate objects in the image. A Model is a representation of an object, based on the geometry of its contour edges. Models, which are object references for the search, are created and edited using the IntelliFind®'s Model Editor. These can be automatically or manually taught from a given “golden” image or from a synthetic binary image of the object. A synthetic model can be manually drawn or mathematically generated into a binary image.

Model Edge Contours Selection

For IntelliFind[®] to reliably locate instances of objects at runtime, a Model as “perfect” as possible must be taught beforehand.

The set of guidelines for model edge contours selection follows:

- The Model should define the average shape of the part.
- The image used to build the Model should be a clean image where the geometry of the object is clearly visible. This is especially important when the runtime images are very cluttered and difficult. If such an image cannot be produced, then a synthetic image of the geometry of the object should be used.
- The geometry of the background should be removed, if possible. In other words, the background should be uniform.
- Features should be selected such that they are stable in shape and relative positioning with respect to each other, i.e., no shadows or specular reflections.
- Use good “discriminating” features in the model:
 - Relatively long features (edge contours) in the outline image. This is not mandatory but will improve robustness.
 - The edge contours features selected should be rich in curvature variations and distributed along its length for robustness to occlusions.
 - The edge contours selected should contain geometry not present in other objects in the image.
 - Use features that define individually or as a group a complete Translation/Rotation/Scale transform between the model feature(s) and the image contours.
 - A good model should not be symmetrical in rotation unless the rotation is not searched for as is the case for a circle.

Note: A model is taught automatically by simply dragging the IntelliFind® Setup ROI to the region of interest in the image. The IntelliFind® then builds the Model, based on Contour Detection and Feature Selection parameters. This is appropriate for most applications. It is possible, though, to edit these “auto-taught” models in the IntelliFind® Model Editor Interface dialog, which is fully described in the Training section.

Training

The object contours must be trained before IntelliFind® can locate it. A Setup Step, the IntelliFind® Setup Step, provides an ROI that can define the Model boundaries. A new or existing model is trained whenever **Train** is clicked in FrontRunner, or whenever the IntelliFind® set-up ROI is moved or resized around an object in the input image when AutoTeach is enabled in FrontRunner. It is also possible to finetune the contours selected by the AutoTeach capability of IntelliFind® by using the integrated model editor. This editor is invoked by pressing “**Edit Model...**” in the IntelliFind® Setup property page.

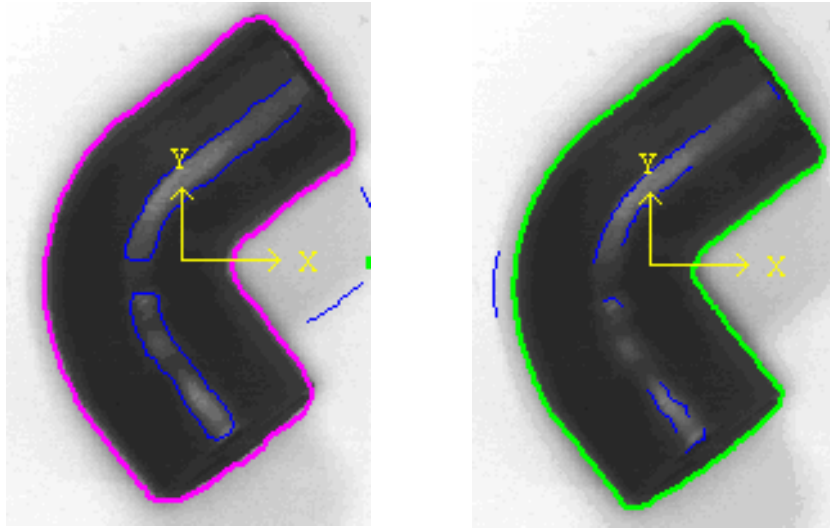
Model building is a three-step process:

1. Edge contours are detected in the image.
2. Then, from the detected contours, appropriate features are selected for both the Outline image and the Detail image.
3. Finally, the Model can be optionally edited manually to add, remove and edit the features that IntelliFind® selected automatically.

In Figure 7–17, magenta contours were the one selected at the outline level, while green contours were the one selected at the Detail level.

Note: Shadows and specular reflections were automatically discarded by IntelliFind®.

In this case, the model is fine and does not require additional editing.

FIGURE 7-17. Outline Level and Detail Level

All contours detected are shown in blue. Note that the contours selected are the same; this does not need to be this way. Also, the outline contours will always seem smoother because they are detected usually on a reduced resolution image but are shown always on the original image resolution (i.e., expanded back up for display).

Model parameters can be changed or defined in the integrated model editor. For the example part above, the editor is shown in Figure 7-16. Notice the following:

- IntelliFind® selected a level of 8 for the outline image and 2 for the detail image (because the image shadows causes these edges to be somewhat blurry and soft).
- By default, IntelliFind® uses the Contour Detection parameters **Contrast Threshold** and **Contrast Mode** set in the property page, and AutoTeach is turned ON “Automatic Levels”.
- Also, IntelliFind® is programmed to use no tracking inertia and has feature selection set to Normal, which indicates that only the best contours as defined in the guidelines section above and according to IntelliFind® internal AutoTeach capability should be selected.

FIGURE 7–18. IntelliFind® Model Editor Dialog

The rest of the UI available in the editor allows manual editing of the contours selected by default by AutoTeach with the extreme ability of manually specifying everything about the model.

If every edge contour has to be manually edited in the Training UI in order for IntelliFind® to work, it is very likely that the image used for training the model is not suitable, either because it has too much clutter or is too noisy. In that case, we recommend, instead, that an ideal image of the model be constructed mathematically and stored in a binary image used to autoteach the model. For example, this can be done by simply painting a binary model using a third party Draw or Paint application or using code to draw model boundaries at precise location in the image and saving the result as a tif image.

Controls in Model Editor UI

The interface in Figure 7–16 modifies existing models and/or creates new models. These models are saved in a model file that has a name as specified by the Model Name Datum in the IntelliFind® Setup property pages and a .hdb extension. By default, this model (Untitled Model1.hdb) is stored in the \Vscape\Jobs\Contours directory on your disk.

Note: Whenever an Job that has IntelliFind® tools in it is copied to a different machine, then the accompanying model files must be copied as well to the destination machine Contours directory.

Typically, the elements that you will create or set for the Model in this interface are:

- Set the bounding area that will restrict the region of the Image used to build the Model. By default, this is set to the position of the IntelliFind® Setup ROI before the Editor is invoked.
- Contour detection parameters for the model edge contours that will be generated from the image or the synthetic image.
- Feature selection parameters for automatically selecting good features from the detected edge contours.
- Edition of selected features, i.e., edge contours.

- Position and orientation of the model reference point or hotspot.

By default, IntelliFind[®] automatically builds a Model from the image when **Train** in FrontRunner is clicked, or when the Setup ROI is moved or resized in the Image. From within the Editor UI, pressing **Build Model** serves the same purpose. There are two ways that a Model can be edited:

- Set up the Contour Detection parameters and Feature Selection slider and click **Build Model**, which executes the AutoTeach feature of IntelliFind[®].
- You are happy with the AutoTaught settings, but simply want to edit/add/remove edge contours. This is done with the mouse selecting edge contours and/or manipulating the “Model Contents” section of the Editor.

Note: Method 1 can be followed by method 2 to come up with the final Model.

Teaching and Building a Model

Click **Build Model** to construct a model:

- Use the Feature Selection slider to increase or decrease the amount of features that are selected from the edge contours found within the Setup ROI. If you want to manually select all the features for a Model afterwards, then set the Feature Selection slider to none. For example, if you want to select all edge contours for a synthetic image of the model, then set it to All.
- If you want to let IntelliFind[®] decide on the best level of coarseness for Outline and Detail level images, then leave the Automatic Levels checked. Otherwise, uncheck it and set these manually:
 - **Outline Level** — The higher the setting (max 32), the lower the contour resolution. This value can only be higher than, or equal to, the Detail level.
 - **Detail Level** — The lower the setting (min 1), the finer the contour resolution. This is always lower than, or equal to, the Outline level.
- **Contrast Threshold** and **Contrast Mode** control the amount of edge contours detected in the model and the sensitivity of the Edge detection at teach time. These parameters are fully described in the Contour parameters.
- Tracking inertia allows edge contours with a gap up to 1 pixel to be connected together into longer edge contours.

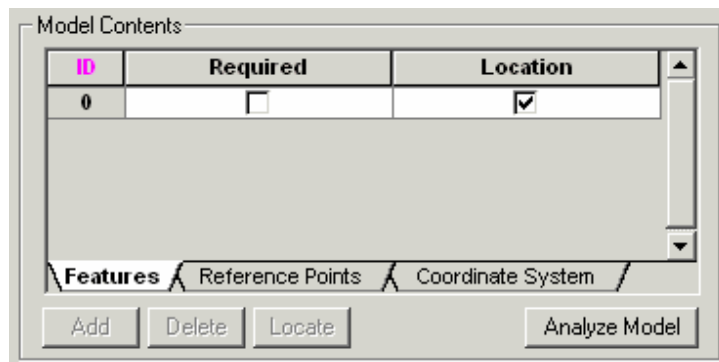
Build Model initiates the Model creation process, which clears the current Model, re-detects the edge contours using the new settings, and selects features from these edge contours according to the Feature Selection slider. The Message frame in the Editor UI displays messages pertaining to the model creation process, including problems that IntelliFind® encountered in building the Model or potential problems that may occur at runtime when searching for the Model.

Editing/Adding or Removing Features from a Model

You can manually select and add features to a Model once it has been built, i.e., once the edge contours have been detected in the image. A feature can be created at either the Detail level or the Outline level. They need not be the same edge contours, since outline contours find candidate matches of the Model and Detail edge contours accurately subpixel fit each candidate match.

Features appear in the Model Contents sub-section of the Editor UI (see Figure 7–19), and also in the image as magenta lines in Outline mode and green lines in Detail mode. If a feature (edge contour) must be broken up into two pieces, first delete the feature from the Feature list and then select from the image again.

FIGURE 7–19. Model Contents Sub-section



Use these steps to add features to the model:

1. In the display, select the level in which you will create the feature:
 - Outline
 - Detail

2. Within the image display, click on the edge contour (blue lines) you want to select as a feature. The entire unbroken contour line turns light blue, indicating that it is ready for selection.
3. To select the entire contour, double-click on the contour.
4. To select only a portion 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. The selected portion appears in red. Modify the selected contour section by holding Ctrl and clicking elsewhere on the line segment. To invert the selected/non-selected portion of a contour, hold the Shift key while clicking on the contour.
5. Click **Add**. The added feature appears in the Model Contents sub-panel. Repeat for additional features.

Follow these steps to remove a feature from the Model:

1. 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 and is highlighted in the grid.
2. Click **Delete** to remove it.

Feature Sub-panel

The grid in the Features sub-panel lists the features (edge contours) that make up the Model. The Required and Location properties can be enabled/disabled independently for each feature in the grid for the outline or the detail level contours.

Location Features

When location is enabled for a feature, this feature is used by IntelliFind® to calculate the location of object matches. If it is unchecked, the feature is only used for object recognition.

For example, a tag or label that is glued to an object often varies in position with respect to the object. Such a label is good for object recognition but should not be used for precise object location.

Required Features

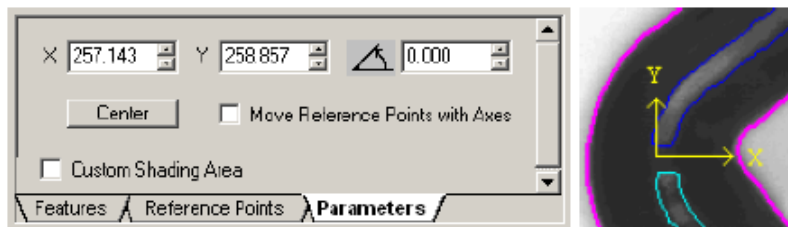
When the required checkbox is enabled, IntelliFind® will not identify an object in which the Required feature has not been found. A feature can be enabled as required even if it is not used for Location.

Setting the Model Reference Frame (HotSpot)

The model reference frame or hotspot is visible in the display area as a small left-handed yellow triad. Use the mouse to move and position this anywhere within the image. The orientation can be set by selecting either the x or the y axis and rotating the mini frame. Sub-pixel positioning can be done by first zooming the display.

Alternatively, a position and orientation can be typed in the Coordinate System sub-panel, as shown in Figure 7–20.

FIGURE 7–20. Coordinate System Sub-panel



Note: The hotspot can also be placed programmatically by using the Model Datum custom COM Interface.

Checking Model Contours

Analyze Model analyzes a Model that you have manually created or edited. 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 box (Setup ROI). The rest of the image is lost.

- Once you have cropped the Model, you cannot return to the original image you used to build the Model.

- Cropping reduces the size of Models in memory, especially for complex Jobs with many IntelliFind[®] tools.

Description

The IntelliFind[®] Setup properties page allows the modification of properties pertinent to teaching IntelliFind[®] Models, as shown in Figure 7–19.

FIGURE 7–21. IntelliFind[®] Setup Properties Page

Settings

- **Model Name** — This is the name that stores the Model to disk. Models are stored in the Vscape\Jobs\Contours directory and must be unique between IntelliFind[®] tools.

Default: Set to Untitled ModelN, where N is the nth IntelliFind[®] tool in the Inspection.

When using the IntelliFind[®] tool in multiple inspections, each instance of the tool must have a unique Model name. If the tool's model names do not have unique names, the training information will be lost when the job is reloaded.

- **Automatic Levels** — Check to let the Tool decide the best decimation levels for the outline and detail level images built from the Model image when the Model is trained.
- **Outline Level** — Resolution of the image used to build coarse contours at Train time. The higher the setting (max 32), the lower the contour resolution. This value can only be higher than, or equal to, the Detail level. Note that Automatic Levels must be unchecked for this setting to be used.
- **Detail Level** — Resolution of the image used to build fine contours at Train time. The lower the setting (min 1), the finer the contour resolution. This is always lower than, or equal to, the Outline level. Note that Automatic Levels must be unchecked for this setting to be used.

- **Feature Selection** — Use this enumeration to increase or decrease the amount of features (contours) that are selected from the edge contours found within the Setup ROI at Train time. If you want to manually select all the features for a Model afterwards, then set the Feature Selection to none. For example, if you want to select all edge contours for a synthetic image of the model, then set it to All. Individual feature selection is done by launching the Model editor using the **Edit Model** button.
- **Contrast Mode / Contrast Threshold** — These control the amount of edge contours detected in the model and the sensitivity of the Edge detection at train time. These parameters are fully described in the Contour parameters at runtime. Note that Train and Runtime Contrast Threshold and Mode need not be the same, especially if the contours are trained on a synthetic image.

Contrast Threshold Range: 1 to 255

- **Lock Model Pixels** — Whenever the IntelliFind® is trained, the current Image is updated into the Model and this option automatically checked. This guarantees that no other image will be used until the option is unchecked again, at which point new pixels will be stored into the Model image and the Model taught again. While **Lock Model Pixels** is checked, the Model can be re-taught and/or contours edited from the Model Editor Dialog Interface.
- **Crop Model Pixels to ROI** — Crop Model reduces the Model to the area contained within the bounding box (Setup ROI). The rest of the image is lost when saving the model to disk. Once you have cropped the Model, you cannot return to the original image you used to build the Model. Cropping reduces the size of Models in memory, and should be used especially for complex Jobs with many IntelliFind® tools.
- **Edit Model** — Launches the model editor, which allows manual editing of individual contours in the Model.

The IntelliFind® Tool properties page is shown in Figure 7–22.

FIGURE 7-22. IntelliFind® Tool Properties Page

IntelliFind Tool | IntelliFind Setup

Default Datum Order ▼

IntelliFind Tool - Inputs

InputBuffer	→ Snapshot.SnapOutputBuffer
Model Contour	→ IntelliFind Setup.IntelliFind Model
Graphics Level	Show Basic Graphics
Extended Graphics Level	<none>
Pass On No Data	<input type="checkbox"/>
Contrast Mode	Adaptive NormalSensitivity
Contrast Threshold	255
Accept Threshold	0.800
Instance Ordering	Evidence
Effort Level	Normal
Positioning Accuracy	Normal
Enable Scale Search	<input type="checkbox"/>
Nominal Scale	1.000
Minimum Scale	0.850
Maximum Scale	1.150
Enable Rotation Search	<input checked="" type="checkbox"/>
Nominal Rotation (degree)	0.000
Minimum Rotation (degree)	-180.000
Maximum Rotation (degree)	180.000
Respect Contrast Polarity	<input checked="" type="checkbox"/>
Locate All Symmetric Instances	<input type="checkbox"/>
Based on Outline Only	<input type="checkbox"/>
Find Every Instance	<input type="checkbox"/>
Instances to Find:	1
Search Timeout (msec)	5000.000
Use Default Conformity Tol	<input checked="" type="checkbox"/>
Conformity Tolerance	7.126
Min Model Percentage	50

TABLE 7-6. Links to Property Descriptions

For Information About...	Go To...
Accept Threshold	page 7-73
Based On Outline Only	page 7-77
Conformity Tolerance	page 7-74
Contrast Mode	page 7-74
Contrast Threshold	page 7-75
Effort Level	page 7-75
Enable Rotation Search	page 7-78
Enable Scale Search	page 7-78
Extended Graphics Level	page 7-73
Find Every Instance	page 7-78
Graphics Level	page 7-72
Instance Ordering	page 7-75
Instances to Find	page 7-79
Locate All Symmetric Instances	page 7-79
Maximum Rotation (degree)	page 7-79
Maximum Scale	page 7-79
Min Model Percentage	page 7-76
Minimum Rotation (degree)	page 7-80
Minimum Scale	page 7-80
Model Contour	page 7-72
Nominal Rotation (degree)	page 7-80
Nominal Scale	page 7-80
Pass On No Data	page 7-76
Positioning Accuracy	page 7-76
Respect Contrast Polarity	page 7-80
Search Timeout (msec)	page 7-76
Use Default Conformity Tol	page 7-77

- **InputBuffer** — Displays the name of the image buffer that the tool will process. The default buffer is the output buffer of its parent. This property is read-only.

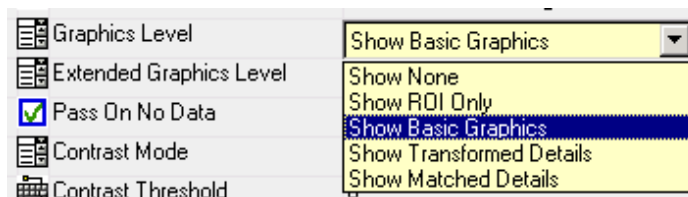
- **Model Contour** — Specifies which Model is connected to this IntelliFind® tool. The default model is the model output of its setup tool. This property is read-only.

Graphic Feedback Properties

These properties control the amount of visual feedback provided by the Tool.

- **Graphics Level** — Five graphics levels are provided for drawing the result of this tool.

FIGURE 7-23. Graphics Level



- **Show None** — No graphics are displayed.
- **Show ROI Only** — Draws the search ROI only. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red.
- **Show Basic Graphics** — Draws the ROI. When the tool passes, the ROI is drawn in green. When the tool fails, the ROI is drawn in red. The instance(s) matched by the tool are drawn as a little triad that show the position and orientation of the match. When more than one instance is programmed to be found, the best match is drawn in yellow and subsequent matches are drawn in blue, as shown below: The triad is positioned at the hotspot as defined by the Model.
- **Show Transformed Details** — Draws graphics per Show Basic Graphics. In addition, the boundaries of the Model transformed to that instance position, orientation and scale are drawn as well. The best match is in yellow and subsequent matches are drawn in blue.
- **Show Matched Details** — Draws graphics per Show Basic Graphics. In addition, the boundaries of the Model matched to that instance position, orientation and scale are drawn as well. The best match is in yellow and subsequent matches are drawn in blue. However, only the pieces of contour that were actually found in the instances in the image that match the model contours are shown.

Note: Whenever the Tool fails and has located matches that have a FitQuality lower than the AcceptThr, then the graphics will show position and boundaries in red of all these instances. If no instances are found, then only the ROI will be drawn in red.

- **Extended Graphics Level** — Two graphics levels are provided for drawing the edges found by the tool within the search area during the search process and/or the locate process. These graphics are displayed regardless of whether the Tool finds a match or not.

FIGURE 7-24. Extended Graphics Level



- **Show Contour Scene Outline** — Draws the edge contours detected in the reduced-resolution image or outline image in magenta according to the gradient detection parameters set at teach time or programmed at runtime into the Tool (see “Runtime Properties” on page 7-73).
- **Show Contour Scene Detail** — Draws the edge contours detected in the normal-resolution image or detail image in green according to the gradient detection parameters set at teach time or programmed at runtime into the Tool (see “Runtime Properties” on page 7-73).

Runtime Properties

A number of properties control how the edge contours are detected and how much work is done by the tool to find and accurately position each match.

- **Accept Threshold** — This is the minimum MatchQuality that each instance must have (i.e., \geq) in order for the IntelliFind® tool to Pass. The match quality is a number between 0.1 and 1.0, where 1.0 means that 100% of the model edge contours were successfully matched to the actual edge contours detected in the image for each instance.

Note: This setting works in concert with the Minimum Model Percentage parameter, on page 7-76.

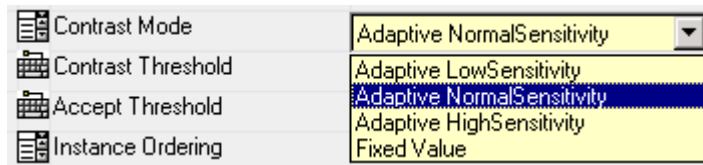
Default: 0.8 (80%)
Range: 0.1 to 1.0

- **Conformity Tolerance** — When the **Use Default Conformity Tolerance** parameter is not set, then a fixed value can be entered in this field.

Range: 0.667 to 40

- **Contrast Mode** — Specifies which method thresholds the edge contour information at runtime. Four methods are available:

FIGURE 7–25. Contrast Mode



The Adaptive options calculate a gradient threshold at runtime for detection of the edge contours based on image content:

- **Adaptive LowSensitivity** — Sets the contrast threshold such that only strong edges (with gradients > 25 or so) are retained, and noise is eliminated at the risk of losing significant edge contour segments.
- **Adaptive NormalSensitivity** — This is the default; it sets the contrast threshold such that mostly good edges (with gradients > 15 or so) are retained, which is appropriate for reasonably well contrasted images.
- **Adaptive HighSensitivity** — Sets the contrast threshold such that low contrast edges and sometimes noisy edge segments (with gradients > 2 or so) are retained.

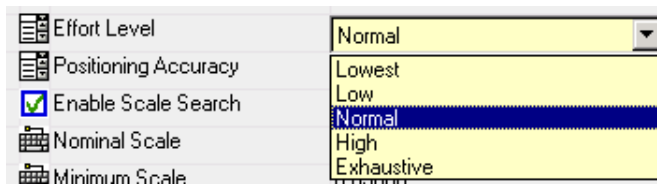
Note: For all the adaptive sensitivity settings, the contrast threshold selected by the Tool will be updated into the **Contrast Threshold** parameter each time the tool executes as if it were an output result.

- **Fixed Value** — Sets the contrast threshold to a fixed absolute value as defined in the **Contrast Threshold**. A typical situation for the use of a fixed value is when there is little variation in the image contrast or when specular reflections and shadows produce edge contours that have high-contrast but are noisy with respect to the model contours.

Note: It is also possible to dynamically change the contrast value when the mode is set to “Fixed Value” by connecting the value directly to another tool result output that calculates the contrast threshold that should be used for each image.

- **Contrast Threshold** — This numerical fixed value corresponds to the minimum step in greylevel values required to detect and record edge contours in the image.
- **Effort Level** — This parameter provides a mechanism for adjusting the recognition effort for the tool. There are five options available. This setting has a large impact on the robustness of the tool at the cost of performance.

FIGURE 7–26. Effort Level



- **Lowest** — This setting improves speed for easily recognizable objects on clean backgrounds with mostly closed contours.
- **Exhaustive** — This setting is appropriate for hard to recognize objects, consisting of many small features on a cluttered background with mostly discontinuous contours.

Note: Normal setting is appropriate for most models and provides the best performance/robustness tradeoff. We recommend that this parameter be set to a level that reliably locates object instances and then be lowered until the tool fails to detect objects in normal variation of lighting, occlusion and/or background clutter.

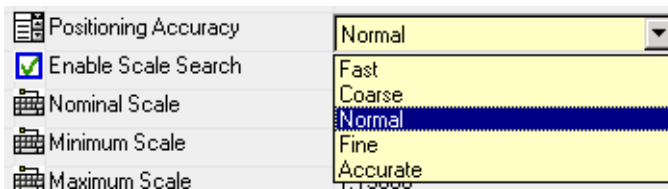
- **Instance Ordering** — This parameter sets the order in which match instances are organized in the results array. At the default Evidence setting, the instances are ordered according to their match quality strength from highest to lowest. The instances can also be ordered in the order they appear in the image, i.e., leftmost, rightmost, topmost or bottommost.

- **Min Model Percentage** — This parameter allows the tool to be configured for occlusion. It represents the minimum % of model edge contours (as length) that must match before the instance is considered a possible match. Increasing this setting can accelerate the recognition but, of course, degrades the robustness to occlusion of the tool. If the **Effort Level** is set to exhaustive and this parameter has low values (< 40%), the execution time significantly increases.

Note: If portions of the object edge contours are outside the search region or outside the image buffer, reducing this parameter will cause the tool to find these instances as well, as long as at least “Min Model Percentage” of the edges are matched.

- **Pass On No Data** — In some situations, the “absence” of an object instance in the image is a valid inspection performed by the tool. Setting this parameter allows the Tool to pass when no instances are found in the image.
- **Positioning Accuracy** — This parameter provides a mechanism for adjusting the speed/accuracy tradeoff of the positioning process through a scale that ranges from Fast to Accurate.

FIGURE 7-27. Positioning Accuracy



- Fast — Adequate when accuracy is not as important as performance. This option can be selected for presence/absence application.

Note: Refer also to the **Based On Outline Only** parameter in “Search Constraints” on page 7-77

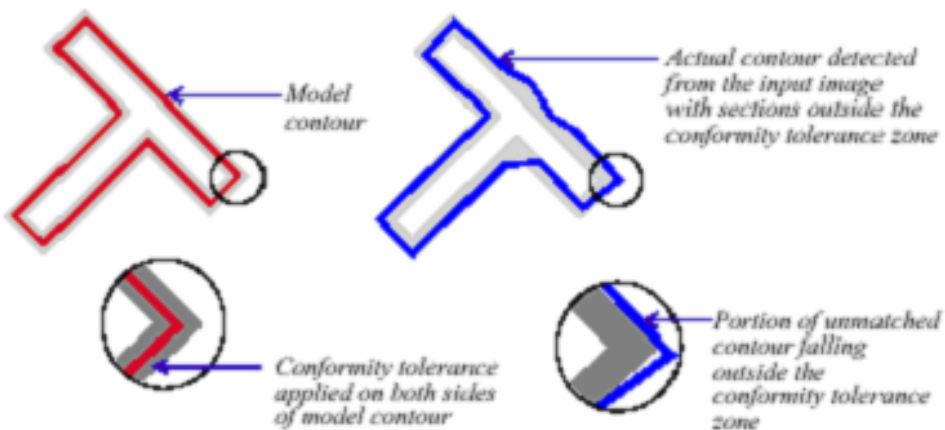
- Accurate — Provides a slight gain in sub-pixel accuracy at the cost of some performance.
- **Search Timeout(msec)** — For time critical applications or applications that must run within an allotted amount of time, a timeout in msec can be set.

When the timeout condition is reached, only the instance matches found up to the timeout are available as results.

Range: 2 to 100000

- **Use Default Conformity Tolerance** — If the features of the model can be locally distorted in the image at runtime, increasing the conformity tolerance can improve the recognition process. However, large values can also trigger false recognitions. When this parameter is checked, the tool will come up with a default value based on the model. Typically, it is set to 2 pixels. Its definition is illustrated in Figure 7–28. The value selected by the tool at teach time is reflected in the next parameter as an output.

FIGURE 7–28. Default Conformity Tolerance Illustration



Search Constraints

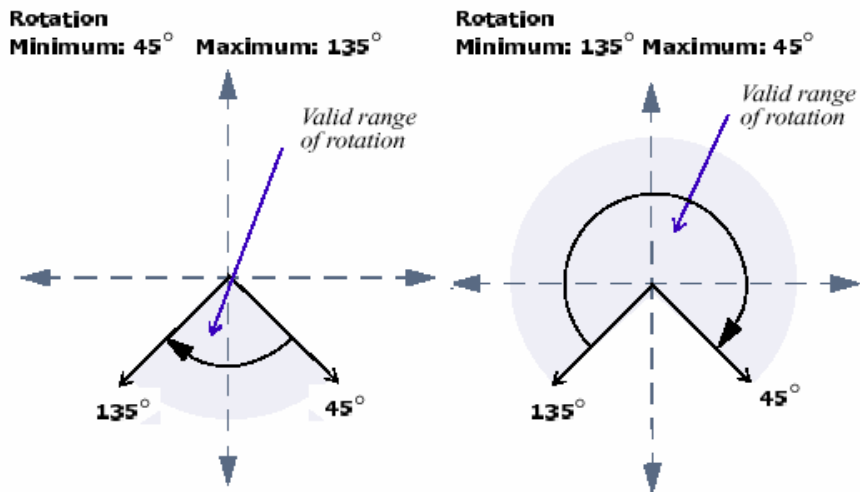
These properties provide restrictions for the IntelliFind's search process and let the tool be configured for the best robustness/performance trade-off.

- **Based On Outline Only** — This mode is best used for well-contrasted images that provide clear contours, with little noise. Because of the lack of positional accuracy, this mode is best suited also for time-critical applications where objects need only be coarsely identified and located, as is the case in a presence/absence inspection. Enabling this parameter can significantly reduce the execution time, at the expense of a loss in positional accuracy.

Note: When there is severe noise and many confusing edge contours at the Detail level in the search ROI, enabling this option can sometimes allow the IntelliFind[®] tool to find the object where it would otherwise fail and report no matches.

- **Enable Rotation Search** — By default, the IntelliFind[®] tool finds objects within a full range of rotation values, i.e., between -180° and $+180^{\circ}$. Allowable rotation can be restricted by either limiting the range of rotation or setting a fixed nominal value (with respect to the Model). The rotation range spans clockwise from the minimum angle to the maximum angle, as shown in Figure 7–29.

FIGURE 7–29. Rotation Range



- **Enable Scale Search** — The scale of objects to be located can be set to a fixed nominal value (i.e., the model scale), or as a range of scale values. This is enabled by this parameter.
- **Find Every Instance** — The number of objects that can be recognized by IntelliFind[®] in an image can be arbitrarily large, and is limited only by physical factors such as the number of objects that can fit within the search area. It is possible for the tool to search for all object instances and outputs, as many as can fit in the results array as set by the **Instances To Find** parameter.

- **Instances To Find** — Limits the total number of objects to be found in the input image within the search area to this value. This is NOT the total number of objects present in the image necessarily. If the number of objects exceeds this constraint, the IntelliFind[®] tool outputs only the best instances found up to that number.

Note: If **Find Every Instance** is NOT enabled but the tool has been set with an instance count constraint by setting the **Instances To Find** parameter, then IntelliFind[®] will search and locate up to the specified count of objects that have a match quality higher than the **Accept Threshold** parameter. This is not necessarily the N best instances, unless the count specified is greater than the number of instances that actually exist in the image.

- **Locate All Symmetric Instances** — When unchecked, the IntelliFind[®] tool only outputs the most plausible position and orientation for an instance. This could cause the tool to fail to recognize an object that is flipped from its model orientation or seen as “View from Behind”. In that case, enabling this constraint for symmetrical objects causes the IntelliFind[®] tool to be able to search for all possible orientations symmetric from one another of a single object. This is equivalent to teaching all symmetrical views of the same object and searching for them.
- **Maximum Rotation** — Upper value of rotation range when the tool is configured to find objects at different orientations.

Note: If a nominal rotation value is used with objects that present a slight variation in orientation from image to image, the objects may be positioned less accurately because their true orientation is not measured. In such cases, it is preferable to configure a narrow rotation range of $\pm 1^\circ$ instead of turning off orientation search all together, to improve sub-pixel position accuracy of the matches.

Range: -180° to 180°

- **Maximum Scale** — Upper value of the scale range when the tool is configured to find instances with variable scale change.

Note: If a nominal scale value is used with objects that present a slight variation in scale from image to image, the objects may be positioned less accurately because their true scale is not measured. In such cases, it is preferable to configure a narrow scale range, such as $\pm 2\%$ (0.02), instead of turning off scale all together, to improve sub-pixel position accuracy of the matches.

Range: 0.1 to 10

- **Minimum Rotation** — Lower value of rotation range when the tool is configured to find objects at different orientations.

Range: -180° to 180°

- **Minimum Scale** — Lower value of the scale range when the tool is configured to find instances with variable scale change.

Range: 0.1 to 10

- **Nominal Rotation (degree)** — If the rotation relationship between the model and instances in the image is known and fixed, then a different value can be set here.

Default: Nominal value of 0

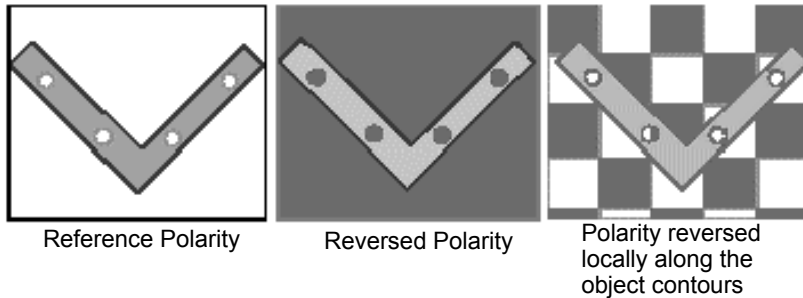
Range: -180° to 180°

- **Nominal Scale** — If the scale relationship between the model and instances in the image is known and fixed, then a different value can be set here.

Default: Nominal value of 1

Range: 0.1 to 10

- **Respect Contrast Polarity** — This parameter defines the polarity of the change in greylevel values between an object and its surrounding background. This polarity can be dark to light or light to dark. The original polarity for a search is defined by its model (see reference polarity in Figure 7–30). By default, this parameter is checked as, in most applications, the polarity of objects is invariable and set by the optics/lighting used. The detection in that case of object instances is usually more robust and resilient to false detections and also faster. In the case where contrast reversal is expected for normally lit conditions, then this parameter should be unchecked, as shown in Figure 7–30.

FIGURE 7–30. Contrast Polarity

Results

The results that are available from the IntelliFind[®] tool can be displayed in the parameter pane of the Job Tree view, as shown in Figure 7–31. They can also be selected as result in the Inspection Step; this is also referred to as “tagging a result for upload”. All of the IntelliFind[®] results can be used as input to other tools. For example, the Instance position can be input directly into a OnePt Locator and Warp Tool to correct for position, orientation and scale of the object at runtime. Results of the IntelliFind[®] tool can also be used as input to a Tolerance Tool to compute statistics about the position and orientation of an object.

FIGURE 7-31. IntelliFind[®] Tool Results

Parameters	ESP Values
[-] Outputs	
[-] Output 1 Parameters	
-- Output On	Mismatch or Noread
-- Polarity	Negative
-- Pulse Width	5
[-] Output 2 Parameters	
-- Output On	Mismatch or Noread
-- Polarity	Negative
-- Pulse Width	5
[-] Beeper	
-- Status	On Good
-- Beeper Speed	Fast
[-] Bar Code Output	
-- Status	Good Read
-- When to Output	As Soon As Possible
-- Laser ON/OFF	Disabled
[-] Noread Message	
-- Status	Enabled
-- Message Output	NOREAD

TABLE 7-7. IntelliFind[®] Tool Results

Name	Symbolic Name	Type	Description
...			Additional Instance up to "Instance Count" found this run.
Fit Scores List	FitScoresLst	Datum.DblList	List of instance scores
Instance Count	InstCount	Datum.Int	Number of instances found this run up to "Instances To Find" programmed into the IntelliFind [®] Tool.
Instance Points	InstPoints	Datum.PtList	List of instance points
Instance1 Fit Quality	Inst1FitQual	Datum.Double	Instance 1 Fit Quality (0.0->1.0)
Instance1 Point	Inst1Pt	Datum.Point	Instance 1 Position, Orientation and Scale encoded as [X, Y, R, S]
Instance1 Scale	Inst1Scale	Datum.Double	Instance 1 Scale
Instance2 Fit Quality	Inst2FitQual	Datum.Double	Instance 2 Fit Quality (0.0->1.0)
Instance2 Point	Inst2Pt	Datum.Point	Instance 2 Position
Instance2 Scale	Inst2Scale	Datum.Double	Instance 2 Scale
Status	Status	Datum.Status	Pass/Fail Boolean value for the Tool. If every instance found has a fit quality > AcceptThr then the Tool passes, otherwise it fails.

I/O Summary

The IntelliFind® Tool provides an I/O summary in the Status Bar located at the bottom of the FrontRunner window, as shown in Figure 7–32.

FIGURE 7–32. IntelliFind® Tool I/O Summary

Inputs: QFitThr: 0.800
Outputs: Process: 42.50 ms 1(1) instances(total) found Ipt: X=299.000 Y=212.000 R=0.000 S=1.000 Qfit=1.000

Inputs: QFitThr: 0.800 [number from 0.0 to 1.0]

Represents the minimal percentage of model contours matched in length against the object instance in the Image in order for IntelliFind® to accept the object instance as a valid match.

Outputs: Process: xxx msec 1(1) Instances(total) found Ipt: X=xxx Y=xxx
R=xxx S=xxx Qfit=xxx

Where: xxx — Actual value for that result.

Process time — Time spent in the IntelliFind® tool only.

1(1) Instances(total) — How many instances were requested and found total by IntelliFind®.

X, Y — Position (% of model contours matched in length).

R — Orientation (% of model contours matched in length).

S — Scale factor (% of model contours matched in length).

Qfit — Fit quality (% of model contours matched in length).

Output messages:

- Process: xxx ms No Match Candidates

Correlation Analysis

Correlation

Correlation locates a particular pattern in an image using a correlation search algorithm. The pattern is called the template. This template is matched with the image in the ROI. The position of the best match is determined, as well as the correlation value at that position. The correlation value indicates how well the selected image match is (1.0 = perfect match, 0.0 = no correlation).

During setup, a section of the image containing the pattern of interest is identified. The template is remembered by the system in full gray level detail at training. This template subsequently locates the pattern in new images by searching for sections that are similar to the template.

Ideally, the template should contain an unchanging part of the pattern and some of its background without containing variable objects in the background.

The correlation technique works for arbitrarily shaped objects. It locates the pattern with sub-pixel accuracy, and can locate the pattern accurately even when the objects have small defects and when the light level changes.

Choosing a Template

The selection of the template has a great impact on the speed, accuracy, and robustness of the correlation. A good template should:

- Have some relatively large, distinctive features that are unique within the search region
- Have both vertical and horizontal edges
- Possess features that are not greatly affected by normal process variations

Figure 7–33 demonstrates good template examples taken from actual images. Each template shown contains chunky features with good contrast against the background, good for fast searching.

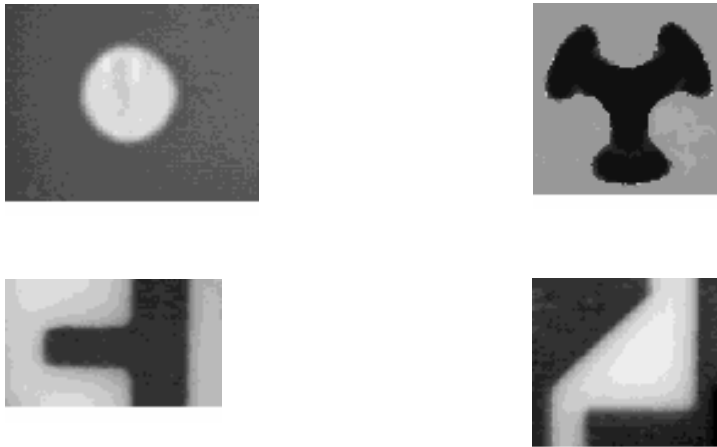
FIGURE 7-33. Good Templates for Fast Searching

Figure 7-34, Figure 7-35, and Figure 7-36 show examples of poor choices.

FIGURE 7-34. Poor Choice #1 — No Horizontal Edges

Possible problems with the template in Figure 7-35 include slow search (no chunky features) and uniqueness, which is a problem when the pattern is repeated beyond the template region.

FIGURE 7-35. Poor Choice #2 — No Chunky Features

The example in Figure 7-36 demonstrates poor contrast, and dominant features that are susceptible to process.

FIGURE 7-36. Poor Choice #3 — Poor Contrast

Template Find

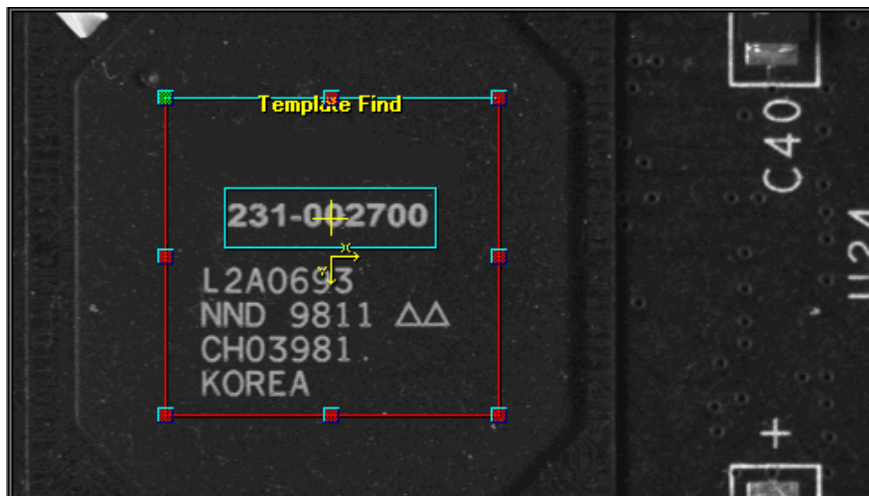
Template Find locates a user-defined pattern in an image. You must train Template Find to specify the pattern to locate. You must also specify the area in which to perform the search. The result of running Template Find is the point on the image where the template pattern is found.

Other Steps Used

TemplateSetup

Theory of Operation

Template Find locates a template image within a given ROI. A template image is first created by training TemplateSetup, as shown in Figure 7-37.

FIGURE 7-37. TemplateSetup — Trained

When Template Find is run, a search is performed to locate matches to the template within an ROI. The result of the Template Find is one or more points displaying the locations of the matches.

Description

A Template Find step requires a template that defines the pattern to be found at runtime. The template can be created either by its set-up step, TemplateSetup, or by using a template output produced by a TemplateSetup step owned by another Template Find step in the Job.

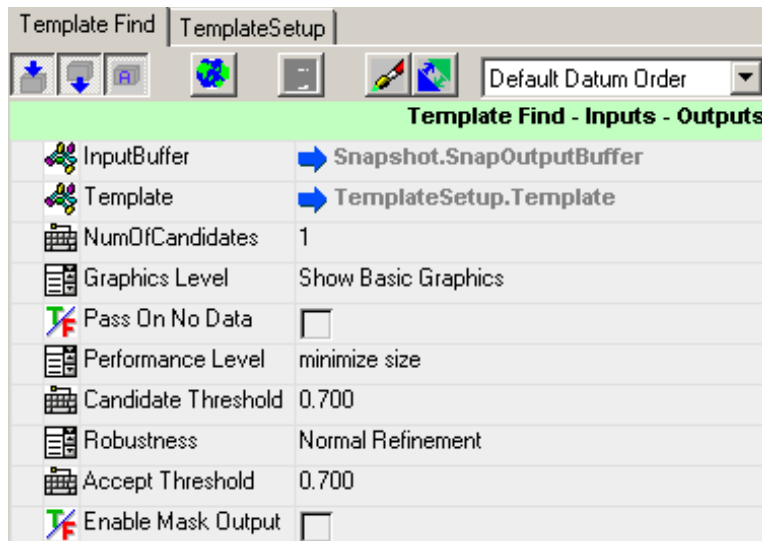
By default, when a Template Find is inserted into a Job, a TemplateSetup step is also created for training the template. This set-up step can be deleted from the Job if a connection has been set up to use another TemplateSetup step's output. This connection can be made through the datum view of the Template Find step.

Both Template Find and TemplateSetup steps have their own ROI labeled "Template Find" and "TemplateSetup," respectively. Both ROIs can be independently adjusted by moving and sizing the shapes. The Template ROI defines the template pattern and the Template Find ROI defines the search area at runtime.

Train the template by training the Template Find step that owns the Template setup. Once a template is trained, all those Template Find steps that use the template are trained.

Template Find allows editing through the Template Find properties page, as shown in Figure 7–38.

FIGURE 7-38. Template Find Properties Page



Settings

- **Template** — Allows connecting an external template (most likely created by a TemplateSetup step in the Job) as input. When the Template Find step is first created, this Template points to the output template of the step's own TemplateSetup. Once this is connected to an external template, the Template Find's own TemplateSetup step can be deleted. The set-up step can also be inserted back to the Template Find step.
- **NumOfCandidates** — Sets the maximum number of candidates to be returned from the step. A candidate is defined as a point at which the Template Find locates a match to the template in the image.

Default: 1

- **Graphics Level** — Sets the amount of output detail shown:
 - Show Basic Graphics (Default) — Draws a yellow mark at the best match location.
 - Show Details — Draws a yellow mark at the best match location and up to the value set at NumOfCandidates of blue marks at the other candidate locations.

- Show Details And Mask — Draws a yellow mark at the best match location and up to the value set at NumOfCandidates of blue marks at the other candidate locations and displays any input mask defined for the step.
 - Show None — Disables the output graphics.
 - Show ROI Only — Draws the outline of the step's ROI.
- **Pass On No Data** - Allows the program execution to continue even if no match is found. This is useful when using the Template Find for alignment purposes. If no match is found, the original trained point can be used for the alignment and proceed with the inspection. By default, this setting is turned off, so the Template Find must come up with a match in order for the step to pass.
- **Performance Level** — The correlation search used by the Template Find can run in a mode in which it minimizes the amount of memory used during execution. This can be set up so that memory use is minimized by choosing the Minimize Size option (default).

The Template Find step supports dynamic location only when this datum entry is set to Minimize Size.

For Jobs that need faster processing, use the Maximize Speed option so that speed performance is optimal.

- **Candidate Threshold** — The minimum correlation score above which to consider a candidate. When the Template Find determines that there is a match of the Template within the ROI, it must ensure that the score generated by the match is above this input threshold.

Default: 0.70

Range: 0.1 to 1.0

- **Robustness** — Sets the robustness level of a search performed by a Template Find. There are five levels of increasing refinement, each of which forces the Find to perform a more exhaustive search. This increases the time it takes to complete the search but reduces the chances of a match being missed:
 - Normal Refinement (default)
 - Added Refinement

- More Refinement
- High Refinement
- Maximum Refinement
- **Accept Threshold** — Sets the minimum correlation score needed for a point to be considered a match.

Default: 0.70

Range: 0.1 to 1.0

Note: A correlation score below 0.5 is considered poor and should be used with caution.

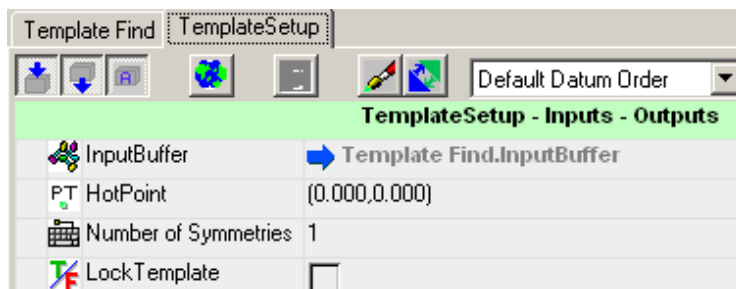
- **Enable Mask Output** — When enabled, an image mask will be created for use by a DynamMask Tool elsewhere in the inspection.

Default: Disabled

Description

Template Setup allows editing through the Template Setup properties page, as shown in Figure 7–39.

FIGURE 7–39. TemplateSetup Properties Page



- **HotPoint** — Specifies a location on the template image relative to the top left corner of the template. When the Template Find step locates a match to this template, the point it returns in image coordinates will correspond to the HotPoint location in the template.

- **Number of Symmetries** — The number of templates (including rotated ones) that will be created in order to take advantage of the symmetry of the template. Symmetry information helps determine the hot point of the template accurately at training time. The templates will be created at an angle spacing that equals 360° divided by the number of symmetries. For example, when this parameter is set to 2, two templates will be created at rotations of 0° and 180° , respectively. When it is set to 3, three templates will be created at 0° , 120° and 240° , respectively.

Note: To train successfully, each rotation of the template image must be found in the search area. The object being trained should have the same number of symmetries as set in the tool. In other words, 2 symmetries for a rectangular feature, 3 for a triangular feature, 4 for a square feature, or any number for circular features.

Default: 1 (Single unrotated template)

- **Lock Template** — When this option is enabled, a previously trained template image will not be overwritten at the next training action. If the Template Find is not trained, the template pattern will be updated only on the first time Template Find is trained and not overwritten on any subsequent training.

Training

Training involves placing the Template ROI around an area of an image from which you want to make a template, and initiating the train. The anchor point will be the center of the ROI. During training, a copy of the selected region of the image is placed into the step program, and statistics about the template are calculated to help speed the find process. When a program is saved, the template image is also saved. Then, when the program is loaded, the original template image is reloaded and Template Find is automatically trained on this image. The Template Find ROI must be adjusted so that the trained template can be found within that region as the part moves normally under the camera.

Once a template is trained, it will remain “Trained” until it is re-trained. This implies that a trained template will stay trained even though the shape of the template has been changed since the template was last trained.

Results

The results of the Template Find will be a point in the Template Find's ROI that best matches the template image according to its correlation score. If no matches are found with a correlation score of the Accept Threshold, then no point will be returned.

- **Status** — Is set to true if a match is found.
- **PointsFound** — A list of all points found that meet the Accept Threshold. The number of points returned in this list is limited by the input parameter NumOfCandidates.
- **ValuesFound** — A list of correlation scores; each entry corresponds to an entry in the PointsFound list.
- **PointofBestMatch** — The single point with the best correlation score.
- **CorrofBestMatch** — The correlation score of the point of best match.
- **NumOfMatches** — The total number of matches found that satisfy the match criteria.

I/O Summary

The Template Find provides the following I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Robustness: aaa AcceptThr: bbb CandThr: ccc NumCandidates: ddd

Where: aaa = Least, Some, More, Better, or Most, depending on the setting for Robustness

bbb = the setting for Accept Threshold

ccc = the setting for Candidate Threshold

ddd = the setting for NumOfCandidates

Outputs: MPt: X=eee Y=fff Score=ggg NumMatches:hhh

Where: eee = the X value of the point of best match found

fff = the Y value of the point of best match found

ggg = the correlation score of the best match found

hhh = the number of matches found

Note: The Output line of the I/O summary may also read “No Match Candidates Found” if the Template Find failed to locate a match to the template.

Vector Tool

This tool extracts edge points from an image along an arbitrary set of vectors. Then, these points can be fit to a line or circle using additional fitting steps. A vector set can consist of any number of vectors, each of which can be freely placed anywhere in the image.

Edge points are pixel locations in the image where there is a large change in grayscale value around its location. The inputs to Vector Tool are the buffer that contains the image and the VectorFamilyShape that describes the set of vectors. The output of the Vector Tool is a set of points, one from each vector.

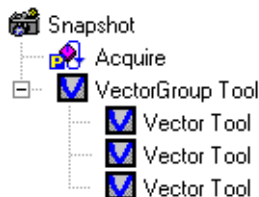
The Vector Tool can be inserted into a Job anywhere there is an image available, such as a snapshot step or morphology, as shown in Figure 7–40. It can also be inserted into a VectorGroup Tool for speed optimization. When not inserted into a VectorGroup Tool, the vectors will be processed normally by the Vector Tool.

FIGURE 7–40. Vector Step Inserted into a Job



When the Vector Tools are inserted into a VectorGroup Tool (see Figure 7–41), the set of Vector Tools that are child steps of the VectorGroup Tool will be processed as a group. This will be faster when there are multiple Vector Tools to be processed on the same image. When there is only a single Vector Tool, there is no speed advantage to using the VectorGroup Tool.

FIGURE 7–41. Vector Tools Inserted into VectorGroup Tool



Description

When a Vector Tool is inserted into a Visionscape Job, the settings that control the behavior are available on the tool's properties pages, as shown in Figure 7–42 and Figure 7–43.

FIGURE 7-42. Vector Tool Properties Pages — Parallel (Default)

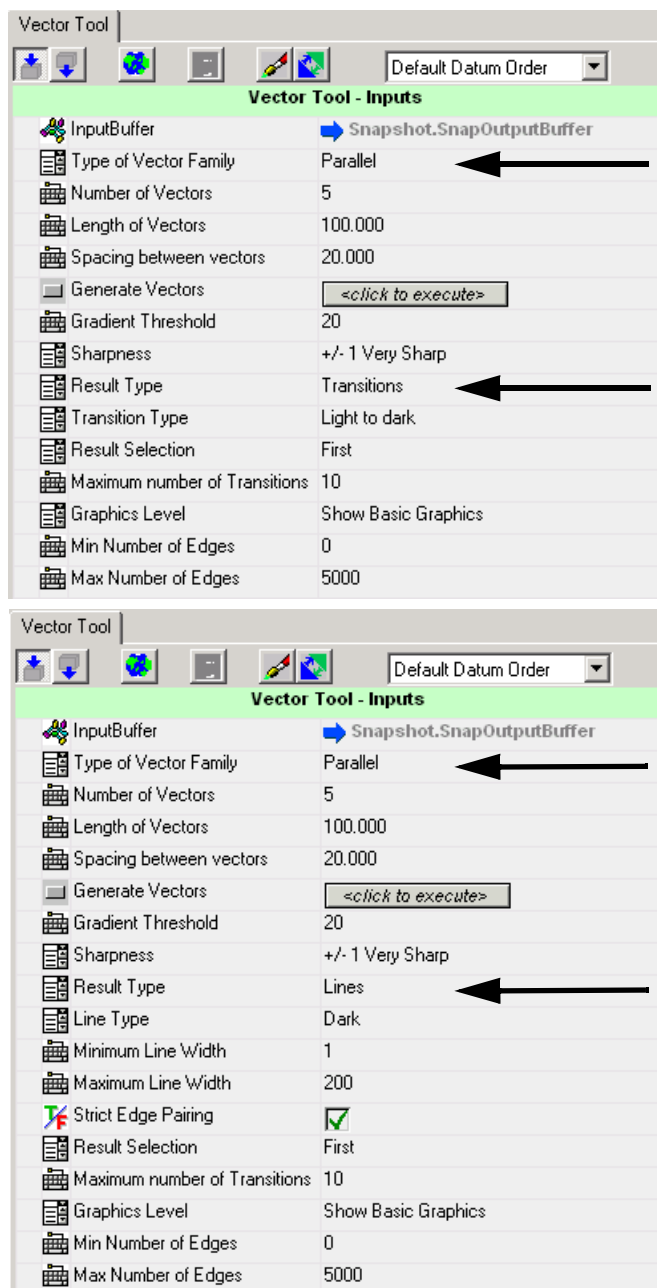


FIGURE 7-43. Vector Tool Properties Pages — Radial

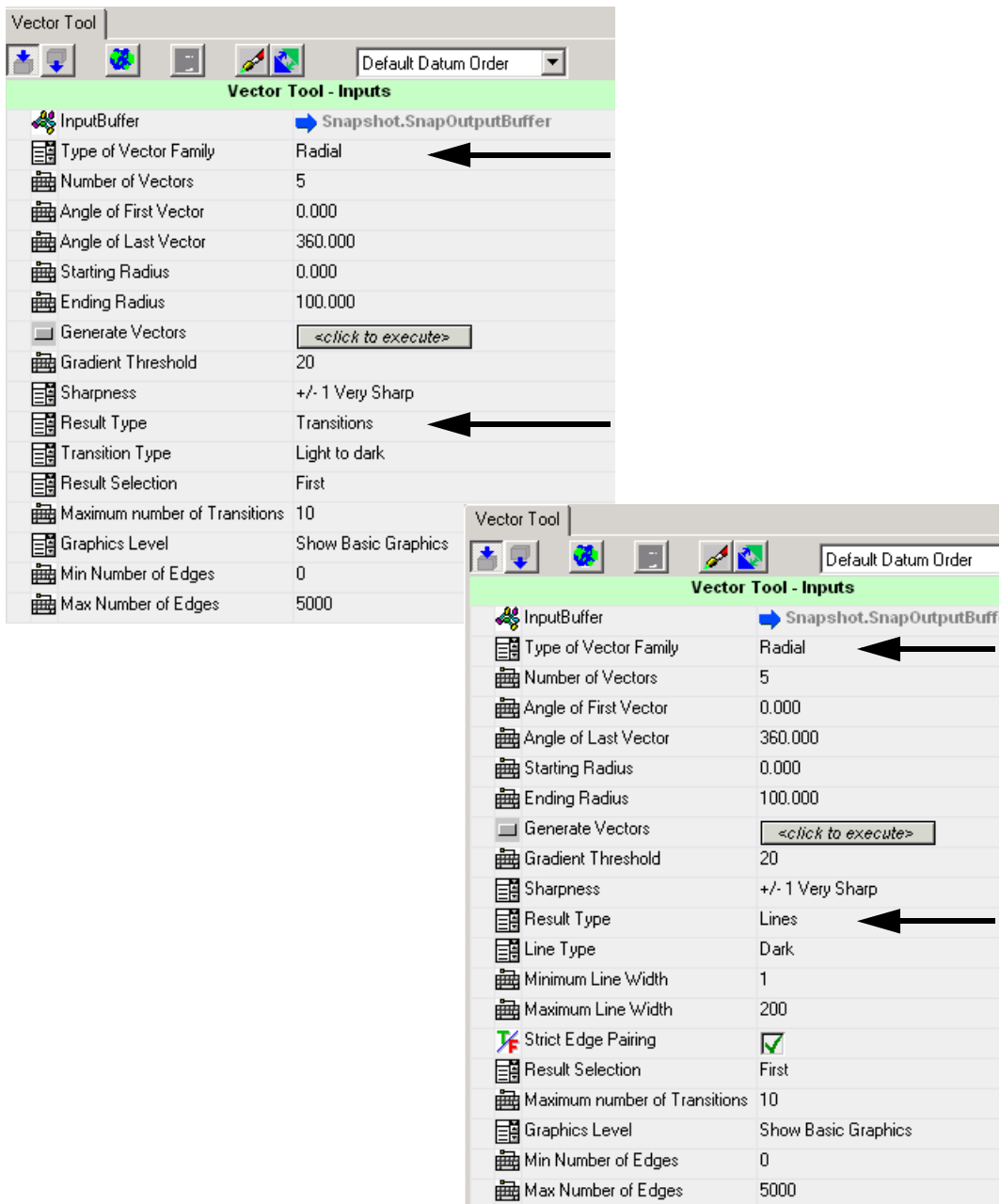


TABLE 7-8. Links to Property Descriptions

For Information About...	Go To...
Angle of First Vector	page 7-97
Angle of Last Vector	page 7-97
Ending Radius	page 7-97
Generate Vectors	page 7-98
Gradient Threshold	page 7-98
Graphics Level	page 7-100
Length of Vectors	page 7-97
Line Type	page 7-99
Max Number of Edges	page 7-100
Maximum Line Width	page 7-99
Maximum number of Transitions	page 7-100
Min Number of Edges	page 7-100
Minimum Line Width	page 7-99
Nth result	page 7-100
Number of Vectors	page 7-97
Result Selection	page 7-99
Result Type	page 7-98
Sharpness	page 7-98
Spacing between vectors	page 7-98
Starting Radius	page 7-97
Strict Edge Pairing	page 7-99
Transition Type	page 7-99
Type of Vector Family	page 7-97

Settings

- **Type of Vector Family** — Provides a choice between Parallel and Radial. This property specifies how the set of vectors will be generated when **Generate Vectors** is clicked. This selection will also affect which of the other vector family specification datums are used.
 - When you select Parallel, a set of horizontal parallel vectors will be generated; the center of the group will coincide with the anchor point of the shape. This is the default.
 - When you select Radial, a set of vectors will be generated that are centered at the anchor point, then extend outward equally spaced.
- **Number of Vectors** — Specifies the number of vectors to create when **Generate Vectors** is clicked. When **Type of Vector Family** is set to Radial, this setting will determine the angular spacing of the vectors.
- **Angle of First Vector/Angle of Last Vector** — These values specify the angular range, in degrees, of the vector set when generating a radial set of vectors. The vectors will be equally spaced within this range. For example, when generating a set of 4 radial vectors with the Angle of First Vector set to 45° and the Angle of Last Vector set to 135°, vectors will be placed at 45°, 75°, 105°, and 135°.

Default Angle of First Vector: 0°

Default Angle of Last Vector: 360°

- **Starting Radius/Ending Radius** — When generating a radial set of vectors, these values specify the distances (in pixels) from the anchor point of the shape for the start and end points of the vectors.

Default Starting Radius: 0

Default Ending Radius: 100

Range: 1 to 1000000

- **Length of Vectors** — This value specifies the length of each vector. All vectors within a set will be the same length. After the set is generated, the length of the vectors can be changed by dragging the shape handle of any of the vectors.

Default length: 100 pixels

Range: 3 to 10000

- **Spacing between vectors** — Specifies the vertical spacing in pixels between the generated vectors. This value is only relevant when generating parallel vectors, and has no effect when generating a radial set of vectors.

Default: 20 pixels

Range: 1 to 10000

- **Generate Vectors** — This button initiates the creation of the vector set. When selected, the current set of vectors will be replaced by a set of vectors as described by the above parameters. After creating the vector set, the vectors can be moved anywhere in the FOV. After generating the set, the number of vectors cannot be changed.

- **Gradient Threshold** — Sets the minimum value that the gradient must exceed for a point to qualify as an edge.

Default: 20

Range: 0 to 255

- **Sharpness** — Tunes the gradient detection based on blurred or sharp edges in the image. The number of pixels over which the gradient is computed changes based on this setting:

- ±1 Very Sharp (Default)

- ±2 Sharp

- ±3 Blurred

- ±4 Very Blurred

- **Result Type** — Specifies the type of output and provides a choice between Transitions and Lines:

- **Transitions (default)** — The output point will correspond to the location of the selected gradient transition.

- **Lines** — The output point will correspond to the midpoint between a pair of matched transitions.

For example, the midpoint between a light to dark transition and the next dark to light transition.

- **Line Type** — Selects the polarity of the lines:
 - **Dark (default)** — The vector is first scanned for a light to dark transition according to the Result Selection, and then finds a matching light to dark transition within the Minimum and Maximum Line Width range.
 - **Light** — The vector is first scanned for a dark to light transition according to the Result Selection, and then finds a matching dark to light transition within the Minimum and Maximum Line Width range.

- **Minimum Line Width** — Sets the minimum distance in pixels between matching edge transitions for the pair to be considered a line.

Default: 1 pixel

Range: 1 to 99999

- **Maximum Line Width** — Sets the maximum distance in pixels between matching edge transitions for the pair to be considered a line.

Default: 200

Range: 1 to 99999

- **Strict Edge Pairing** — When enabled, point on lines are found only when edges of opposite polarity are adjacent and within the Minimum and Maximum Line Width limits. For example, a point on a dark line is formed only when a light to dark edge is followed by a dark to light edge and the distance between these edges are within the Minimum and Maximum Line Width limits. When disabled, line points are found when an edge of opposite polarity is found within the min/max line width limits, even if the edges are not adjacent.

- **Transition Type** — Selects the polarity of the edges. Options are Light to Dark, Dark to Light or Both. This setting is valid only when the **Result Type** is set to Transitions.

Default: Light to Dark

- **Result Selection** — Selects which transition or line to output as a result:
 - **First (default)** — The first transition or line meeting the Gradient Threshold requirement along the direction of the vector is output as the result.
 - **Best** — The result corresponds to the transition with the highest gradient. This option is not available when **Result Type** is set to Lines.

- Last — The result corresponds to the last qualifying transition or line.
- Nth — The result corresponds to the Nth transition or line along the direction of the vector.
- All — All qualified transitions or lines are shown on the display and included in the Point List output. The individual vector output point datum Vector Point n will contain the nth transition or line result.
- **Nth result** — Specifies which transition or line to output when **Result Selection** is set to Nth. When **Result Selection** is set to All, this property specifies which transition will be recorded into the output point datums. This setting has no effect when **Result Selection** is set to other options.

Default: 1

- **Maximum number of Transitions** — Specifies the number of output point datums when the Number of Vectors is 1. Normally, the Vector Point n output points correspond to the set of vectors such that there is one output point for each vector. But when the Number of Vectors is 1, the Vector Point n outputs correspond to the transitions along the single vector. This property specifies the maximum number of output datums to provide from this vector.

Range: 1 to 1000

- **Graphics Level** — Enables various graphics options at runtime.

Default: Show Basic Graphics

- **Min Number of Edges** — When the number of points found falls outside of this range, the Vector Tool fails.
- **Max Number of Edges** — When the number of points found falls outside of this range, the Vector Tool fails.

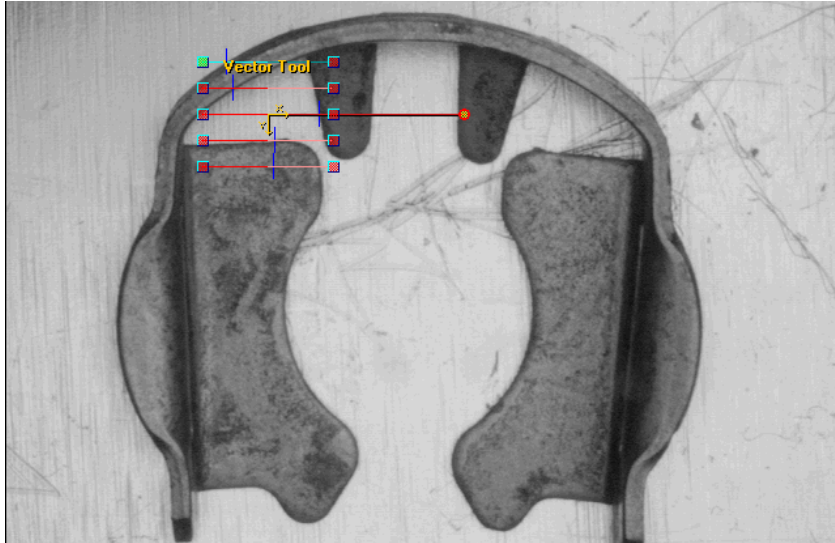
Training

No special training is required for Vector Tool; only generating and positioning the vector shapes.

Setting Up the Vector Shape

By default, when a vector tool is inserted, the tool will generate a set of 5 parallel vectors, as shown in Figure 7-44.

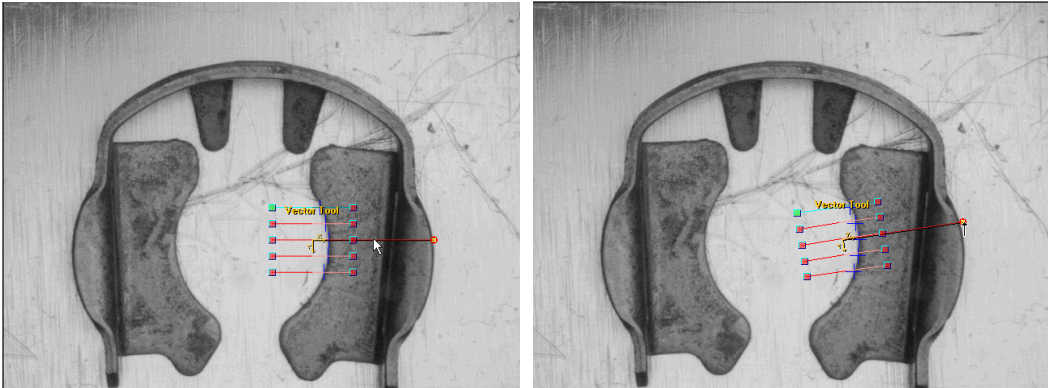
FIGURE 7-44. Five Parallel Vectors Generated



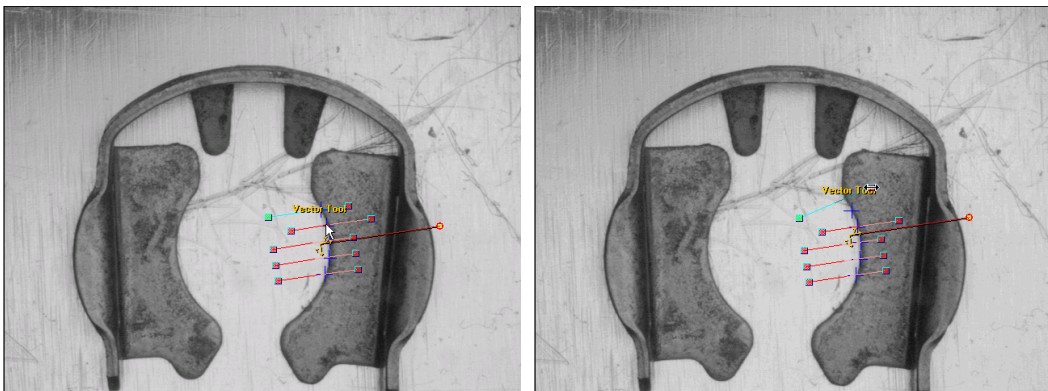
The blue vector indicates vector 0, the one next to it is vector 1, and so on. The direction of the vector is indicated by the two-tone brightness of the vectors. Vector 0 goes from light blue to dark blue, the other vectors go in the direction from red to pink.

Shape Manipulation

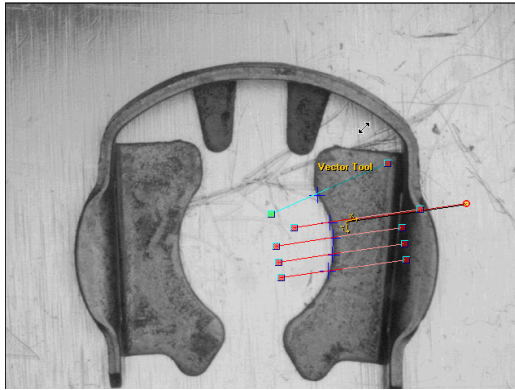
The set of vectors can be positioned as a group or individually. To move the group of vectors, click on the X,Y axis indicator or handle and drag, as shown in Figure 7-45. To rotate the group of vectors, click on the control point of the shape handle.

FIGURE 7-45. Clicking on the Control Point

To move an individual vector, click on the line of the vector and drag, as shown in Figure 7-46. To move just one end point of a vector for rotation, click on the handle and drag.

FIGURE 7-46. Moving an Individual Vector

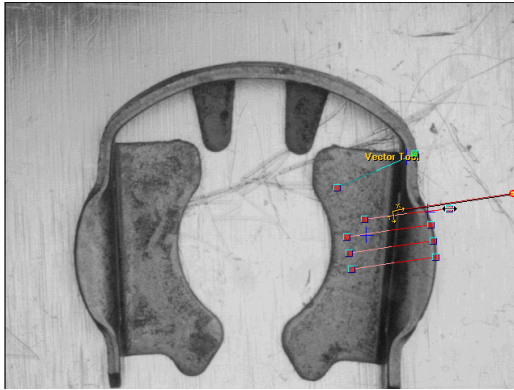
The length of the vectors can also be adjusted by clicking on the end point of a vector and dragging, as shown in Figure 7-47.

FIGURE 7-47. Adjusting the Length of the Vector

Note: When the length of any vector is changed, all vectors in the group also change to match. When the end point of a vector is clicked and dragged, both the angle of the single vector and the length of the group of vectors can be adjusted. To restrict the movement to only adjust either the angle of one vector or the length of the group of vectors, press and hold the Shift key before clicking on the vector handle. This will restrict the vector modification to rotation or stretching, depending on how the mouse is moved immediately after you click on the handle. A movement along the direction of the vector will restrict it to length modification. A mouse movement perpendicular to the direction of the vector will restrict it to rotation. To reverse the direction of the group of vectors, hold Shift, click on a control point of a vector, and drag it to the other side, as shown in Figure 7-47. The scan direction is now right to left.

Generating a Set of Vectors

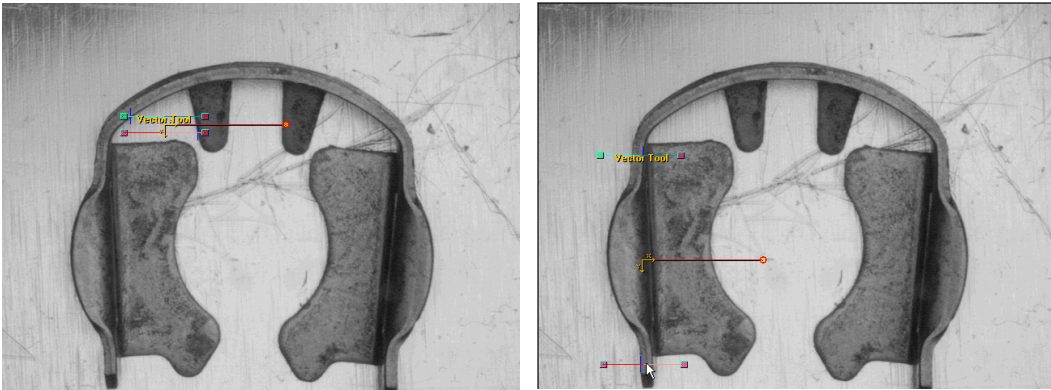
As shown in Figure 7-48, the placement of the vectors can be changed at any time. Only the number of vectors needs to be decided on before placement. There are some parameters for determining how many vectors to use and how they will initially appear. First, the pattern of the vectors should be decided, either a set of parallel vectors or a set of vectors in a radial pattern.

FIGURE 7-48. Changing the Placement of the Vectors

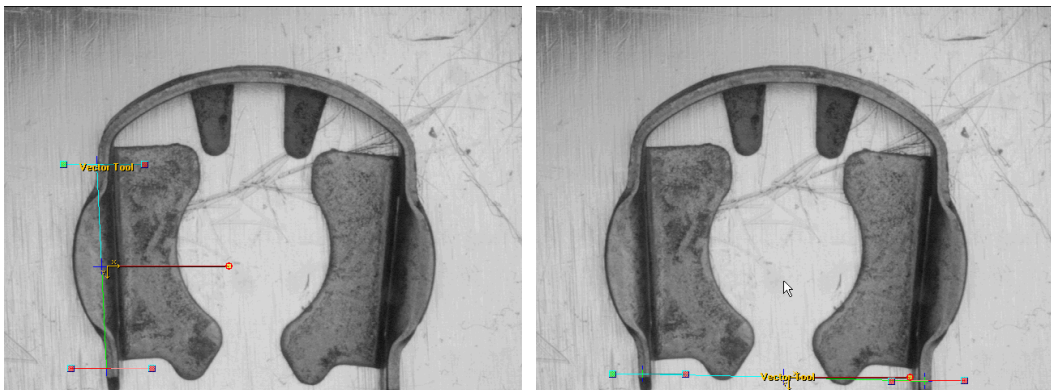
In the case of parallel vectors, you can specify how many vectors, and the initial length and spacing between them. If you will be placing the vectors manually, only the number of vectors is important, as the length and position can be changed after generating the set. For a simple two point measurement, set the **Number of Vectors** to 2, then click **Generate Vectors**.

Note: The previous set of vectors has been replaced with this new set. So, any vector positioning done before you click **Generate Vectors** will be lost.

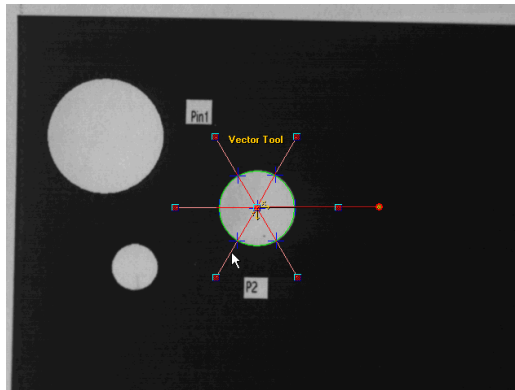
After you click **Generate Vectors**, the shape is immediately changed according to the vector shape properties, as shown in Figure 7-49. The individual vectors can then be positioned independently using the mouse.

FIGURE 7-49. Changing the Shape

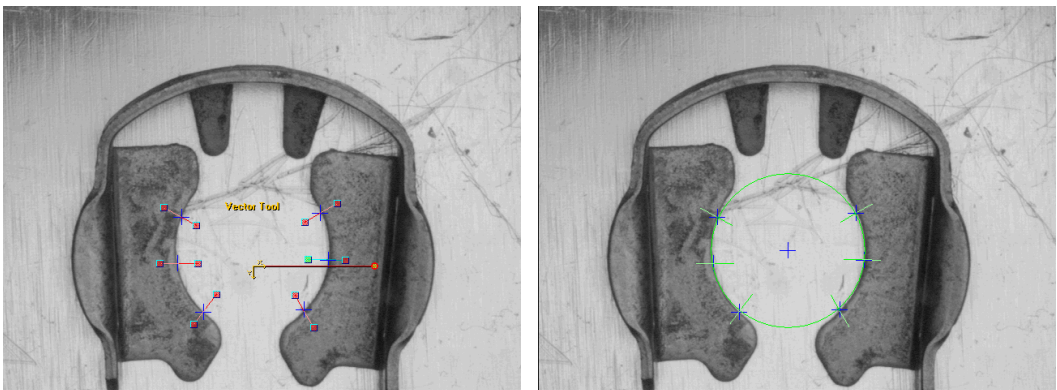
This can be used for simple line measurements with the addition of a LeastSquaresLine Fit step, as shown in Figure 7-50. It can similarly be used for a distance measurement with the use of a Dist2Pts step, as shown in Figure 7-50.

FIGURE 7-50. Simple Line Measurement with LeastSquaresLine

Vectors can also be generated in a radial pattern. This is useful when inspecting circular objects. The vectors will all start at the anchor point of the shape extending outward at the specified length, evenly spaced angularly based on the number of vectors. The point results from this vector set can be fit to a circle using the LeastSquaresCircle Fit step, as shown in Figure 7-51.

FIGURE 7-51. Vector Generated in Radial Pattern

As with the parallel vector example, the **Generate Vectors** button gives a starting set of vectors that can be repositioned after creation. When a circle to be measured is not complete, the vectors can be positioned along the arcs as in Figure 7-52.

FIGURE 7-52. Positioned Along the Arcs

Results

The results from the Vector Tool are listed below. The main result is the Point List that contains the edge points found. As with all steps, a pass/fail status is also output from the step.

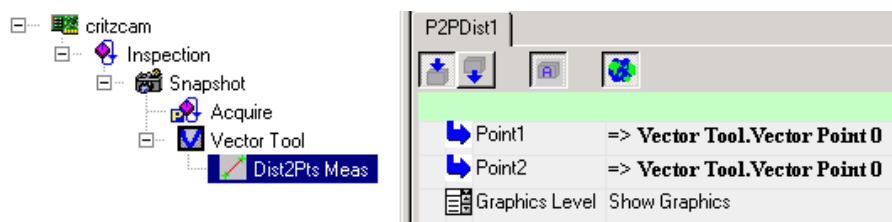
- Distance n ($n = 0 - 4$) — When Result Type is set to detect Lines, this set of Distance datums contains the width of the line at each point. When detecting Transitions, it is the distance from the start of the vector to the edge point.
- Number of Transitions — Total number of transitions found on all vectors. The number of points in the Point List result datum.
- Number of Vectors with Transitions — Number of vectors that contain at least one transition.
- Point List — This PtList datum contains the transitions from all of the vectors. When the Result Selection property is set to All, the Point List output will contain all of the transitions in order from the first transition of the first vector, followed by the remaining transitions of the first vector, then the second vector results and so on. The symbolic name is PtList.
- Status — Pass/Fail status of the tool.
- Transition Count n ($n = 0 - 4$) — This set of Integer datums contains the number of transitions or lines along each vector. The number of transitions found in the first vector is in result Transition Count 0, or symbolic name TransCnt0. The number of outputs in this set is equal to the number of vectors in this tool.
- Transition Results — This VectorResult datum, symbolic name Results, also contains the set of output points.
- Vector Point n ($n = 0 - 4$) — This set of Point datums contains the transition point from each vector. The result from the first vector in the set is Vector Point 0, or symbolic name P0. When there is a single vector in the set, this set of outputs will correspond to the transitions along the vector.

Using Results

The transition results from the Vector Tool can be fit to a line or a circle by inserting a LeastSquaresLine Fit, RobustLine Fit, or LeastSquaresCircle Fit step into the Vector Tool. The Fit Point List input datum of the fit step will be connected to the Point List output datum of the Vector Tool. This connection will pass the vector transition point information to the fitting step for processing.

Additionally, the individual transition points can be tolerated or used in measurement steps, such as the Dist2Pts Meas step. In the case of the distance measurement, the input points Point1 and Point2 will connect to the individual vector output points such as Vector Point 0 and Vector Point 1, as shown in Figure 7–53.

FIGURE 7–53. Points 1 and 2 Connected to Vector Points 1 and 2



FTP Image Logging of Pass/Fail Images

FTP Output Step

This step builds an FTP client into a job step. This client can connect to an FTP server to upload any image the user requires for storage. The user can upload the snapshot buffer directly, or any other buffer that has been processed by other tools. There can be multiple FTP steps in a job, each using a different server and directory.

Theory of Operation

FTP is a client/server protocol and as such, takes time to do its work. The client must connect to the server, log in, create or change to the proper directory, and upload a file. Files can be large depending on the camera. The larger the file, the longer the FTP time. Any use of this step may impact the time to complete one inspection. To help offset some of this time, there are two modes of operation. The first opens a connection at start, runs a keep-alive mechanism during the job run, and then closes the connection at stop. The keep-alive interval is user-configurable. The second mode is to connect to the FTP server every inspection.

If a transfer fails, the step will check to see if the connection is up. If not, it will attempt a re-connect and a re-try of the transfer. If it fails a second time, the FTP of the image will not occur, and will continue to the next inspection.

Potential Use Case

One potential way of using this step is to send failed inspections to a “failed” folder. The user fills in the Output Enable with the passing and failing criteria, and when the criteria is met, the inspection is transmitted to the FTP server. Be aware that the server must have enough storage to handle the number and size of files required. With large file sizes, over millions of inspections, one could easily fill the storage and lose inspection data.

Settings

Inputs

- **Input Buffer** – The buffer to FTP from this job (the image).
- **Server IP Address** – The IP address of the FTP server.
- **Port Connection** – The network port to use. Default is FTP standard 21.
- **UserName** – The user name for server login.
- **Password** – The password for server login.
- **Connect Every Cycle** – Specifies whether the FTP client will connect to the server only once before starting the job and disconnect once when stopping the job, or connect/disconnect every job cycle. Default is **false** or unchecked.
- **Keep-Alive Interval** – This is an advanced datum that will only display when clicking on the **A** button. Only used when **Connect Every Cycle** is false/unchecked. The interval at which FTP keep-alive packets are transmitted to keep the connection to the server from timing out. This value is in seconds. Default is **0** which is a 50 second interval. Requires a change to server settings regarding transfer timeouts.
- **Num Retries** – This is an advanced datum that will only display when clicking on the **A** button. This is used to set the number of times to retry an FTP operation upon a failure. The software will re-FTP the image this number of times, then continue on.
- **FTP Rate** – This setting has six options. **Inline**, **Maximum Rate w/ Drops**, **1 per Second**, **2 per Second**, **4 per Second**, **Every Inspection (No Drops)**.
- **Inline** – In this mode the data will be sent in-line with the inspection. This means the inspection will not continue until the FTP has completed. This will effect inspection rate.
- **Maximum Rate w/ Drops** – This will FTP the data as fast as possible, but will lose data when FTP is busy in a transfer. This will not impact the inspection rate.
- **1 per Second** – This will FTP the data that is current when the timer expires every second.
- **2 per Second** – This will FTP the data that is current when the timer expires every 500 milliseconds.
- **4 per Second** – This will FTP the data that is current when the timer expires every 250 milliseconds.

- **Every Inspection (No Drops)** – This will cause an inspection to block until the data can be sent via FTP. Once the data has been posted to send, the inspection continues.
- **Path** – The path of the destination. Default is `\microscan`.
- **FileName** – The file name for the destination file. Default is the symbolic name for the inspection and buffer. Ex “Insp1.Snapshot1”. This will be appended with a date/time stamp when running from the PC, or a timestamp that represents time since boot up when running on a camera. If using a color camera, this name will be appended with - **xx8-RawBayerData**, where **xx** will be either **RG**, **GR**, **GB**, or **BG**, depending on the camera type. This will save an 8-bit monochrome Bayer pattern image. Having this name appended will allow loading of the image back into the software for viewing in color.
- **FileType** – `.tif` or `.bmp`.
- **Output Enable** – An expression that evaluates true or false that allows transmission of the image to the FTP server. If true, an image will be transmitted to the server. If false, the image will NOT be transmitted.

Outputs

- **Output File Path** – Same as Input value.
- **Output File Name** – Same as Input value.

MS-Linkable Settings

Connect Every Cycle, Server Path, FileName, FileType, Error Code, Output File Path, Output File Name.

Training

None.

I/O Summary

The **FTP Output Step** provides an I/O summary in the **Status Bar** located at the bottom of the **Train Window**.

Inputs

Compile Error (**yy**): **xxx** when **Output Enable** compilation fails, or Input data selected.

xxx = Description of compilation error or OK.

yy = Error code.

Outputs: <string>

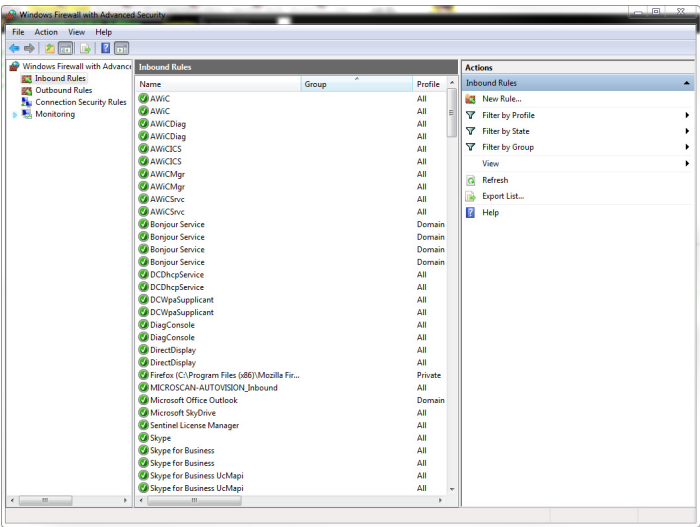
<string> = <error> for no errors, or displays the error string.

Server Setup

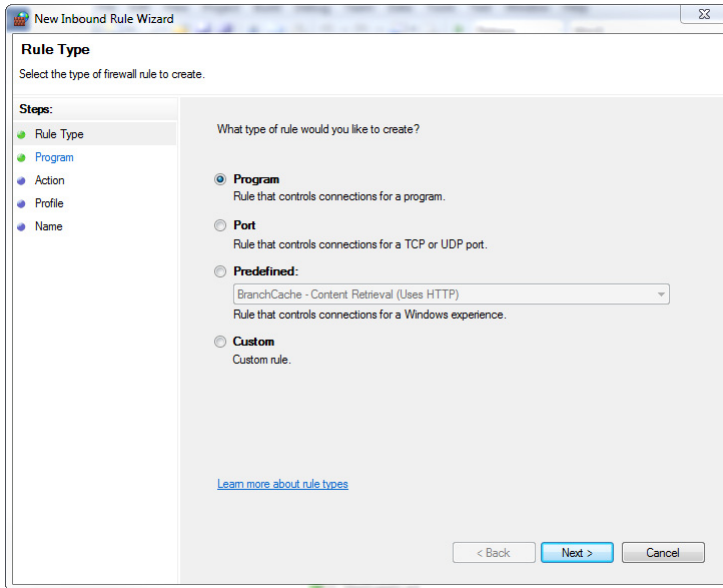
Omron Microscan recommends using the FileZilla FTP server. Install FileZilla then continue as described below.

Your PC may require changes to the firewall settings to allow a camera to connect to FileZilla. To allow FileZilla access through the firewall, do the following:

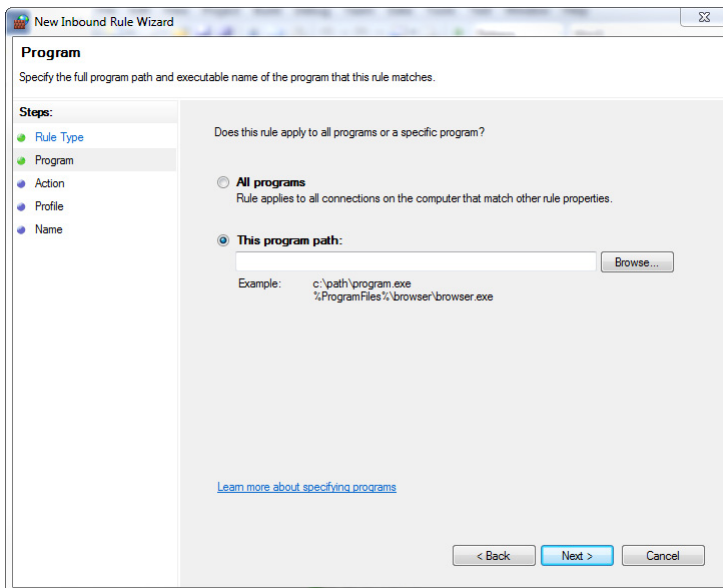
Go to windows button and type **windows firewall** in the search bar. It should bring up the windows firewall as shown below:



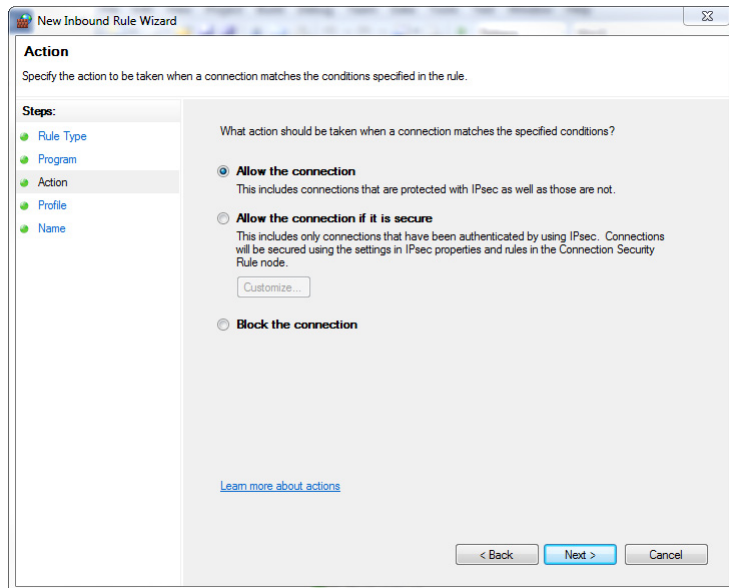
Click on **New Rule...** in the right pane. The following window will appear:



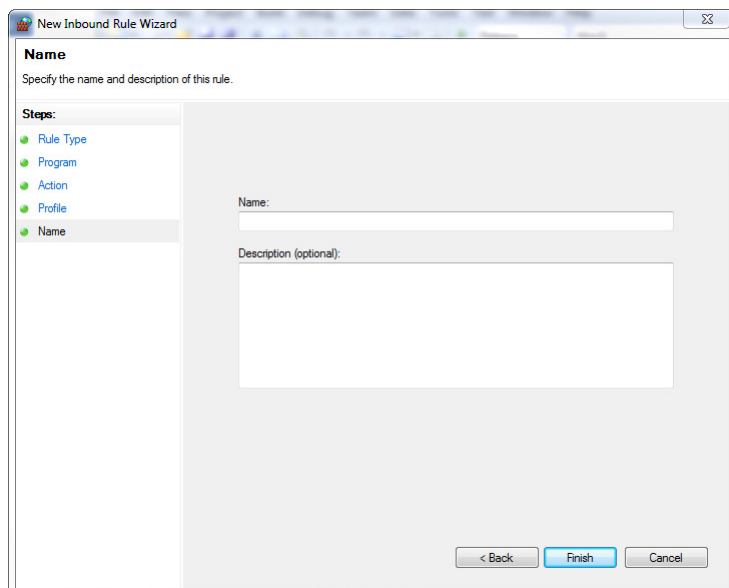
Be sure the program is selected then click **Next**. The following window will appear:



Click **Browse**, and go to **C:\Program Files(x86)\FileZilla Server**. Select **FileZilla Server.exe** and then click **Next**. The following is displayed:

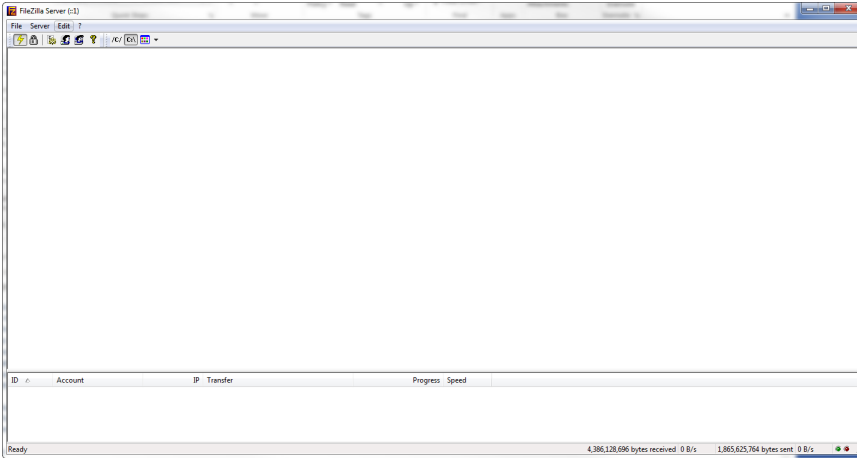


Be sure **Allow the Connection** is selected. Click **Next** through the next two windows until you see the following window:

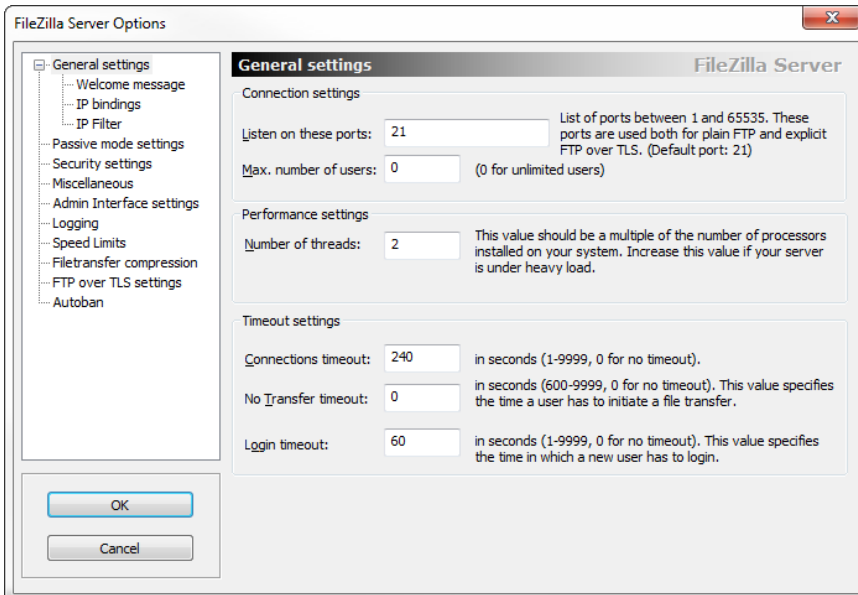


Enter any name you choose, and click **Finish**. This should allow FileZilla server through the windows firewall.

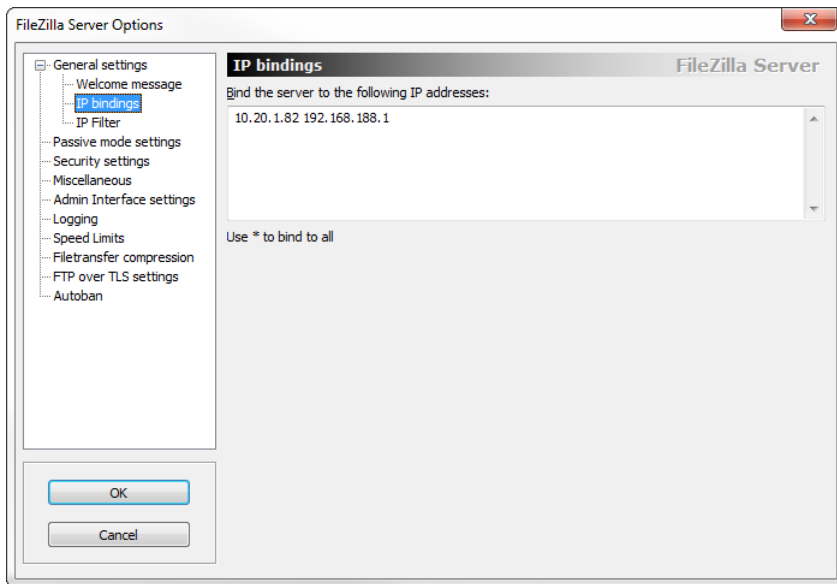
To configure the FileZilla server after install, click the **Edit** menu item, then **Settings**:



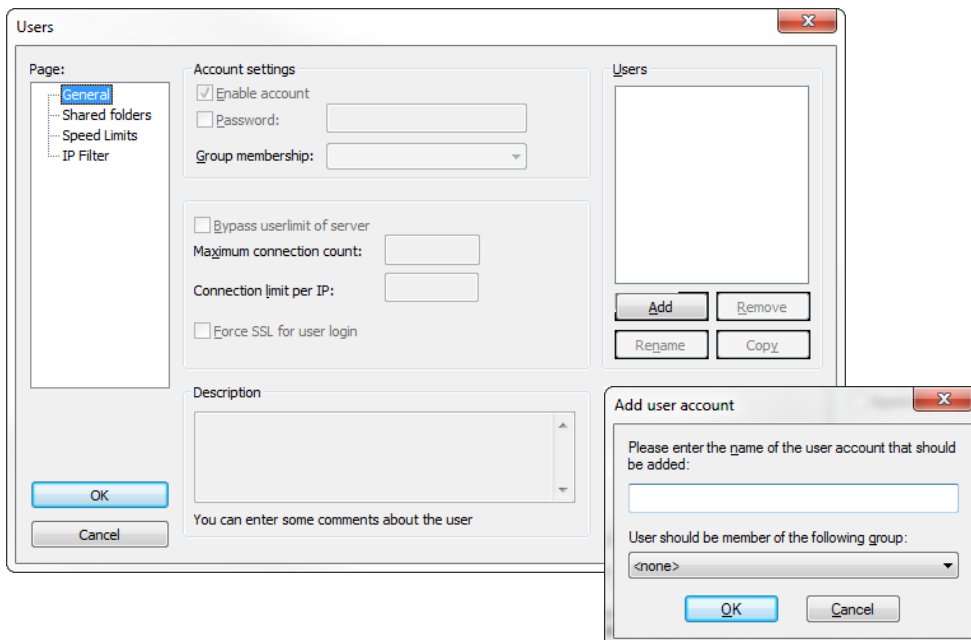
Be sure the **Listen On These Ports** is set to **21**. If you don't plan to not use the **Connect Every Cycle** option, **No Transfer Timeout** must be set to **0**. Make these changes, then click **OK**.



Now click **IP Bindings** and set the IP addresses to which you want to bind this server:

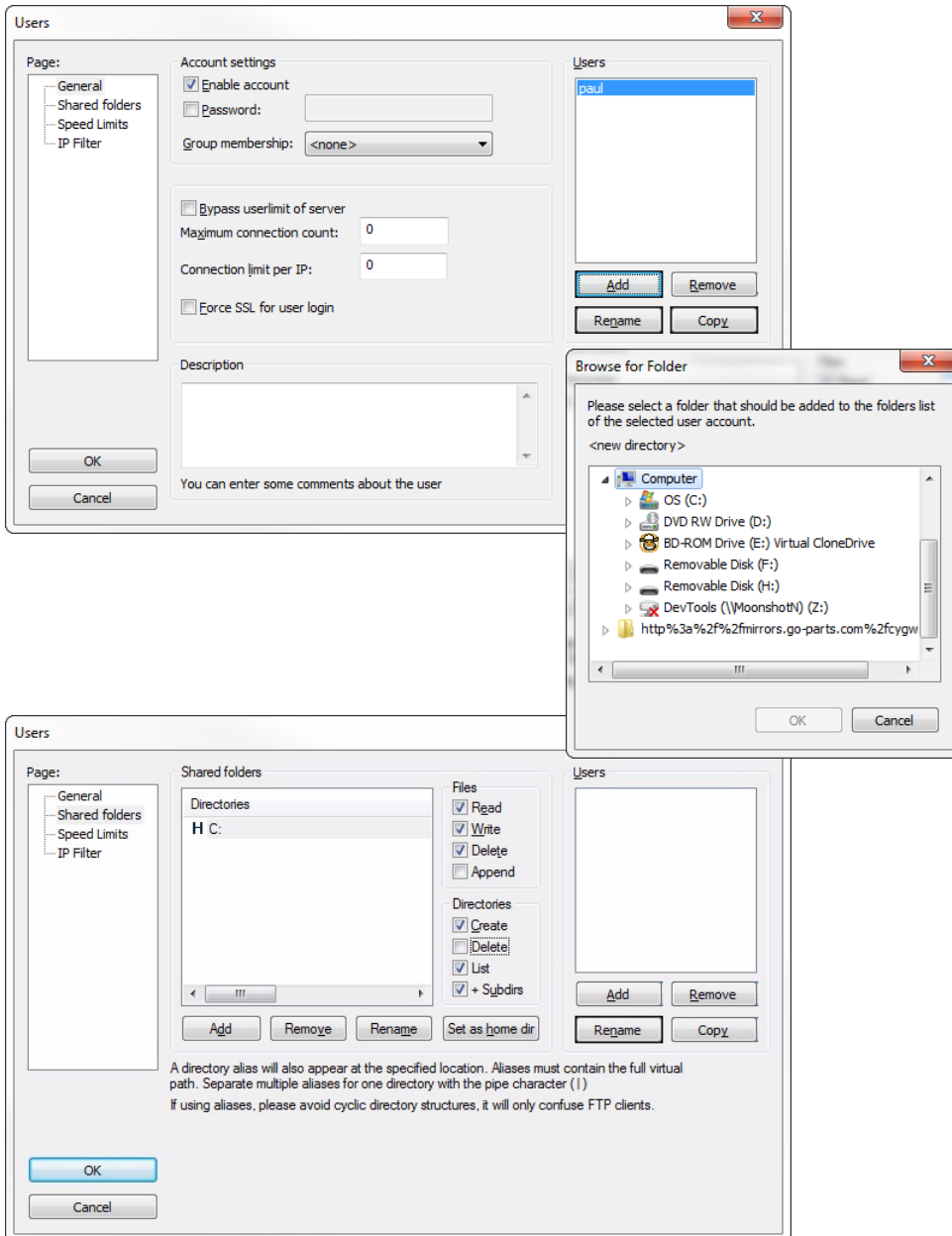


Now configure the users. Click **Edit**, then **Users**, then **Add**:



Click **OK**, then click on the **Password** checkbox and add a password for this user.

Now click **Shared Folders**, then **Add**. Browse for the folders to give this user access to, then click **OK**.

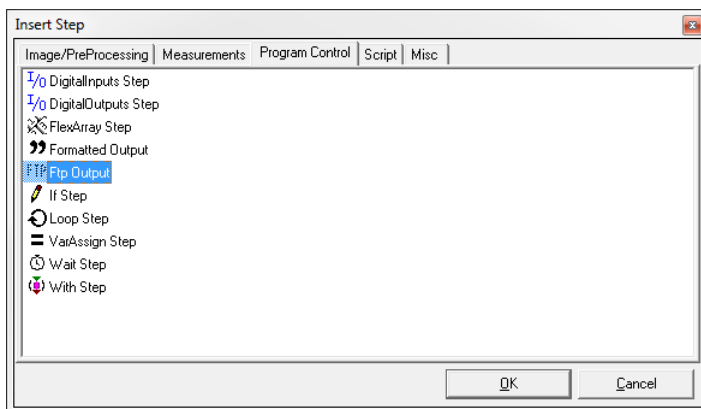


Now give this user permissions. Select (at least) **Read** and **Write** under **Files**, and select **Create** and **List** under **Directories**, then click **OK**.

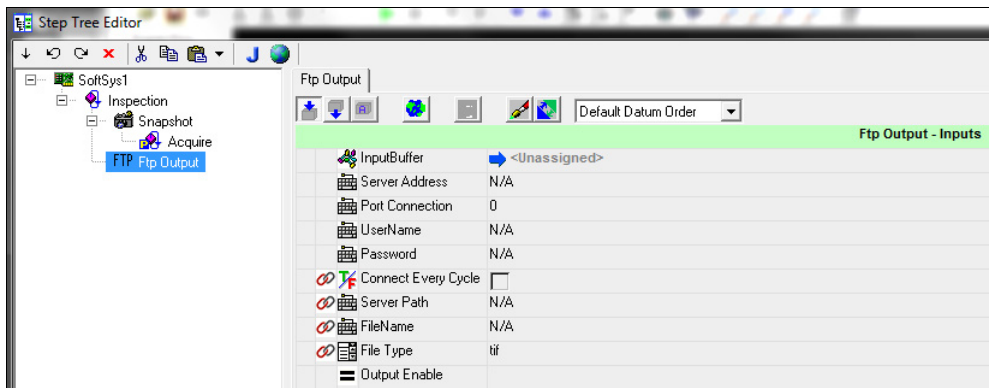
The server should now allow connections to the assigned IP address, and to the directories that were set. This can be tested by downloading the FileZilla client and trying to connect to the new server.

Configuring the FTP Output Step

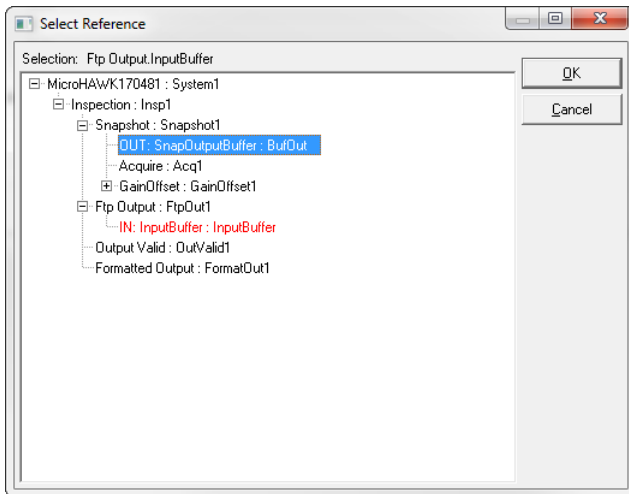
Bring up **Visionscape FrontRunner**, then connect to a device. Select a job or create a new job. Click **Editor**. This brings up the **Step Tree Editor**. Insert a new step in the position required. Then click **Program Control**.



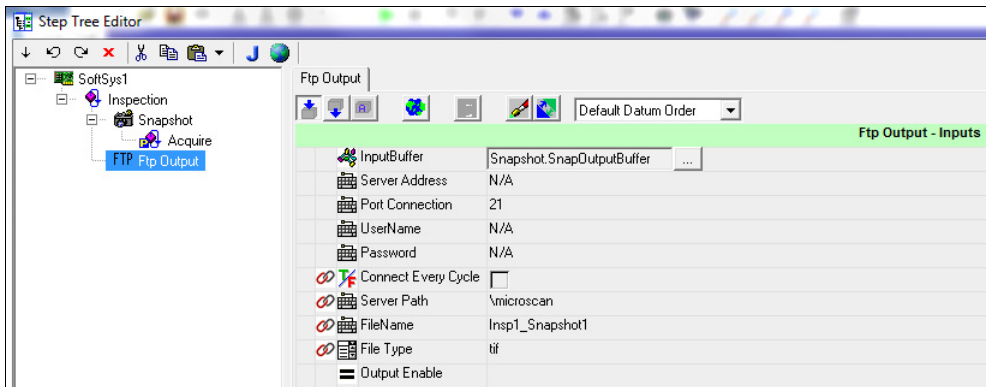
Select the **FTP Output** parameter:



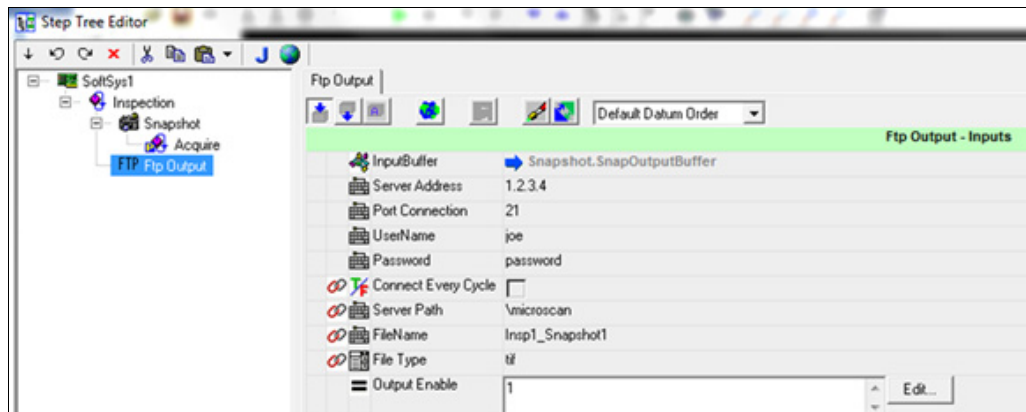
Select **Input Buffer**, then select the image to FTP, then click **OK**:



After the **Input Buffer** is selected, Omron Microscan default values are populated for **Port Connection**, **Connect Every Cycle**, **Keep-Alive Interval**, **Server Path**, **FileName**, and **File Type**.



Enter **Server Address**, **UserName**, and **Password**. Any default setting can be changed now as well. Add an expression to the **Output Enable** field. If no expression is required, a 1 value will allow all images to be sent.



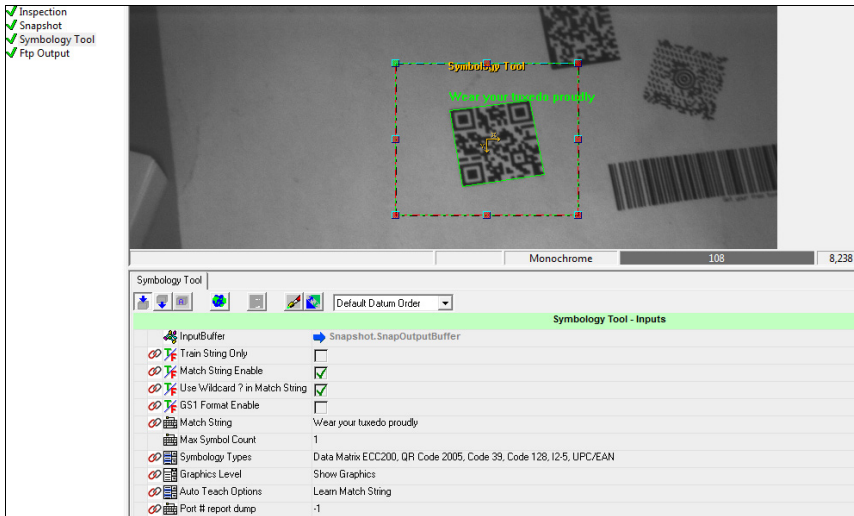
Exit the editor, and run an inspection to be sure it is connected properly to the FTP server. At the bottom of the screen, errors will be placed in the **Outputs:** area. Go to the server and switch to the directory provided. The image file with an added date/time (if run from the PC) or a time stamp (if run from a camera) will be seen. Open the file and check to be sure it is what was expected. If so, download and run the job.

Output Enable Examples

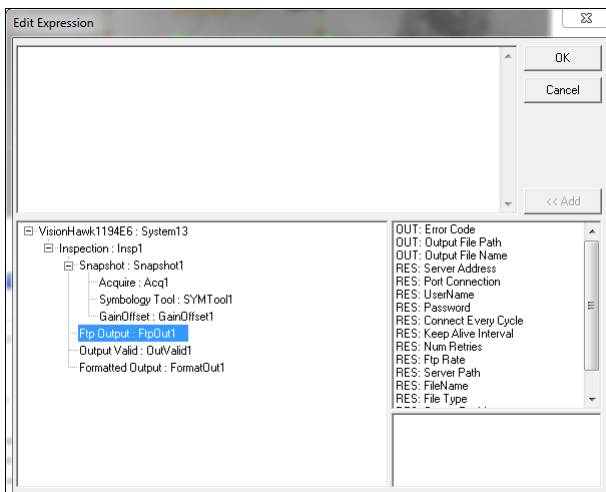
See page 8-59 of the *Visionscape Tools Reference Manual* for information regarding **Expressions**.

FTP images for symbology decodes that match a match string.

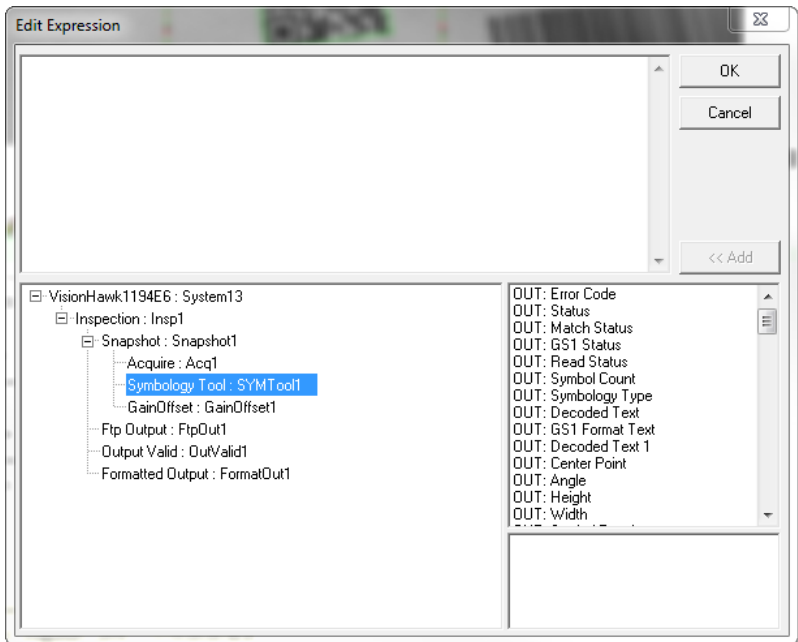
Set up your symbology match. Click **Match String Enable**, then enter the **Match String**.



Click in the **Output Enable** box, then click the **Edit** button to the right of **Output Enable**. The following window is displayed:

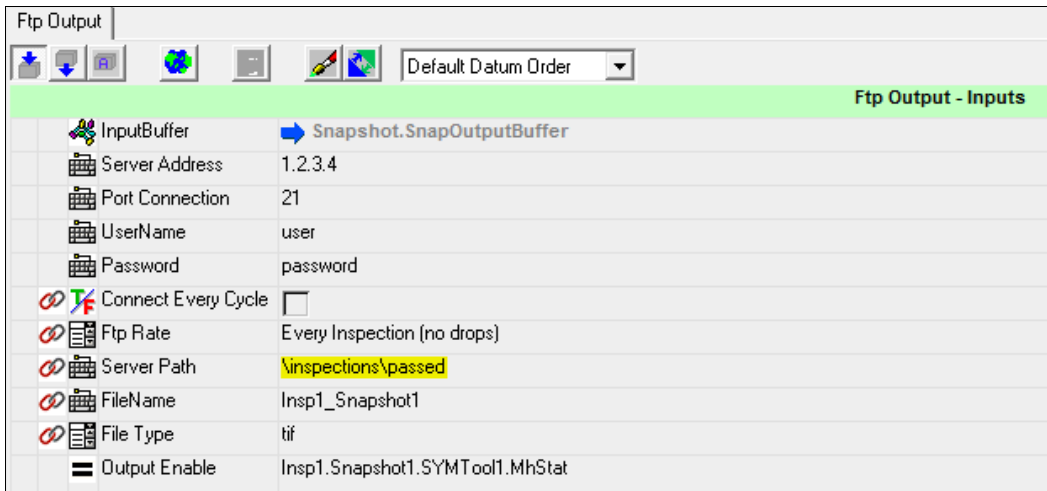


Now select **Symbology Tool: SYMTool1**:



Select **Match Status** in the right pane, then click the **<<Add** button, then click **OK**.

Your window should now look as follows, and if the Symbology Tool match string matches the decoded text, it will FTP the image to the **inspections\passed** directory:



FTP all failed symbology matches.

Use exactly the same steps as above, then add a ! in front of the expression as follows: **!Insp1.Snapshot1.SYMTTool1.MhStat**. This will FTP all images that do not match the symbology match string to the server. The FTP screen should look as follows:

Ftp Output - Inputs	
InputBuffer	Snapshot.SnapOutputBuffer
Server Address	1.2.3.4
Port Connection	21
UserName	user
Password	password
Connect Every Cycle	<input type="checkbox"/>
Ftp Rate	Every Inspection (no drops)
Server Path	\\inspections\failed
FileName	Insp1_Snapshot1
File Type	tif
Output Enable	!Insp1.Snapshot1.SYMTTool1.MhStat

FTP an image if the entire inspection fails.

Use a two inspection model as follows:

Ftp Output - Inputs	
InputBuffer	Snapshot.SnapOutputBuffer
Server IP Address	1.2.3.4
Port Connection	21
UserName	user
Password	password
Connect Every Cycle	<input type="checkbox"/>
Ftp Rate	Maximum Rate w/drops
Server Path	\\inspections\failed
FileName	Insp2_Snapshot1
File Type	tif
Output Enable	!Insp1.Status

Geometric Fitting and Measurement Tools

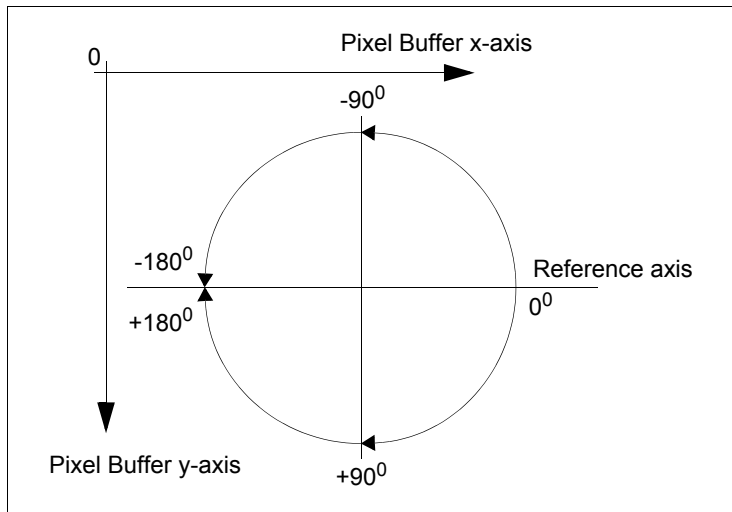
Visionscape provides tools for line and circle fitting, geometric measurements, and tools for tolerancing the measurements. These measurements generate the following results: Area, Angle, Distance, Point, Line, Integer, String, and Double. All results can be reported in calibration units except the scalar results Integer and Double, and the String result. In addition to the predefined measurement types, customized measurements can be defined by mathematical expressions.

Measurement Types

Angle

Angles are measured in degrees clockwise from the x-axis. In buffer space, the x-axis direction is horizontal from left to right on the pixel buffer. The y-axis direction is vertical from top to bottom on the pixel buffer, as shown in Figure 8–1.

FIGURE 8–1. Angle Measurement



The singular point that separates the positive angle range from the negative range can be arbitrary from 0° to 360° . By default, the singular point is set at 180° . The singular point can be adjusted through access routines from Visual Basic programming only.

In World space, the x-axis and y-axis directions are defined by the camera calibration.

Positive angles are measured from the positive x-axis toward the positive y-axis. This direction is clockwise in buffer space, and either clockwise or counter-clockwise in World space, depending on the camera calibration and the **View from Behind** parameter at calibration time.

Angles can also be computed between two lines, in which case the first line always acts as the reference line (i.e., the X axis for the angle), and the second line defines the angle. For more information, see “Line” on page 8-3.

Note: Internally, an Angle result stores the angle, along with two lines that define the reference and measure lines this angle is computed from. These lines are in the system of coordinates of the Angle result itself.

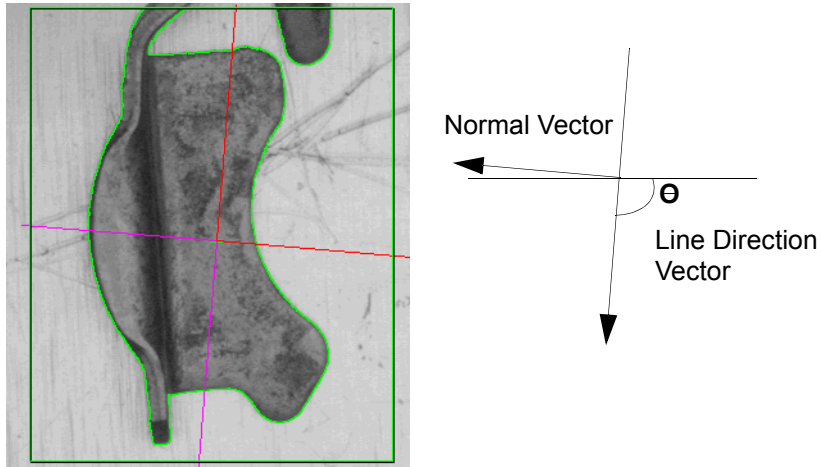
Distance

Distances are measured in pixels in buffer space and in calibrated units in World space. Horizontal distance measures the difference between the x-coordinates of two points, and vertical distance measures the difference between the y-coordinates of two points.

Note: A Distance result stores two data points (x,y), which represent the start and end points of the distance segment. These accurately calculate distances in World coordinates.

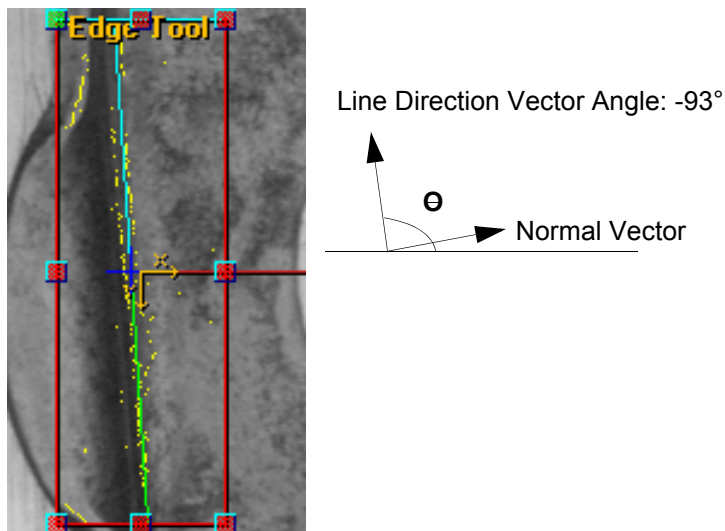
Line

A line is measured in degrees from the positive x-axis to the line. Because a line extends in both directions, the line's angle can either be n° or $(n \pm 180^\circ)$. As a result, the line is said to have two directions. Based on the tool that generated the Line result, the line will be pointing in one direction or the other. As a rule, lines will be pointing in a direction such that their normal vector and the direction vector form a right-handed system of coordinates, as shown in Figure 8–2. In addition, all lines drawn on the screen are color coded, where one side of the line is darker than the other. Figure 8–2 shows a Blob major axis line.

FIGURE 8-2. Lines Example

The major axis line points from the dark to the light section of the segment; therefore, the angle of the line is around 100° (as opposed to -80°).

Lines generated by the Edge or FastEdge tools are such that their normal vector points in the direction of the light area of the edge. As shown in Figure 8-3, the line direction is pointing up: dark to light.

FIGURE 8-3. Line Pointing Down - Dark to Light

Typically, lines are used as inputs to other Visionscape measurement tools including: Dist2Pts Meas, IntersectLines Meas, BisectLines Meas, PointToLine Meas, and Tolerance Meas.

Point

A point is measured in pixels in buffer space or in calibrated units in World space based on its x- and y-coordinates. The upper left corner of the image buffer represents the origin of the buffer space coordinate system. The camera calibration defines the origin of the World coordinate system, typically in the center of the calibration target.

In addition, points can optionally store an orientation in degrees and a size based on the Tool that produced the point as a result. For example, a Blob center also stores the major axis orientation.

Typically, points are used as inputs to other Visionscape measurement tools including: Dist2Pts Meas, PointToLine Meas, and PointTolerance Meas.

Area

Area is measured in number of pixels in buffer space or in calibrated units in World space. For example, a camera, calibrated as 1 pixel equaling 0.1 inches, would result in an area of 1 pixel equaling 0.01 square inches in World space.

Note: An area result stores a point that is usually in the center of the area being measured. This point estimates the unit area when converting the result between different coordinate systems, such as Calibrated World space vs. Pixel space.

Integer and Double

Integer and Double are non-dimensional measurement types. Measurements of type Integer or Double remain the same in buffer space as in World space. Examples of Integer or Double types are the roundness feature for a Blob or the number of transition points for an Edge.

Customized Measurement

A customized measurement can be created by a user-defined mathematical expression involving measurements of known types. Refer to “Expressions” on page 8-59 for more information.

Geometric Fitting

LeastSquaresCircle Fit

This tool computes the circle that fits through a set of points using a least squares fit. LeastSquaresCircle Fit fits circles to edge points extracted from an image. Edge points are contained in a point list datum. The Edge Step is an example of a step that outputs a point list datum. The input to the LeastSquaresCircle Fit is the point list datum containing the set of points to fit to. The outputs are:

- The circle radius
- Point that is the center of the circle
- Number of points used in the fit
- Fit error
- Mean, min and max deviations of the points from the circle
- Minimum radius
- Maximum radius
- Furthest point from the center
- Closest point to the center

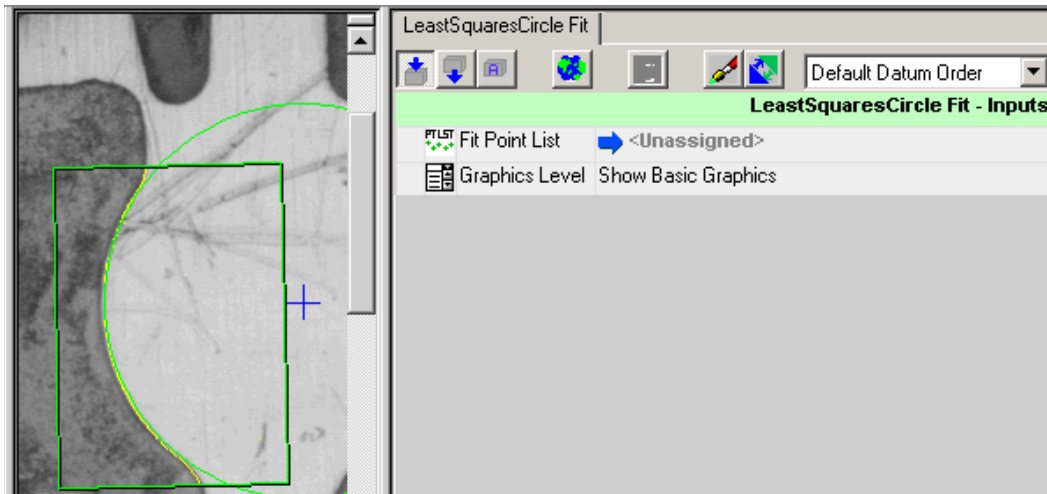
Other Steps Used

LeastSquaresCircle Fit can only serve as a child step of another step that produces a point list. Two such steps are:

- EdgeTool — The Edge Tool finds straight, circular, or elliptical edges or line segments in the ROI. The edges are output in a point list.
- Vector Tool — The Vector Tool finds transition edge points in the image along predefined directions called probes.

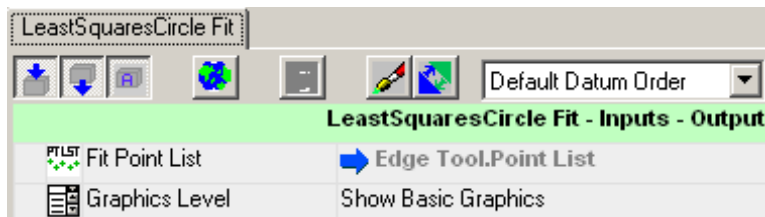
Theory of Operation

LeastSquaresCircle Fit finds a circle through the points using a least square algorithm. Figure 8–4 displays an example of a fitted circle, shown in green, which correctly follows the parts perimeter.

FIGURE 8-4. LeastSquaresCircle Fit — Example

Description

LeastSquaresCircle Fit allows editing through the LeastSquaresCircle Fit properties page, as shown in Figure 8-5.

FIGURE 8-5. LeastSquaresCircle Fit Properties Page

Settings

- **Fit Point List** — Allows selection of the LeastSquaresCircle Fit edge points. Upon insertion, LeastSquaresCircle Fit searches its parent step for an output point list and connects its input datum to this output datum. Once located, the input datum is displayed. You can change this to another point list datum if necessary.
- **Graphics Level** — Selects the graphics to be displayed at the completion of step execution.

- Show Basic Graphics (Default) — The circle to which the points were fit is displayed in green and a blue cross is displayed at the center of the circle.
- Show Detail — The graphics associated with Show Basic Graphics are drawn. In addition, a blue cross at the two yellow lines perpendicular to the circle at the two farthest points are drawn.
- Show None — No graphics are drawn.

Training

None.

Results

The result datums returned by LeastSquaresCircle Fit include:

- Status (Status) — Status datum showing whether the step passed or failed
- Fit Point (Pt) — Fitted circle center point
- Radius Datum (Radius) — Circle radius distance
- No of Points used in Fit (UsesPts) — Number of points used in the fit
- Fit Error (FitErr) — Least square fit error
- Mean Deviation (MeanDev) — Mean deviation of the points from the circle
- Minimum Deviation (MinDev) — Distance from the circle of the point closest to the circle center
- Maximum Deviation (MaxDev) — Distance from the circle of the point furthest from the circle center
- Minimum Radius (MinRad) — Distance datum indicating the minimum radius found in the point set
- Maximum Radius (MaxRad) — Distance datum indication the maximum radius found in the point set
- Point Furthest from Center (MaxPt) — Point furthest from the circle center
- Point Closest to Center (MinPt) — Point closest to the circle center

I/O Summary

LeastSquaresCircle Fit provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Point count: ddd

Outputs: Circle: X=xxx Y=xxx R=xxx MeanDev=xxx Max+Dev=xxx
Max-Dev=xxx

Where: ddd = Number of points in the input point list
xxx = X,Y coordinates of the center point of the circle, or radius R,
or deviations in pixels

LeastSquaresLine Fit

This tool computes the line that fits through a set of points using a least squares fit. The input is the set of points that is contained in a point list datum. The outputs include the line and the point that is the geometric center of the line within the ROI. Other outputs include the quality of the fit, the closest and furthest points from the line, and the amount of deviation.

Other Steps Used

LeastSquareLine Fit can only serve as a child step of another step that produces a point list. Two such steps are:

- EdgeTool — The Edge Tool finds straight, circular, or elliptical edges or line segments in the ROI. The edges are output in a point list.
- Vector Tool — The Vector Tool finds transition edge points in the image along predefined directions called probes.

Theory of Operation

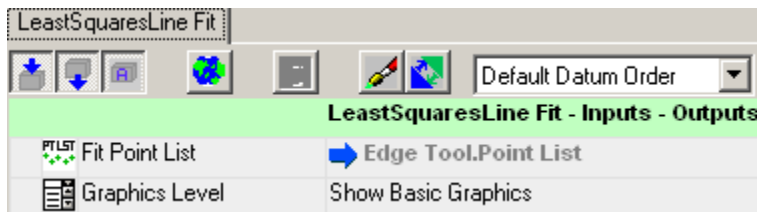
LeastSquaresLine Fit fits straight lines to edge points extracted from an image. It is normally placed inside the Edge Tool. Edge points are extracted by Edge Tool and are contained in a point list datum. A set of points (i.e., a point list datum) coming from a step other than the Edge Tool is also allowed as input.

LeastSquaresLine Fit finds the equation of the line through the points using a least squares algorithm. Unlike a robust algorithm, it will use all the edge points in the line fit. Figure 8–6 shows LeastSquaresLine Fit. The fitted edge, shown in green, does not lie on the parts edge. All the edge points are included in the computation.

FIGURE 8-6. LeastSquaresLine Fit — Edge

Description

LeastSquaresLine Fit allows editing through the LeastSquaresLine Fit properties page, as shown in Figure 8-7.

FIGURE 8-7. LeastSquaresLine Fit Properties Page

Settings

- **Fit Point List** — Allows selection of the LeastSquaresLine Fit edge points. Upon insertion, LeastSquaresLine Fit searches its parent step for an output point list and connects its input datum to this output datum. Once located, the input datum is displayed. You may change this to another point list datum, if necessary.
- **Graphics Level** — Selects the amount of graphics to be displayed at the completion of step execution:

- Show Basic Graphics (Default) — The line to which the points were fit is displayed in dark blue to light blue, and a blue cross is displayed at the center point of the line. The line direction goes from dark to light.
- Show Detail — The graphics associated with Show Basic Graphics are drawn. In addition, a blue cross at the two farthest points of the line and a yellow line perpendicular to the fit line (dark blue -> light blue) at the two farthest points are drawn.
- Show None — No graphics are drawn.

Training

None.

Results

The result datums returned by LeastSquaresLine Fit include:

- Status (Status) — Status datum showing whether the step passed or failed.
- Fit Point (Pt) — Contains the fitted line center point. This center point is located at the midpoint between the two points of intersection with the ROI.
- Line Datum (Line) — Contains the fitted line equation.
- Line Angle Datum (Angle) — Angle datum that contains the angle of the fitted line.
- No of Points used in Fit (UsesPts) — Int datum indicating how many points were used in the line fit.
- Fit Error (FitErr) — Distance datum indicating the fit error, defined as the square root of the sum of all the deviations.
- Mean Deviation (MeanDev) — Distance datum indicating the mean distance for all points to the line.
- Minimum Deviation (MinDev) — Distance datum indicating the distance from the closest point to the line.
- Maximum Deviation (MaxDev) — Distance datum indicating the distance from the furthest point to the line.
- Furthest point after line (MaxPt) — Point datum indicating the furthest point above the line.

- Furthest point before line (MinPt) — Point datum indicating the furthest point below the line.

I/O Summary

LeastSquaresLine Fit provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Point count: ddd

Outputs: Line: A=xxx B=xxx C=xxx N=nnn Point: X=xxx Y=xxx

Where: ddd = Number of points in the input point list
nnn = Number of points used in the line fitting
xxx = X,Y coordinates of center point of line, or Line
equation coefficients: A, B, C

RobustCircle Fit

This tool computes the circle that fits through a set of points using a proprietary Omron Microscan algorithm. RobustCircle Fit fits a circle to edge points extracted from an image. Edge points are contained in a point list datum. The Edge Step and Vector Tool are examples of steps that output a point list datum.

In Table 8–1 and Table 8–2, fitted points refers to the set of points used (not discarded) by the circle fit. A fitted point is any single point in that set.

TABLE 8–1. RobustCircle Fit Step Inputs

Symbolic Name	Datum Type	Description
Calculations To Perform	Enum	Selects the statistics computed each time the tool is run.
CircleCompleteness	Enum	Selects your criteria for circle completeness.
CircleFitMethod	Enum	Selects fitting method: Normal is used for most cases. Alternate1 may be useful in cases where there are many outlier points.
CirclePolarity	Enum	Selects the polarity of the circle and the background in the image.
CircleRangeMethod	Enum	Method for computing a likely range for the center and radius of the circle given the input points. The Classic method computes the variance in the X,Y coordinates of the input points. The 3PointGroups method splits the data up into 3 groups and takes one point from each set to produce a 3pt circle fit. The median and variance of these centers and radii compute the range.
CircleRoughness	Enum	Selects your criteria for how closely the resultant circle must follow the input points.
EnableCenterFilter	Status	Checkbox enabling filtering of the location of the fitted circle center.
EnableRadFilter	Status	Checkbox enabling filtering of input points by radius.
Location XRange Value	Distance	Current allowable range of X-coordinate for fitted circle.
Location YRange Value	Distance	Current allowable range of Y-coordinate for fitted circle.

TABLE 8–1. RobustCircle Fit Step Inputs (continued)

Symbolic Name	Datum Type	Description
LocXRange	Distance	Allowable range of X-coordinate for fitted circle center when location filtering is active.
LocYRange	Distance	Allowable range of Y-coordinate for fitted circle center when location filtering is active.
Max Radius Value	Distance	Current maximum allowable radius for fitted circle.
MaxRadius	Distance	Maximum allowable radius used when filtering input points by radius.
Min Radius Value	Distance	Current minimum allowable radius for fitted circle.
MinNumPts	Integer	Minimum number of points that must be used in circle fit.
MinRadius	Distance	Minimum allowable radius used when filtering input points by radius.
NomCenterPoint	Point	Point indicating the nominal location of fitted circle center. Used when radius and/or location filtering is active.
Nominal CenterX Value	Distance	Current X coordinate of the nominal center point for fitted circle.
Nominal CenterY Value	Distance	Current Y coordinate of the nominal center point for fitted circle.
PointList	Point List	The set of points to fit to.
UpdateNominalRuntime	StatusDm	If enabled, the nominal input point, radius range and location range are updated each time the tool is run.

TABLE 8–2. RobustCircle Fit Step Outputs

Symbolic Name	Datum Type	Description
CenterPoint	Point	Center Point of the fitted circle.
FitErr	Distance	Circle Fit Error (least squares fit error?).
Radius	Distance	Radius of the fitted circle.
UsesPts	Integer	Number of points that were used in the circle fit.
FITTED points and statistics		
MaxFitAbs	Distance	The maximum distance of any fitted point from the circle.

TABLE 8–2. RobustCircle Fit Step Outputs (continued)

Symbolic Name	Datum Type	Description
MaxFitDev	Distance	The distance from the circle to the fitted point furthest from the circle center.
MaxFitPt	Point	The fitted point furthest from the center of the circle.
MaxFitRad	Distance	The maximum radius of the circle (i.e., distance from circle center to fitted point furthest from the circle center).
MeanFitDev	Distance	The average deviation of the fitted points from the circle.
MinFitDev	Distance	The distance from the circle to the fitted point closest to the circle center.
MinFitPt	Point	The fitted point closest to the center of the circle.
MinFitRad	Distance	The minimum radius of the circle (i.e., distance from circle center to fitted point closest to the circle center).
CANDIDATE points and statistics		
MaxAbs	Distance	The maximum distance of any input1 point from the circle.
MaxDev	Distance	The distance from the circle to the input1 point furthest from the circle center.
MaxPt	Point	The input1 point furthest from the center of the circle.
MaxRad	Distance	The maximum radius of the circle (i.e., distance from circle center to fitted point furthest from the circle center).
MeanDev	Distance	The average deviation of the input points from the circle.
MinDev	Distance	The distance from the circle to the input1 point closest to the circle center.
MinPt	Point	The input1 point closest to the center of the circle.
MinRad	Distance	The minimum radius of the circle (i.e., distance from circle center to fitted point closest to the circle center).

Other Steps Used

The RobustCircle Fit tool can be inserted into a tool that produces a point list:

- **EdgeTool** — The Edge Tool finds straight, circular, or elliptical edges or line segments in the ROI. RobustCircle Fit can serve as a child step of an Edge Tool. In this case, Edge Tool must be inserted into your Visionscape Job before you have access to RobustCircle Fit.
- **FastEdgeTool** — The Fast Edge Tool finds straight, circular, or elliptical edges or line segments in the ROI. RobustCircle Fit can serve as a child step of a Fast Edge Tool. In this case, Fast Edge Tool must be inserted into your Visionscape Job before you have access to RobustCircle Fit.
- **VectorTool** — The Vector Tool finds straight, circular, or elliptical edges or line segments in the ROI. RobustCircle Fit can serve as a child step of a Vector Tool. In this case, Vector Tool must be inserted into your Visionscape Job before you have access to RobustCircle Fit.
- **Custom Step / Custom Vision Tool** — RobustCircle Fit can serve as a child step of a Custom Step or Custom Vision Tool which outputs a point list datum.

Theory of Operation

RobustCircle Fit Step fits a circle through the points using a proprietary Omron Microscan algorithm. Whereas the least squares circle fit includes all points in the circle fit, the robust circle fit attempts to exclude points that are not part of the circle.

To assist the robust circle fit in finding the correct circle, user inputs provide stricter control over inputs and results. **Enable Radius Filter** enables filtering of the input points based on their location with respect to a nominal circle center point. Points that are obviously not part of the circle can be discarded immediately. **Enable Location Filter** restricts the location of the circle. This lets the circle fitting code know that the circle center must fall within a particular region. These restrictions are particularly useful when multiple circular objects are present.

When radius filtering or location filtering is enabled, a nominal center point is needed along with either a radius range, a location range, or both. There are two sets of datums in the properties page used for filtering parameters. One set consists of input datums; the other set consists of resource (user settable) datums. The input datums have precedence. If the input datum is connected to another datum in the Job, then the value of that (connected) datum is used. If the input datum is not connected, then the value from the resource datum is used.

For example, with radius filtering enabled, the Nominal Center Point could be connected to a Blob Filter's output for the center of a blob and the min and max radius values read from the resource datums. The center point can be updated each run (from the blob tool) by enabling `UpdateNominalRuntime`; this value overrides any value in the resource datum for Nominal Center Point.

Note: The RobustCircle Fit Step supports “dynamic placement” in this configuration.

The min and max radius inputs are not connected, so the value from the resource datums are used.

The RobustCircle Fit Step has a shape that is useful for:

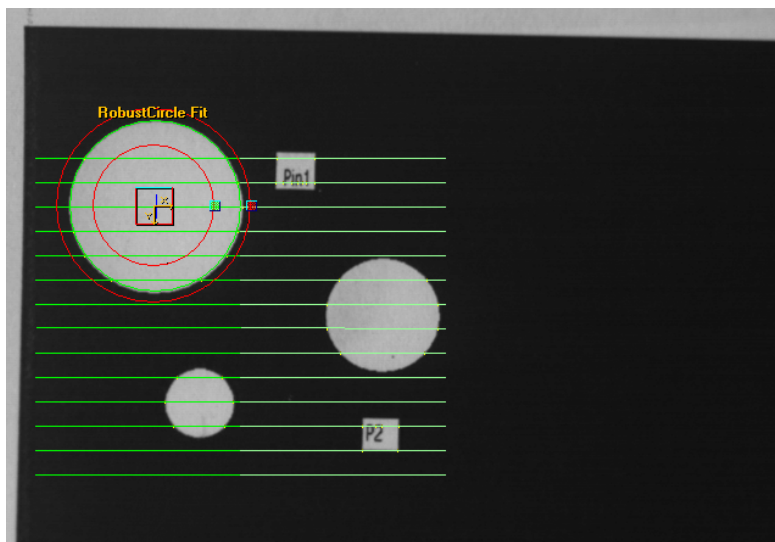
- Graphical set up of the filtering parameters when input datums are not connected
- Graphical feedback showing the current filtering parameters

The shape contains a point that corresponds to the input Nominal Center Point. There is also a rectangle around the point that corresponds to the Location XRange and Location YRange values. The shape has two circles that correspond to the Min Radius value and Max Radius value. If neither radius nor location filtering is enabled, the shape will be hidden.

If no circle is found, or a circle matching the input criteria cannot be computed, or an error occurs, then the `LastError` output datum is set to `ER_STEP_NOT_VALID`. All other output datums are invalid.

Figure 8–8 displays an example of a fitted circle, shown in green, which correctly follows the parts perimeter. The RobustCircleFit is inserted into a Vector Tool with 14 horizontal vectors, and all transitions are in the input point list for RobustCircleFit. The RobustCircleFit has radiusFiltering enabled so all transitions outside of the doughnut shaped area between minRadius and maxRadius (from the RobustCircleFit shape shown in red) are discarded and the largest circle is found.

FIGURE 8–8. Fitted Circle Example



Description

RobustCircle Fit allows editing through the RobustCircle Fit properties page, as shown in Figure 8–9.

FIGURE 8–9. RobustCircle Fit Properties Page

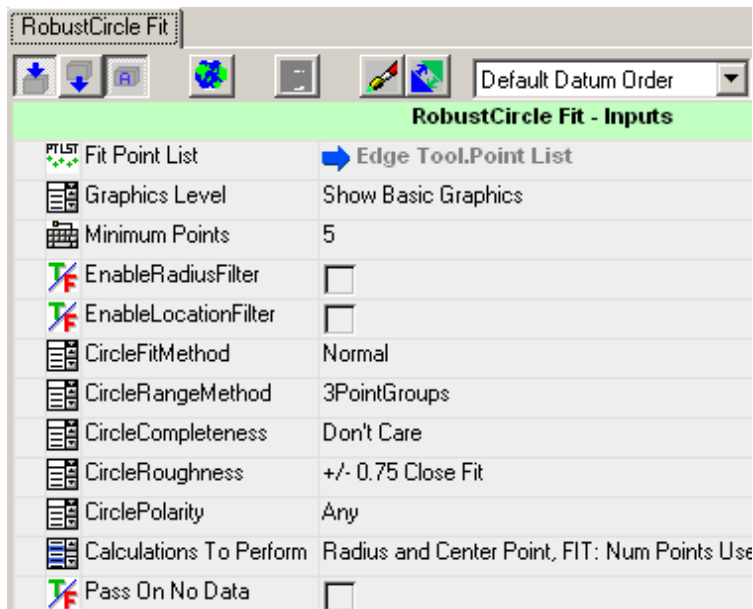


TABLE 8–3. Links to Property Descriptions

For Information About...	Go To...
Calculations To Perform	page 8–22
CircleCompleteness	page 8–21
CircleFitMethod	page 8–21
CirclePolarity	page 8–21
CircleRangeMethod	page 8–21
CircleRoughness	page 8–21
EnableLocationFilter	page 8–20
EnableRadiusFilter	page 8–20
Fit Point List	page 8–20
Graphics Level	page 8–20
Minimum Points	page 8–20
Pass On No Data	page 8–22

Settings

- **Fit Point List** — Allows selection of the RobustCircle Fit input points. Upon insertion, RobustCircle Fit searches its parent step for an output point list and connects its input datum to this output datum. Once located, the input datum is displayed. You can change this to another point list datum if necessary.
- **Graphics Level** — Selects the graphics to be displayed at the completion of step execution:
 - **Show None** — No graphics are drawn.
 - **Show Basic Graphics (default)** — The circle to which the points were fit is displayed in green and a blue cross is displayed at the center point of the circle.
 - **Show Detail** — The graphics associated with Show Basic Graphics are drawn. In addition, a blue cross is drawn at each of the two furthest points from the circle. A yellow line is drawn from each of the two furthest points perpendicular to the circle.
- **Minimum Points** — An integer value that indicates the minimum number of points that must be used in the circle fit.

Default: 3

- **Enable Radius Filter** — When checked, the input points are filtered before the circle fitting is done. The Nominal Center, Minimum Allowable Radius and Maximum Allowable Radius together specify a doughnut shaped area where input points are accepted. Input points that do not fall in this area are discarded. When Enable Radius Filter is not checked, the input points are not filtered.
- **Enable Location Filter** — When checked, the location of the center point of the fitted circle must fall within a specific rectangular region. The center point of the region is the Nominal Center point. The region spans \pm LocationXRange around the center point in X and \pm LocationYRange around the center point in Y (i.e., the width of the region is $2 * \text{LocationXRange}$ and the height is $2 * \text{LocationYRange}$). If no circle is found that matches this criteria, the step will fail. When this input is not checked, the location of the fitted circle is unrestricted.

- **CircleFitMethod** — Specifies which robust circle fitting algorithm to use:
 - Normal — The default location method. Use CircleRangeMethod to specify how the initial circle location is determined.
 - Alternate 1 — Uses an alternate method of initial circle location by discarding the most outlying points first.
- **CircleRangeMethod** — When CircleFitMethod is “Normal,” this datum specifies how the initial circle position is determined:
 - Classic — The initial circle location is taken from an average of all input points. This method requires a uniform distribution of points around the circle. This is the default.
 - 3PointGroups — This method splits the input points into sets of 3 and fits simple circles to each set. The initial robust circle location is then found from a subset of these simple circles which meet the location and radius specifications.
- **CircleCompleteness** — This input has two selections:
 - All Quadrants — Selecting All Quadrants requires that the circle fit use points from all 4 quadrants of the circle.
 - Don’t Care — Selecting Don’t Care means circle fit data can be used from any combination of quadrants.
- **Circle Roughness** — Specifies how closely the points used in the circle fit must fit the output circle. If a circle cannot be found that matches the input points closely enough, the step fails. The numerical values are specified in fractions of a pixel:
 - ± 0.10 Extra Close Fit
 - ± 0.25 Very Close Fit
 - ± 0.75 Close Fit (default)
 - ± 2.75 Rough Fit
 - ± 7.75 Very Rough Fit
- **Circle Polarity** — Specifies the polarity of the circle and the background as:
 - Light On Dark

- Dark On Light
- Any
- **Calculations To Perform** — Select from a list of possible computations for the RobustCircleFit Step. The selections can be individually selected and unselected. The following choices are available in this step:
 - Radius and Center Point
 - Num Points Used in Fit
 - (fitted) Mean, Min, Max, Abs Deviations
 - (fitted) Furthest Points off Circle
 - (candidate) Mean, Min, Max, Abs Deviations
 - (candidate) Furthest Points off Circle
 - Fitted Circle Point List
 - Num points inside circle rejected (not in CircleRoughness range)
 - Num points outside circle rejected (not in CircleRoughness range)
- **Pass On No Data** — Determines whether the RobustCircleFit Step should pass or fail when no circle is found. The default is fail.

Training

The tool is trained on a good image with no filtering active. Training provides a nominal center point.

Results

The result datums returned by RobustCircle Fit include:

- **Status** — Set to true after a successful execution of the step.
- **Fit Point** — The center point of the circle fit.
- **Radius Datum** — Distance Datum indicating the radius of the circle.
- **No of Points used in Fit** — The number of points that were used in the final circle fit.

- Fit Error — Least square fit error.
- RMS Error — Root Mean Square error returned by the Circle Fit algorithm.
- Mean Fit Deviation — Mean deviation of the fitted points from the circle.
- Minimum Fit Deviation — Distance from the circle to the fitted point closest to the circle center.
- Maximum Fit Deviation — Distance from the circle to the fitted point furthest from the circle center.
- Maximum Absolute Value Fit Deviation — A Distance Datum that takes the absolute values of the Minimum and Maximum Fit Deviations, and returns the larger of the two. This value is based only on the final list of points used in the fit, and will not include any filtered points.
- Minimum Fit Radius — Minimum radius found in the fitted point set.
- Maximum Fit Radius — Maximum radius found in the fitted point set.
- Fit point closest to circle center — The fitted point closest to the circle center.
- Fit point furthest from circle center — The fitted point furthest from the circle center.
- No of Candidate Points — The number of candidate points. This is the number of points after filtering is done and before the fitting is done.
- Mean Deviation — Mean deviation of the input1 points from the circle.
- Minimum Deviation — Distance from the circle to the input1 point closest to the circle center.
- Maximum Deviation — Distance from the circle to the input1 point furthest from the circle center.
- Maximum Absolute Value Deviation — A Distance Datum that takes the absolute values of the Minimum and Maximum Deviations, and returns the larger of the two. This value is based on all of the input points.
- Minimum Radius — A Distance Datum indicating the distance from the circle center to the input point that is closest to the center. This includes all input points.

- **Maximum Radius** — A Distance Datum indicating the distance from the circle center to the input point that is furthest from the center. This includes all input points.
- **Point closest to center** — The input1 point closest to the circle center.
- **Point furthest from center** — The input1 point furthest from the circle center.
- **Fitted Circle Point List** — A Point List Datum that includes only the points used in the final circle fit.
- **No of inside points rejected** — The number of points that fall inside the circle and are not within the CircleRoughness range.
- **No of outside points rejected** — The number of points that fall outside the circle and are not within the CircleRoughness range.

Error Codes

RobustCircle Fit returns the following error codes.

- **STEP_ER_NOTVALID**
 - No points in the input point list.
 - No points in the input list that match the minimum and maximum radii requirements.
 - No circle is found, or a circle matching the input criteria cannot be computed.

In these cases, all output datum values are invalid.

- **ROBUSTCIRCLE_FIT_STEP_ER_NOCPUMEM**
 - Runtime memory allocation has failed. In this case, all output datum values are invalid.

I/O Summary

RobustCircle Fit provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

With no filtering active:

Inputs: Point count: ddd / uuu RadFilter: NONE LocFilter: NONE

With (both) filtering active:

Inputs: Point count: ddd / uuu RadFilter: center: (xxx,yyy), radius: rrr to rrr
LocFilter: (xxx, yyy) to (xxx,yyy)

All cases:

Outputs: Circle: X=xxx Y=xxx R=xxx MeanDev=zzz Max+Dev=zzz Max-Dev=zzz

Where: ddd = Number of points in the input point list
uuu = Number of points actually used in the circle fit
xxx, yyy, rrr = Coordinates of X,Y center point or radius R
zzz = deviations in pixels

RobustLine Fit

Using a robust squares fit, RobustLine Fit computes the line that connects a set of points. RobustLine Fit means that points that are not on the line are discarded and do not affect fit accuracy. This differs from least square fit where all data points are used. The input is the set of points, and the outputs include the line and the point that is the geometric center of the line within the ROI. Other outputs show the quality of the fit, the closest and furthest points from the line, and the amount of deviation.

Other Steps Used

RobustLine Fit can only serve as a child step of another step that produces a point list. Two such steps are:

- **Edge Tool** — The Edge Tool finds straight, circular, or elliptical edges or line segments in the ROI. The edges are output in a point list.
- **Vector Tool** — The Vector Tool finds transition edge points along predefined directions.

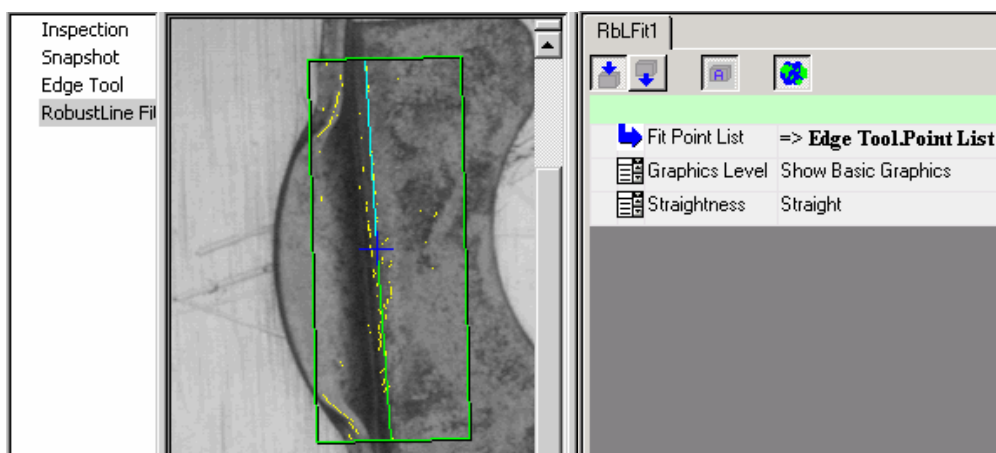
Theory of Operation

Typically, RobustLine Fit fits straight lines to edge points extracted from an image. Normally, it is placed inside the Edge Tool, but can be inserted in tools that generate point lists like Vector or a custom Perl vision tool. Edge points are extracted by the Edge Tool and are contained in a point list datum. RobustLine Fit finds the equation of the best line through the points using a robust algorithm. It can find lines in edge data even when there is significant noise. It will attempt

to find the longest line segment in the edge data by pairing points into line segments and finding the most popular line segment in terms of slope and intercept. The data points that fall on the most popular line are then combined and subject to a final least square fit to produce the result.

A RobustLine Fit is shown in Figure 8–10. The edge, shown in green, precisely fits to the edge of the strip. Edge points not on this edge do not affect the fit as they are discarded.

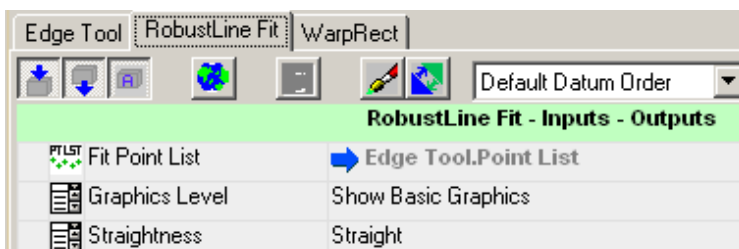
FIGURE 8–10. RobustLine Fit — Trained



Description

RobustLine Fit allows editing through the RobustLine Fit properties page, as shown in Figure 8–11.

FIGURE 8–11. RobustLine Fit Properties Page



Settings

- **Fit PointList** — Allows selection of the RobustLine Fit edge points. The edge points are contained in a point list datum.
- **Graphics Level** — Selects the amount of graphics to be displayed at the completion of step execution:
 - **Show Basic Graphics (Default)** — The line to which the points were fit is displayed in dark blue to light blue, and a blue cross is displayed at the center point of the line. Direction of the line goes from dark to light.
 - **Show Detail** — The graphics associated with Show Basic Graphics are drawn. In addition, a blue cross at the two farthest points of the line and a yellow line perpendicular to the fit line (dark blue -> light blue) at the two farthest points are drawn.
 - **Show None** — No graphics are drawn.
- **Straightness** — Selects the roughness of the detected edge:
 - **Very Straight**
 - **Straight (default)**
 - **Rough**
 - **Very Rough**

The fit algorithm requires some indication of the roughness of the edge data. This determines how close the points must be to the best line so that they are included in the final fit. It allows the fit to include edge points that lie close to the line but would be discarded by a stringent fit.

Training

None.

Results

The result datums returned by this step include:

- **Status (Status)** — Status datum showing whether the step passed or failed.

- Fit Point (Pt) — Point datum containing the fitted line center point. This center point is located at the midpoint between the two points of intersection with the ROI.
- Line Datum (Line) — Line datum that contains the fitted line equation.
- Line Angle Datum (Angle) — Angle datum that contains the angle of the fitted line.
- No of points used in the fit (UsesPts) — Int datum indicating how many points were used in the line fit.
- Fit Error (FitErr) — Distance datum indicating the fit error, defined as the square root of the sum of all the deviations.
- Mean Deviation (MeanDev) — Distance datum indicating the mean distance for all points to the line.
- Minimum Deviation (MinDev) — Distance datum indicating the distance from the closest point to the line.
- Maximum Deviation (MaxDev) — Distance datum indicating the distance from the furthest point to the line.
- Furthest point after line (MaxPt) — Point datum indicating the furthest point above the line.
- Furthest point before line (MinPt) — Point datum indicating the furthest point below the line.

I/O Summary

RobustLine Fit provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Point count: ddd Straightness: aaa

Outputs: Line: A=xxx B=xxx C=xxx N=xxx Point: X=xxx Y=xxx

Where: aaa = Very Straight or Straight, or VeryRough or Rough

ddd = Number of points

xxx = X,Y coordinates of the center point of the line, or line

equation coefficients: A, B, or C, or the number of points

N used in the fit

Measurement Tools

Dist2Pts Meas

This tool measures the distance between two points with respect to their common system of coordinates. The inputs are two points, and the outputs include the distance between the two points, the midpoint between the two points, and the line and angle from point 1 to point 2.

Other Steps Used

None.

Theory of Operation

Dist2Pts Meas requires:

- 2 input point datums
- 3 output distance datums
- 1 output line datum
- 1 output angle datum
- 1 output point datum

Dist2Pts Meas transforms the two input points to a common system of coordinates and computes the distance between them. Because of this, Dist2Pts Meas can measure the distance between any two points even if they are in different buffers.

The distance result is calculated and reported in the World coordinate system when the two input points come from two different camera Snapshot images. The distance is meaningful in that case only if the System has been calibrated.

The distance result is calculated and reported in the Camera coordinate system when the two input points come from the same camera image or when one of the points comes from the camera image and the other comes from an intermediate images of that camera image (for example, from a Warped or Gray Morphology image).

If the two input points are in the same coordinate system, then the distance result will be calculated and reported in that coordinate system. Note that this is not necessarily the coordinate system of a camera image, as is the case when the two input points comes from the same Warp or Gray Morphology image.

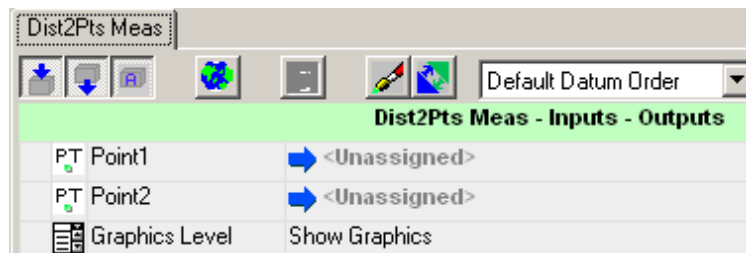
Note: If the option to report all results in calibrated units is turned on in the Report form of FrontRunner™, then all distance results will be reported as follows:

- If the System **has been calibrated**, results are shown in the World coordinate system.
- If the System **has not been calibrated**, results are shown in the Camera coordinate system.

Description

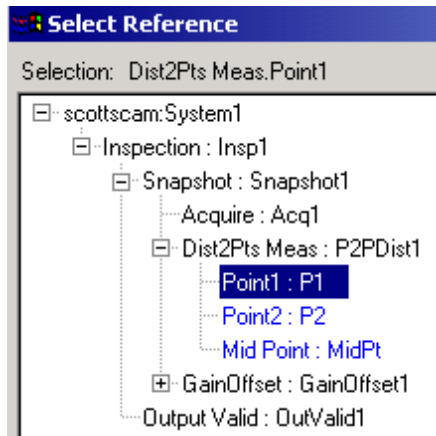
Dist2Pts Meas allows editing through the Dist2Pts Meas properties page, as shown in Figure 8–12.

FIGURE 8–12. Distance2Pts Meas Properties Page



Settings

- **Point1** — Allows the selection of a point datum from the Job. This datum acts as the first end-point of the distance measurement. Click the drop-down list button located to the right of the **Point1** drop-down list to set up a measurement of the distance between any two points. This displays the Select Reference dialog box, as shown in Figure 8–13. In the tree, any point datums in the Job available as inputs will be highlighted for selection.

FIGURE 8–13. Dist2Pts Meas — Select Reference Dialog Box

- **Point2** — Allows the selection of a second point datum from the Job. This datum acts as the second end-point of the distance measurement.
- **Graphics Level** — Allows the selection of graphics to be displayed when executed:
 - **Show Graphics (Default)** — Displays a line segment between the start point (Point1) and the end point (Point2). Line direction is also shown as a two tone line in the image (from dark to light blue).
 - **Show None** — Disables graphics display.

Training

None.

Results

The result is a distance datum that contains the distance measurement value (in pixels) in the coordinate system common to both points.

- **Status (Status)** — Status datum showing whether the step passed or failed.
- **Pt-Pt Distance (Dist)** — Straight line distance between the two points.
- **Horizontal Distance (HDist)** — Horizontal distance between the two points.
- **Vertical Distance (VDist)** — Vertical distance between the two points.

- Pt-Pt Line (Line) — LineDm, which is the line between the two points. The line direction is from Point1 to Point2.
- Line Angle (Angle) — AngleDm, which is the angle from point 1 to point 2.
- Mid Point (MidPt) — PointDm, which is the midpoint between the two points.

I/O Summary

Dist2Pts Meas provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Point1: X=xxx Y=xxx Point2:X=xxx Y=xxx

Outputs: Dist: www, Line: A=yyy B=yyy C=yyy (Ang=yyy Deg),
Midpoint: X=zzz Y=zzz

Where: www = Distance between the two points
xxx = X, Y coordinates of the two points of interest
yyy = Coefficients and angle of the line between the two points
zzz = Coordinates of the midpoint between the two points

Error Message:

- Dist2Pts Meas failed

IntersectLines Meas

This tool computes the intersection point of two lines. The inputs are the two line datums to intersect. The output includes a point datum holding the intersection point and an angle datum holding the included angle from input line1 to input line2 as defined by the direction vectors of the individual lines.

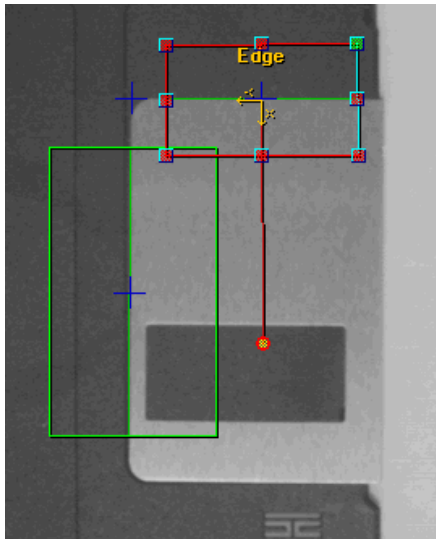
Other Steps Used

None.

Theory of Operation

IntersectLines Meas computes the intersection point of two lines. Each line is defined by an input line datum, which is output from another step such as a LineFit Step. The intersection point is output as a point datum. The example in Figure 8–14 demonstrates the use of the IntersectLines Meas Step to compute the intersection point of two edges. The line inputs should be connected to output line datums of other steps.

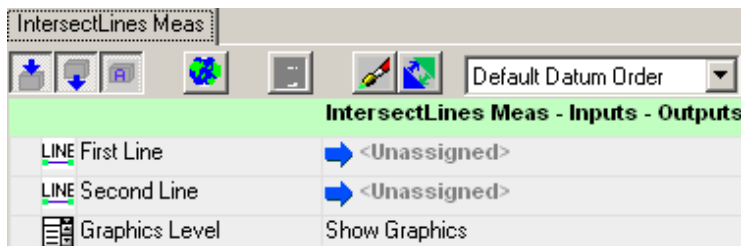
FIGURE 8–14. IntersectLines Meas — Example



Description

IntersectLines Meas allows editing through the IntersectLines Meas properties page, as shown in Figure 8–15.

FIGURE 8–15. IntersectLines Meas Properties Page



Settings

- First Line — Defines the first line datum.
- Second Line — Defines the second line datum.
- Graphics Level — Allows selection of graphics to be displayed when executed:
 - Show Graphics (default) — Enables graphics display.
 - Show None — Disables graphics display.

Training

None.

Results

- Status (Status) — Status datum showing whether the step passed or failed.
- Point (Pt) — Intersection point returned by this step in a PointDm.
- Angle between the Lines (Angle) — Angle, returned in an AngleDm, that is the included angle from the first input line to the second.

I/O Summary

IntersectLines Meas provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Line1: A=xxx B=xxx C=xxx Line2: A=xxx B=xxx C=xxx

Outputs: Intersection: X=yyy Y=yyy (acute Ang=yyy Deg)

Where: xxx = Coefficients of the input lines
 yyy = Intersection point and angle between the two lines

Error Message:

- LineIntersection Meas step failed

BisectLines Meas

This tool calculates the equations of the two lines that bisect two input lines. The inputs to the step are two line datums. The outputs include two line datums holding the bisector lines, the point of intersection, and the angle between the two input lines.

Other Steps Used

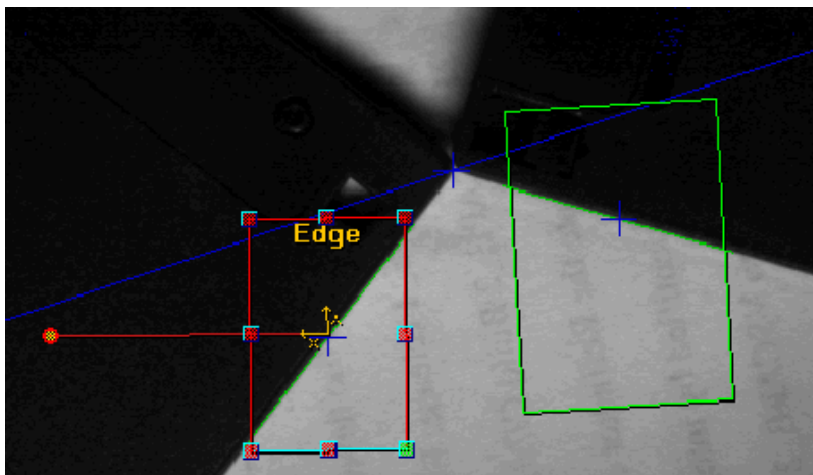
None.

Theory of Operation

BisectLines Meas computes the line equations that divide the two angles formed by two input lines. Each input line datum is an output from another step, such as the RobustLine Fit.

BisectLines Meas computes the two lines that bisect the two input lines, as shown in Figure 8–16.

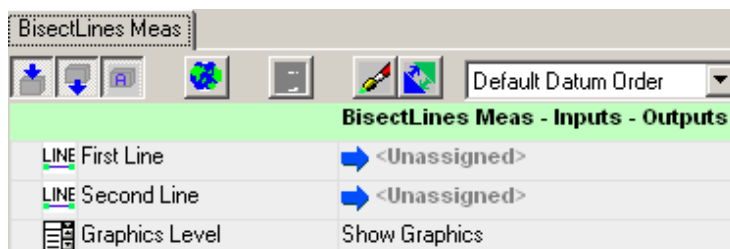
FIGURE 8–16. BisectLine Meas from 2 Edge Tools



Description

BisectLines Meas allows editing through the BisectLines Meas properties page, as shown in Figure 8–17.

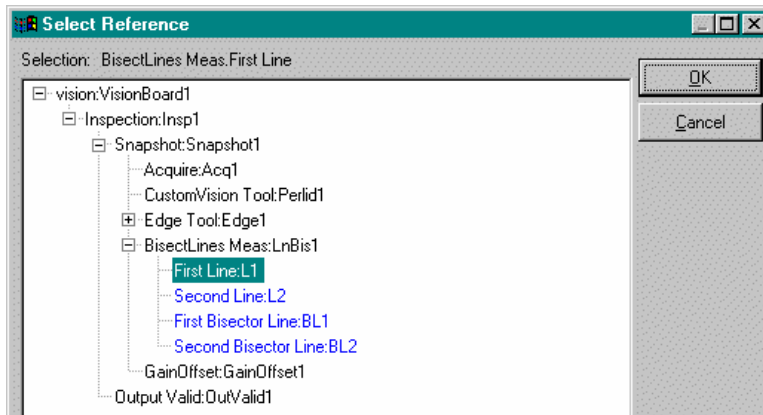
FIGURE 8–17. BisectLines Meas Properties Page



Settings

- **First Line** — Allows the selection of the first line datum from the Job. This datum acts as the first of the two lines to bisect.
- **Second Line** — Allows selection of second line datum from the Job. This Job acts as the second of the two lines to bisect.
- **Graphics Level** — Allows the selection of graphics to be displayed when executed:
 - Show Graphics (default) — Enables graphics display.
 - Show None — Disables graphics display

Click the drop-down list button located to the right of the First Line drop-down list to bisect any two lines. This displays the Select Reference dialog box, as shown in Figure 8–18.

FIGURE 8–18. BisectLines Meas — Select Reference Dialog Box

Select **First Line:L1** from the Job Tree. In the tree, any line datums in the Job that are available as inputs will be highlighted for selection.

Training

None.

Results

- **Status (Status)** — Status datum showing whether the step passed or failed.
- **First Bisector Line (BL1)** — First bisector line returned by this step in line datums.
- **Second Bisector Line (BL2)** — Second bisector line returned by this step in line datums.
- **Point (Pt)** — Intersection point between two lines returned in a point datum.
- **Angle between input Lines (Angle)** — Included angle between the two lines returned in an angle datum.

Note: If the two input lines are parallel, the First Bisector Line (BL1) will be a mid-line between the two inputs, the Second Bisector Line (BL2) and Point outputs will be invalid.

I/O Summary

BisectLines Meas provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Line1: A=xxx B=xxx C=xxx (Ang=xxx)

Line2: A=xxx B=xxx C=xxx (Ang=xxx)

Outputs: Bisector Line1: A=yyy B=yyy C=yyy (Ang=yyy)

Bisector Line2: A=yyy B=yyy C=yyy (Ang=yyy)

Where: xxx = Coefficients and the angles of the input lines that will be bisected
yyy = Coefficients and the angles of the 2 calculated bisecting lines

Error Message:

- BisectLines Meas step failed

PtLineNormal_Meas

This tool computes measurement statistics, given an input point and an input line.

Note: This tool is identical to “Pt to Line Distance” starting on page 8-41. We continue to document it for backward compatibility.

Other Steps Used

PtLineNormal_Meas can be accessed only through the Custom Step or Custom Vision Tool that is configured to do point-to-line measurements.

Theory of Operation

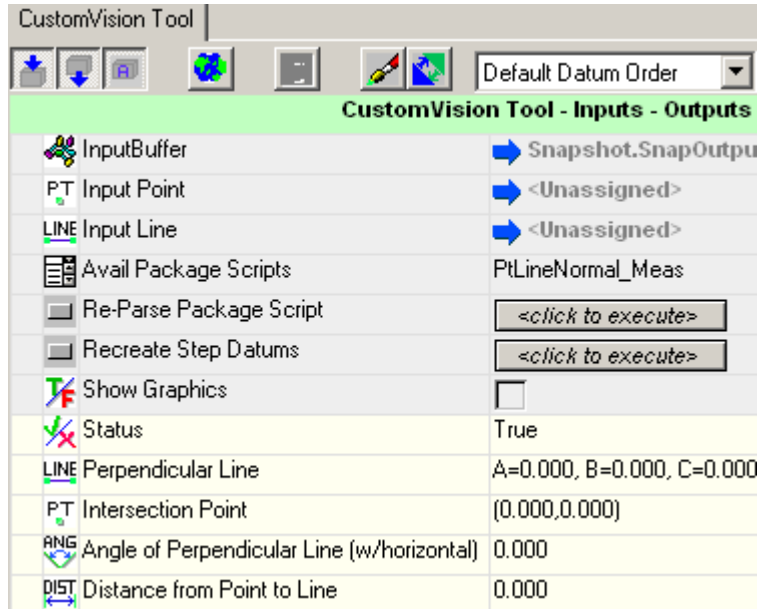
The input point and line are transformed to the same coordinate system so that measurements can be made. The following are computed:

- Equation of the line perpendicular to the input line, which contains the input point
- Intersection point between the input line and this perpendicular line
- Angle between the perpendicular line and the horizontal line
- Distance between the input point and input line

Description

PtLineNormal_Meas allows editing through the CustomVision Tool properties page with PtLineNormal_Meas selected, as shown in Figure 8–17.

FIGURE 8–19. Custom Step/CustomVision Tool — PtLineNormal_Meas



Settings

The CustomVision Tool and/or Custom Step properties page, with PtLineNormal_Meas selected, allows the modification of the following properties:

- **Input Point** — Select the input point from within the Job for this step. This point must lie outside of the line selected for Input Line.
- **Input Line** — Select the input line from within the Job for this step.
- **Avail Package Scripts** — Select the appropriate Perl script package from the drop-down list. The script package that implements the functionality of this step is PtLineNormal_Meas.

- **Re-Parse Package Script** — This button causes the package script to be parsed when clicked. Whenever any changes to the script are made, this button needs to be clicked to make these changes take effect.
- **Recreate Step Datums** — This button causes the input and output datum lists in the step to be recreated. This button only needs to be clicked when a datum is either added, removed or changed in the script. If the script is changed, but no input or output datums are changed, then this button does not need to be clicked. Clicking this button also causes all input datums to be set to their default values and lose their connections to other step results or parameters.

Datums created by the package script are added to the user interface. Input datums are shown as a box with a drop-down list button. The input datums can be linked to other similar type datums in the Job. Clicking the drop-down list button causes the Job Tree to be displayed, allowing you to select the datum to link to the input datum. Resource datums are shown as user-editable boxes that can be set to a value directly. Output datums are not shown in the user interface for this step, but can be seen in the Job Tree that comes up when linking an input datum.

If you use a Custom Vision Tool for this step, then the properties page will have a **Show Graphics** checkbox. If you check (enable) this checkbox, the following graphics are output:

- A crosshair is drawn at the input point
- A crosshair is drawn at the intersection point of the input line and the computed perpendicular line
- The perpendicular line between the input point and the input line is drawn

Training

None.

Results

The following datums are output by this step. Measurement statistics are computed in this step.

- **Status (Status)** — Status datum showing whether the step passed or failed.
- **Perpendicular Line (PerpendicularLine)** — The equation of the line perpendicular to the input line that contains the input point.

- Intersection Point (IntersectionPoint) — The intersection point between the input line and the computed perpendicular line.
- Angle of Perpendicular Line (AngleofPerpendicular) — The angle the computed perpendicular line makes with the horizontal.
- Distance from Point to Line (PointtoLineDistance) — The distance between the input point and the input line.

I/O Summary

PtLineNormal_Meas provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Always Trained

Outputs: Perl Step is run by a PERL script. Script Selected:
PtLineNormal_Meas Status: xx

Where: xxx = OK

Error Message:

- Not Available — Script not found

Pt to Line Distance

This tool computes measurement statistics given an input point and an input line.

Other Steps Used

None.

Theory of Operation

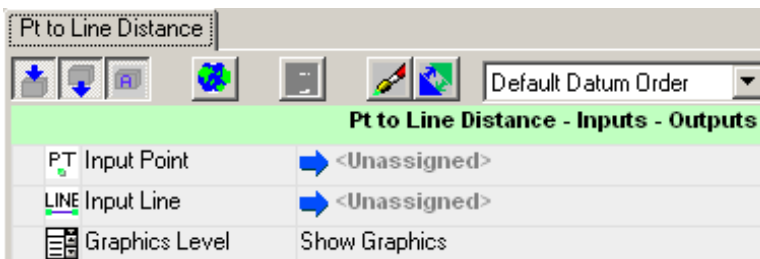
The input point and line are transformed to the same coordinate system so that measurements can be made. The following are computed:

- Equation of the line perpendicular to the input line, which contains the input point
- Intersection point between the input line and this perpendicular line
- Angle between the perpendicular line and the horizontal line
- Distance between the input point and input line

Description

Pt to Line Distance allows editing through the Pt to Line Distance properties page, as shown in Figure 8–20.

FIGURE 8–20. Pt to Line Distance Properties Page



Settings

- **Input Point** — Select the input point from within the Job for this step. This point must lie outside of the line selected for Input Line.
- **Input Line** — Select the input line from within the Job for this step.
- **Graphics Level** — Is either Show None or Show Graphics:
 - If you select Show Graphics, the following graphics are output:
 - A crosshair is drawn at the input point
 - A crosshair is drawn at the intersection point of the input line and the computed perpendicular line
 - The perpendicular line between the input point and the input line is drawn

Training

None.

Results

The following datums are output by this step. Measurement statistics are computed in this step.

- **Status (Status)** — Status datum showing whether the step passed or failed.

- Pt to Line Distance — Distance Datum indicating the perpendicular distance from the input point to the input line.
- Normal Line — Line Datum indicating the line that runs perpendicular to the input line, and runs through the input point.
- Angle of Normal Line — Angle Datum indicating the angle of the line that runs perpendicular to the input line.
- Normal Point — Point Datum indicating the point where the Normal Line intersects the input line.

I/O Summary

Pt to Line Distance provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: InPoint:X=xxx Y=yyy InLine:A=aaa B=bbb C=ccc

Where: xxx, yyy are the input point coordinates
aaa, bbb, and ccc are the coefficients of the input line.

Outputs: Dist: ddd, NormPt: X=xxx Y=yyy, NormLine: A=aaa B=bbb
C=ccc (Ang=ggg Deg)

Where: ddd is the distance from the input point to the input line
xxx, yyy are the coordinates of the normal point
aaa, bbb, and ccc are the coefficients of the normal line
ggg is the normal angle

Error Message:

- Pt to Line Distance measurement step failed.

Measurements Tolerancing

Tolerance Meas

This tool compares the measured input to a given nominal value or a given upper and lower range and decides whether the measured input is within the specified tolerance. Tolerance Meas can perform comparisons on the following types of datums. The type of datum that is selected can be identified by the I/O Summary.

- IntDm — An integer value
- DoubleDm — A floating-point value
- AreaDm — An area value
- DistanceDm — A distance value
- AngleDm — An angle value
- LineDm — A line value

Other Steps Used

None.

Theory of Operation

Tolerance Meas can operate in one of two modes.

- If the step is set up to be the **Nominal** \pm type, it checks whether the difference between a nominal value and the measured value is within a tolerance.
- If the step is set up to be the **Upper/Lower** type, it checks whether the measured value is within a specified range. The nominal value and the tolerance level can be specified manually or can be trained. If the upper/lower range is used instead of the nominal value, the range can only be specified manually.

The output of the step is a StatisticsDm containing the statistics of the measured value. When the step is set up to compare the measured value to a nominal value, the result of pass or fail indicates whether the difference between the measured and the nominal is within the specified tolerance. When set up to compare the measured value to an upper and lower range, the step will pass if the measured value is within this range.

Tolerance Meas can only be inserted into another step that has a datum output that is one of the acceptable datum types.

The nominal and upper/lower range values can be trained to be external values or can be manually specified before the run. They can also be updated during runtime with the values of the external inputs. Tolerance Meas status output can also be displayed through a digital output channel. The nominal, tolerance, and upper/lower range values can be displayed either in buffer screen pixel space or in calibrated world space.

Tolerance Meas collects statistical information about the measured values. The statistical information includes mean, maximum and minimum values, and standard deviation.

Description

Tolerance Meas allows editing through the Tolerance Meas properties page.

- When **Tolerance Type** is set to **Nominal \pm** , the Tolerance Meas (Nominal \pm) properties page will be as shown in Figure 8–21.
- When **Tolerance Type** is **Upper/Lower**, the Tolerance Meas (Upper/Lower) properties page appears, as shown in Figure 8–22.

FIGURE 8-21. Tolerance Meas Properties Page — Nominal \pm

The screenshot shows the 'Tolerance Meas' properties dialog. The title bar is 'Tolerance Meas'. Below the title bar is a toolbar with icons for adding, removing, and applying measures, a globe icon, a refresh icon, a pencil icon, and a 'Default Datum Order' dropdown menu. The main area is titled 'Tolerance Meas - Inputs - Outputs' and contains a table of properties.

Tolerance Meas - Inputs - Outputs	
Input Datum	Snapshot.Acquire Time Stamp : {
Nominal Input	<Unassigned>
Tolerance Type	Nominal +/-
Output To	<Unassigned>
Update Statistics	Never
Reset Statistics	<click to execute>
Use Calibration	<input type="checkbox"/>
Update Nominal at Runtime	<input type="checkbox"/>
Update Nominal	<click to execute>
With Tolerance	5% Nominal
Nominal Value	0.000
Tolerance +/-	0.000

FIGURE 8-22. Tolerance Meas Properties Page — Upper/Lower

The screenshot shows the 'Tolerance Meas' properties dialog for the Upper/Lower configuration. The title bar is 'Tolerance Meas'. Below the title bar is a toolbar with icons for adding, removing, and applying measures, a globe icon, a refresh icon, a pencil icon, and a 'Default Datum Order' dropdown menu. The main area is titled 'Tolerance Meas - Inputs - Outputs' and contains a table of properties.

Tolerance Meas - Inputs - Outputs	
Input Datum	Snapshot.Acquire Time Stamp : { 0.000 }
Lower Input	<Unassigned>
Upper Input	<Unassigned>
Tolerance Type	Upper/Lower
Output To	<Unassigned>
Update Statistics	Never
Reset Statistics	<click to execute>
Use Calibration	<input type="checkbox"/>
Lower Value	0.000
Upper Value	0.000

Settings

TABLE 8–4. Tolerance Meas Properties

Property	Description
Input Datum (Nominal \pm) (Upper/Lower)	<p>Selects a datum from the current Job whose tolerance is to be checked. When you click Reference List, the Select Reference window is displayed showing a tree of datums.</p> <p>Selecting a datum from this tree sets up the tolerance measurement for that datum. When you train this step or click Update Nominal, the Nominal Value datum is set to match the value of this datum.</p>
Lower [value] (Upper/Lower)	<p>Holds the lower range value that is used in the comparison process.</p> <p>Default: 0.0</p>
Lower Input (Upper/Lower)	Must match the datum type selected in Input Datum. When you train this step, this external input datum value is set to be the effective lower range value shown in the corresponding input datum.
Nominal Input (Nominal \pm)	Must match the datum type selected in Input Datum. When you <i>train</i> this step or click Update Nominal, this external input datum value is set to be the effective nominal value shown in the corresponding input datum type.
Nominal Line Angle (Deg) (Nominal \pm)	The value that was trained or set to be compared to for tolerance testing. The default is 0.0, and it is set to the value of the input upon training or update.
Output To (Nominal \pm) (Upper/Lower)	Allows the selection of the digital output channel for the tolerance status output. When the measured input is within the tolerance, this digital output is set to 1. Otherwise, it is set to 0.
Reset Statistics (Nominal \pm) (Upper/Lower)	Clears the existing statistics and sets them to 0.
Tolerance \pm (Nominal \pm)	Holds the current tolerance value corresponding to the setting of With Tolerance.

TABLE 8-4. Tolerance Meas Properties (continued)

Property	Description
Tolerance Type (Nominal \pm) (Upper/Lower)	<p>Sets up the step to compare the measured value to a nominal with a tolerance or to compare it to a range.</p> <p>When you select Nominal \pm, the difference between the measured value and the input nominal value is compared to a tolerance. This is the default.</p> <p>When you select Upper/Lower, the measured value is compared to the range specified by the upper and lower value inputs.</p>
Update Nominal at Runtime (Nominal \pm)	<p>When enabled (checked), the value contained in the Nominal Input is written to the Nominal Value property before each run and used as the nominal for tolerancing the input.</p> <p>Default: Disabled</p>
Update Nominal (Nominal \pm)	<p>The value of the datum in Nominal Input is written to the Nominal Value property immediately after you click this button.</p>
Update Statistics (Nominal \pm) (Upper/Lower)	<p>The statistics collected are mean, maximum and minimum values, the standard deviation, and the number of data collected. The following allow you to modify the times at which statistics are updated.</p> <p><i>Always</i> — Statistical information is collected for each run or trial run of the step.</p> <p><i>Only On Pass</i> — Statistical information is collected only when the part passes.</p> <p><i>Only On Fail</i> — Statistical information is collected only when the part fails.</p> <p><i>Never (Default)</i> — Never update the statistics.</p>

TABLE 8–4. Tolerance Meas Properties (continued)

Property	Description
Upper [value] (Upper/Lower)	Holds the upper range value that is used in the comparison process. Default: 0.0
Upper Input (Upper/Lower)	Must match the datum type selected in Input Datum. When you train this step, this external input datum value is set to be the effective upper range value shown in the corresponding input datum.
Use Calibration (Nominal \pm) (Upper/Lower)	Toggles between the world and the pixel spaces in which the nominal coordinates, tolerances and statistics information are expressed. When enabled, the information is presented in world coordinates. Default: Disabled
With Tolerance (Nominal \pm)	Allows selection of a tolerance level. 1% — Of the corresponding nominal value. 5% (default) — Of the corresponding nominal value. 10% — Of the corresponding nominal value. 3 <i>Std Deviation</i> — The tolerance is set to be three times the current standard deviation. As Is — The current tolerance is never changed from the value displayed in Tolerance \pm .

Training

Tolerance Meas does not have to be trained in order to run it. When trained, if the step is set up as the Nominal \pm type, the value in **Nominal Input** is written to the effective nominal **Nominal Value** property. The tolerance value is adjusted according to the setting in **With Tolerance**.

Results

- **Status** — Indicates whether the measured value is within the tolerance of the nominal value or within the specified upper/lower range.
- **Number of Passes** — The number of passes.
- **Number of Fails** — The number of failures.

- **Statistics** — The statistical information about the measured value: mean, maximum, minimum, and standard deviation of the measured values, and the number of data collected.

I/O Summary

Tolerance Meas provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

- When Input Datum is an **AreaDm** and Tolerance Type is **Nominal ±**:

Inputs: Nominal Area = xxx, Tolerance = xxx, yyy Update
Statistics. zzz Update Nominal on Runtime

- When Input Datum is an **AreaDm** and Tolerance Type is **Upper/Lower**:

Inputs: Upper Area = xxx, Lower Area = xxx, yyy Update
Statistics.

Outputs: Measured: mmm, Statistics: Min=sss Avg=sss Max=sss
StDev=sss, Count=sss

Where: mmm = Input Value

sss = The statistics data taken from the tool executions

xxx = Nominal Value and Tolerance

yyy = Always, Never, Only On Pass, or Only On Fail

zzz = Do NOT or a blank space

- When Input Datum is a **DistanceDm** and Tolerance Type is **Nominal ±**:

Inputs: Nominal Distance = xxx, Tolerance = xxx, yyy Update
Statistics. zzz Update Nominal on Runtime

- When Input Datum is a **DistanceDm** and Tolerance Type is **Upper/Lower**:

Inputs: Upper Distance = xxx, Lower Distance = xxx, yyy Update
Statistics.

Outputs: Measured: mmm, Statistics: Min=sss Avg=sss Max=sss
StDev=sss, Count=sss

Where: mmm = Input Value
 sss = The statistics data taken from the tool executions
 xxx = Nominal Value and Tolerance
 yyy = Always, Never, Only On Pass, or Only On Fail
 zzz = Do NOT or a blank space

- When Input Datum is a **DoubleDm** and Tolerance Type is **Nominal ±**:
 Inputs: Nominal Double = xxx, Tolerance = xxx, yyy Update Statistics. zzz Update Nominal on Runtime
- When Input Datum is a **DoubleDm** and Tolerance Type is **Upper/Lower**:

Inputs: Upper Double = xxx, Lower Double = xxx, yyy Update Statistics.

Outputs: Measured: mmm, Statistics: Min=sss Avg=sss Max=sss StDev=sss, Count=sss

Where: mmm = Input Value
 sss = The statistics data taken from the tool executions
 xxx = Nominal Value and Tolerance
 yyy = Always, Never, Only On Pass, or Only On Fail
 zzz = Do NOT or a blank space

- When Input Datum is an **IntDm** and Tolerance Type is **Nominal ±**:
 Inputs: Nominal Integer = xxx, Tolerance = xxx, yyy Update Statistics. zzz Update Nominal on Runtime
- When Input Datum is an **IntDm** and Tolerance Type is **Upper/Lower**:

Inputs: Lower Integer = xxx, Upper Integer = xxx, yyy Update Statistics.

Outputs: Measured: mmm, Statistics: Min=sss Avg=sss Max=sss StDev=sss, Count=sss

Where: mmm = Input Value
 sss = The statistics data taken from the tool executions
 xxx = Nominal Value and Tolerance or Upper and Lower range value
 yyy = Always, Never, Only On Pass, or Only On Fail
 zzz = Do NOT or a blank space

- When Input Datum is a **LineDm** and Tolerance Type is **Nominal ±**:

Inputs: Nominal Line Angle (Deg)= xxx, Tolerance = xxx, yyz
Update Statistics. zzz Update Nominal on Runtime

- When Input Datum is a **LineDm** and Tolerance Type is **Upper/Lower**:

Inputs: Lower Line Angle (Deg) = xxx, Upper Line Angle (Deg) =
xxx, yyz Update Statistics.

Outputs: Measured: mmm, Statistics: Min=sss Avg=sss Max=sss
StDev=sss, Count=sss

Where: mmm = Input Value

sss = The statistics data taken from the tool executions

xxx = Nominal Value and Tolerance

yyz = Always, Never, Only On Pass, or Only On Fail

zzz = Do NOT or a blank space

Error Message:

- Tolerance Meas measurement failed

PointTolerance Meas

This tool verifies whether the distance between the input point and the nominal point is within a user-specified tolerance. It is also possible to specify an upper and lower range of point values between which the measured point must fall. Statistical data on the measured distances are collected.

The inputs are the nominal point, the distance tolerance, and an input point datum holding the measured point. When the step is set up to use an upper and lower range, the inputs are the upper and lower points and an input point datum holding the measured point. The nominal point and the tolerance can either be trained or manually specified. The output of the step is a statistics datum containing statistics for the distances between the measured and the nominal points. The result of pass or fail indicates whether the measured point falls within the tolerance of the nominal or falls within the upper and lower range.

Other Steps Used

None.

Theory of Operation

PointTolerance Meas can be inserted into another step, as long as at least one of the parent steps has a point datum as output. It can operate in one of two modes:

- If it is set up to be the **Nominal \pm** type, it compares the measured point to a nominal plus a tolerance. By default, the tolerances are set to be 5% of the corresponding nominal coordinates. Whenever the nominal is updated, the tolerances are updated accordingly. You also have the option to manually set the tolerances so that they are not automatically updated with the nominal coordinates.

At runtime, the differences between the measured and the nominal point along x- and y-directions are compared to their corresponding tolerances. When both differences fall within the x and y tolerances, the step passes, otherwise the step fails.

- If it is set up to be the **Upper/Lower** type, the ranges are specified as x- and y- coordinates. At runtime, the measured point along the x- and y-directions is compared to the x-and y-ranges. The step fails when the measured point is outside the range.

PointTolerance Meas status output can also be indicated through a digital or virtual output change.

The nominal points, tolerances, and upper and lower ranges can be displayed either in buffer pixel coordinates or in calibrated world space.

PointTolerance Meas collects statistical information on the measured point. This statistical information includes: mean, maximum, minimum, standard deviation, and number of data.

Description

PointTolerance Meas allows editing through the PointTolerance Meas properties page.

- When **Tolerance Type** is **Nominal \pm** , the PointTolerance Meas (Nominal \pm) properties page appears, as shown in Figure 8–23.
- When **Tolerance Type** is **Upper/Lower**, the PointTolerance Meas (Upper/Lower) properties page appears, as shown in Figure 8–24.

FIGURE 8-23. PointTolerance Meas Properties Page (Nominal \pm Type)

PointTolerance Meas

PointTolerance Meas - Inputs - Outputs

Input Datum	TwoPt Locator.CenterPt : { (0.000
Nominal Input	<Unassigned>
Tolerance Type	Nominal +/-
Output To	<Unassigned>
Update Statistics	Never
Reset Statistics	<click to execute>
Use Calibration	<input type="checkbox"/>
Update Nominal at Runtime	<input type="checkbox"/>
Update Nominal	<click to execute>
With Tolerance	5% Nominal
Nominal X	0.000
Nominal Y	0.000
Tolerance X +/-	0.000
Tolerance Y +/-	0.000

FIGURE 8-24. PointTolerance Meas Properties Page (Upper/Lower Type)

PointTolerance Meas

PointTolerance Meas - Inputs - Outputs

Input Datum	TwoPt Locator.CenterPt : { (0.000,0.000) }
Tolerance Type	Upper/Lower
Output To	<Unassigned>
Update Statistics	Never
Reset Statistics	<click to execute>
Use Calibration	<input type="checkbox"/>
Lower X	0.000
Lower Y	0.000
Upper X	0.000
Upper Y	0.000

Settings

TABLE 8–5. PointTolerance Meas Properties

Property	Description
Input Datum (Nominal \pm) (Upper/Lower)	The point datum that is compared to the nominal at runtime. By default, this input is a point datum of the most immediate parent step that has a point datum output.
Lower X (Upper/Lower)	Holds the x-coordinate of the lower range value that is used in the comparison process, expressed in either pixel space or in world space. Default: 0.0
Lower Y (Upper/Lower)	Holds the y-coordinate of the lower range value that is used in the comparison process, expressed in either pixel space or in world space. Default: 0.0
Nominal Input (Nominal \pm)	The point datum that serves as the nominal point. When you train PointTolerance Meas or click Update Nominal, the settings for Nominal X and Nominal Y are set to match the coordinates of the input point.
NominalX (Nominal \pm)	The x-coordinate of the nominal point, expressed either in pixel space or in world space.
NominalY (Nominal \pm)	The y-coordinate of the nominal point, expressed either in pixel space or in world space.
Output To (Nominal \pm) (Upper/Lower)	Selects the digital or virtual output channel for the tolerance status output. When the measured input is within the tolerance, this output is set to 1. When the measured input is not within the tolerance, this output is set to 0.
Reset Statistics (Nominal \pm) (Upper/Lower)	Clears the current statistics.
Tolerance Type (Nominal \pm) (Upper/Lower)	Sets up the step to compare the measured point to a nominal with a tolerance or to compare it to a range. When you select Nominal \pm , the difference between the measured value and the input nominal value is compared to a tolerance. This is the default. When you select Upper/Lower , the measured value is compared to the range specified by the upper and lower value inputs.

TABLE 8-5. PointTolerance Meas Properties (continued)

Property	Description
Tolerance X \pm (Nominal \pm)	Displays the current tolerance for the x-coordinate.
Tolerance Y \pm (Nominal \pm)	Displays the current tolerance for the y-coordinate.
Update Nominal at Runtime (Nominal \pm)	When enabled (checked), the coordinates of the point datum in Nominal Input are written to Nominal X and Nominal Y before each run and used as the nominal for tolerancing the input point.
Update Nominal (Nominal \pm)	When clicked, the coordinates of the point datum in Nominal Input are written to Nominal X and Nominal Y.
Update Statistics (Nominal \pm) (Upper/Lower)	<p>The output StatisticsDm contains the statistical information for the distance between the measured point and the nominal point: mean, maximum, minimum, standard deviation of the distance, and number of data collected.</p> <p><i>Always</i> — The statistical information is collected for each run or trial run of the step.</p> <p><i>Only On Pass</i> — The statistical information is collected only when the part passes the tolerance step.</p> <p><i>Only On Fail</i> — The statistical information is collected only when the part fails the tolerance step.</p> <p><i>Never</i> (Default) — Never updates the statistics. Unwanted statistics information may be collected on any trial runs when set to any setting other than <i>Never</i>.</p>

TABLE 8–5. PointTolerance Meas Properties (continued)

Property	Description
Upper X (Upper/Lower)	Holds the x-coordinate of the upper range value that is used in the comparison process, expressed in either pixel space or in world space. Default: 0.0
Upper Y (Upper/Lower)	Holds the y-coordinate of the upper range value that is used in the comparison process, expressed in either pixel space or in world space. Default: 0.0
Use Calibration (Nominal \pm) (Upper/Lower)	Allows toggling between the world space and the pixel space in which the nominal coordinates, upper and lower ranges, tolerances and statistics information are expressed.
With Tolerance (Nominal \pm)	Allows selection of a tolerance level: 1% — 1% of the corresponding nominal value. 5% — 5% of the corresponding nominal value. This is the default. 10% — 10% of the corresponding nominal value. 3 <i>Std Deviation</i> — The tolerance is set to be three times the current standard deviation. As Is — The current tolerance is not changed when Update Nominal or the step is <i>Trained</i> .

Training

PointTolerance Meas does not need to be trained in order to run. When trained, if the step is set up as the Nominal \pm type, the value in **Nominal Input** is written to the nominal property. The tolerance value is adjusted according to the **With Tolerance** setting.

Results

- **Status** — Indicates whether or not the measured point is within tolerance of the nominal value or within the specified upper/lower range.
- **Number of Passes** — The number of passes.
- **Number of Fails** — The number of failures.

- Statistics — The statistical information about the distance between the measured point and the nominal point: mean, maximum, minimum, standard deviation of the distance, and number of data collected.

I/O Summary

PointTolerance Meas provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

- If Nominal \pm is selected as the Tolerance Type:

Inputs: Nominal Point: (xxx, xxx). Tolerance: (xxx, xxx). yyy
Update Statistics. zzz Update Nominal at Runtime

- If Upper/Lower is selected as the Tolerance Type:

Inputs: Lower Point: (xxx, xxx), Upper Point: (xxx, xxx)yyy Update
Statistics.

Outputs: Measured Pt: (xxx, xxx), Dist = ddd, Statistics: Min=sss
Avg=sss Max=sss StDev=sss

Where: ddd = Distance between the two points

sss = The statistics data taken from the tool executions

xxx = Points, Point Tolerance, and Upper and Lower Points

yyy = Always, Never, Only On Pass, or Only On Fail

zzz = Do NOT or a blank space

Error Message:

- PointTolerance Meas measurement failed

Expressions

The Expression Datum utilizes a special language to interpret expressions and provide a result. Based on C, the expression language can combine datums of different steps for computation.

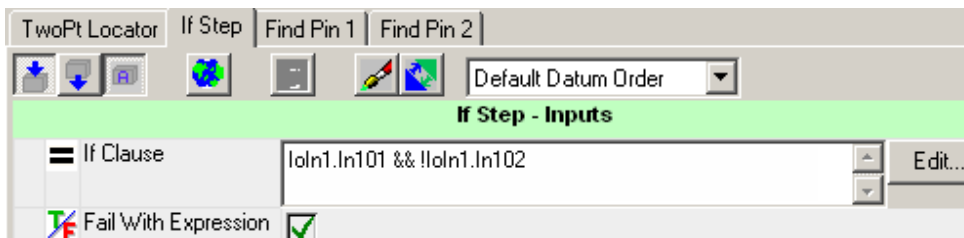
Other Steps Used

None.

Theory of Operation

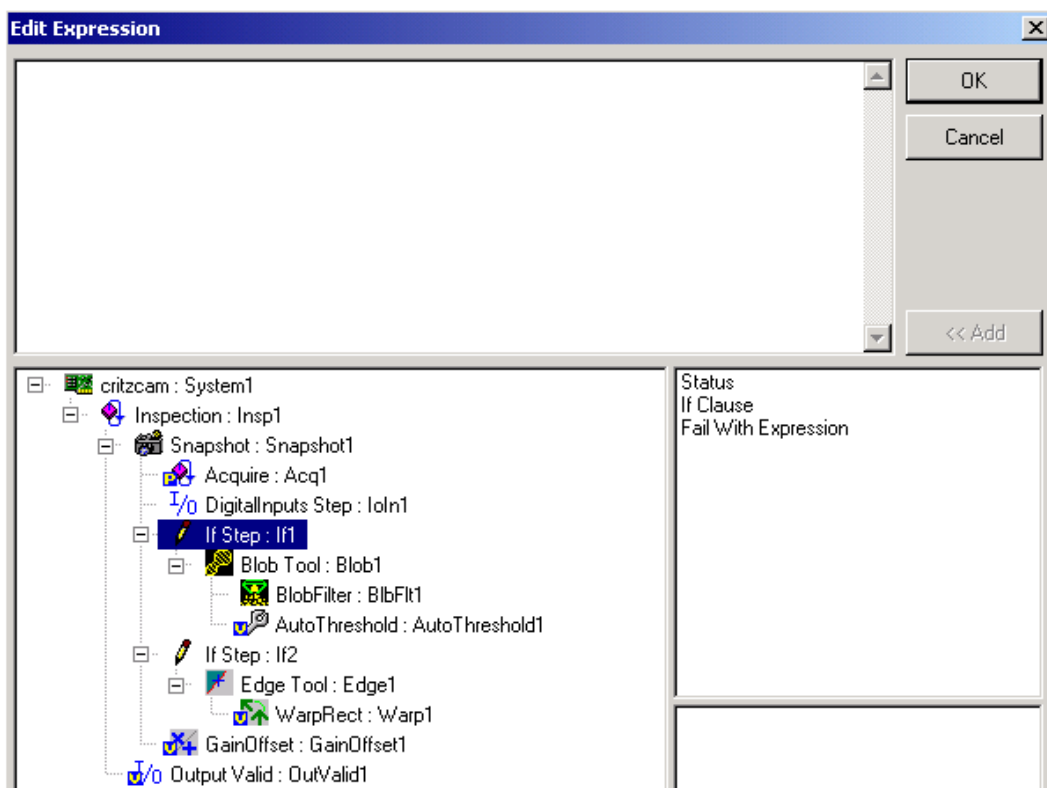
The Expression Datum provides interpreted expressions for different steps. The user interface for the Expression Datum is a multi-line edit control using a fixed width font for readability. You can enter an expression, and it will be compiled, but it is up to the owner of the expression datum to evaluate the expression at runtime. An example of an If Step that uses an expression is shown in Figure 8–25.

FIGURE 8–25. If Step Example

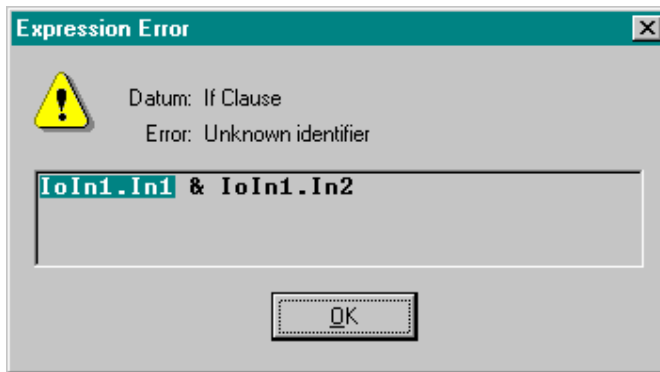


You can either type directly into the expression field or click Edit... to start a multi-line edit window, as shown in Figure 8–26.

FIGURE 8–26. Edit Expression Dialog Box



Error messages are reported in the Expression Error dialog box, as shown in Figure 8–27. This dialog box displays the expression, highlights the error, and describes the error. To exit, click OK.

FIGURE 8–27. Expression Error Dialog Box

Settings

The Expression Datum uses a simple language for evaluating expressions:

- **Identifier** — The symbolic name of a datum or datum field in the program, typically in the format Step1.Datum or Step1.Datum.Field. Identifiers may contain letters, numbers, and underscores, but must start with a letter. To specify datums within a particular step, the symbolic names of the steps must be delimited by a period (.).
- **Literals** — Numbers stored as IEEE floating-point values (e.g., 7, 1.0, 3.1459, and so on).
- **String Literal** — Set of characters surrounded by quotation marks e.g., “an example”.

- Symbols — Special symbols used by the compiler for specific operations. Refer to Table 8–6, Table 8–7, and Table 8–8, which list symbols and operators used by the expression compiler.

TABLE 8–6. Compiler Symbols

Symbol	Name	Usage	Notes or Result
,	Comma	<func>(<expr1>,<expr2>,...)	Separates sub-expressions in multi-parameter functions.
(Open Parentheses	(<expr>)	Opens a sub-expression.
)	Closed Parentheses	(<expr>)	Closes a sub-expression.
{	Open comment	<expr1> <bin_op> { any text in here } <expr2>	You can comment (ignore) text within the braces. The usage example shows a comment within a binary operation.
}	Close comment	<expr1> <bin_op> { any text in here } <expr2>	You can comment (ignore) text within the braces. The usage example shows a comment within a binary operation.
//	Comment to end	<expr> // comment here	Comments text to end of expression.

TABLE 8–7. Unary Operators

Symbol	Name	Usage	Notes or Result
-	Unary minus	-<expr1>	Negates expr1.
!	Boolean Not	!<expr1>	Toggles the Boolean value of expr1.

TABLE 8–8. Binary Operators

Symbol	Name	Usage	Notes or Result
=	Is equal to	<expr1> = <expr2>	TRUE if expr1 is equal to expr2.
!=	Is not equal to	<expr1> != <expr2>	TRUE if expr1 is not equal to expr2.
<	Less-than	<expr1> < <expr2>	TRUE if expr1 is less than expr2.
<=	Less-than or equal to	<expr1> <= <expr2>	TRUE if expr1 is less than or equal to expr2.
>	Greater-than	<expr1> > <expr2>	TRUE if expr1 is greater than expr2
>=	Greater-than or equal to	<expr1> >= <expr2>	TRUE if expr1 is greater than or equal to expr2.
+	Addition	<expr1> + <expr2>	Sets expression to value of expr1 plus expr2.
-	Subtraction	<expr1> - <expr2>	Sets expression to value of expr1 minus expr2.
*	Multiplication	<expr1> * <expr2>	Sets expression to value of expr1 multiplied by expr2.
/	Division	<expr1> / <expr2>	Sets expression to value of expr1 divided by expr2.
%	Modulus	<expr1> % <expr2>	Sets expression to value of expr1 mod expr2. Expr1 and expr2 are first converted to integers (i.e., truncated).
^	Power	<expr1> ^ <expr2>	Sets expression to expr1 raised to the power of expr2.
&&	Boolean AND	<expr1> && <expr2>	Sets expression to TRUE if expr1 and expr2 are both TRUE.
	Boolean OR	<expr1> <expr2>	Sets expression to TRUE if either expr1 or expr2 are TRUE.
^^	Boolean XOR	<expr1> ^^ <expr2>	Sets expression to TRUE if either expr1 or expr2 are TRUE but not both.

Keywords — Case-insensitive words used by the compiler to specify an internal function. Refer to Table 8–9 for a list of case-insensitive supported keywords.

TABLE 8–9. Keyword Functions

Symbol	Usage	Notes or Results
ABS	abs ()	Returns the absolute value of a number.
ACOS	acos (<expr>)	Sets expression to arccosine of expr.
ASIN	asin (<expr>)	Sets expression to arcsine of expr.
ATAN	atan (<expr>)	Sets expression to arctangent of expr.
ATAN2	atan2 (<expr1>,<expr2>)	Sets expression to the arctangent of expr1/expr2.
COS	cos (<expr>)	Sets expression to cosine of expr.
COSH	cosh (<expr>)	Sets expression to hyperbolic cosine of expr.
IF	if (<expr1>, <expr2>, <expr3>)	Sets expression to expr2 if expr1 is TRUE, otherwise, sets to expr3.
INSTR	instr (<string1>,<string2>,<integer>)	Searches for an instance of string2 within string1, optionally with case sensitivity. Returns the zero-based index of the substring or -1 if not found. Same string/integer rules of strcmp apply.
SIN	sin (<expr>)	Sets expression to sine of expr.
SINH	sinh (<expr>)	Sets expression to hyperbolic sine of expr.
STRCOMP	strcmp (<string1>,<string2>,<integer>)	Compares two strings, optionally with case sensitivity. The strings can be either string literals or String Datum types. The integer value is either zero (case-sensitive) or non-zero (case-insensitive). The integer value can also be a scalar datum type. The function returns -1 (string1 < string2), 0 (string1 = string2), or 1 (string1 > string2).
STRLEN	strlen (<string>)	Returns the length of the given string.
TAN	tan (<expr>)	Sets expression to tangent of expr.
TANH	tanh (<expr>)	Sets expression to hyperbolic tangent of expr.
WORLD	world (<id>)	Returns the value of the identifier specified by <id> in calibrated space (e.g., world(pointval.x))

Training

None.

Results

For specific results of different expressions, refer to the following tables:

- Table 8–6, “Compiler Symbols,” on page 8-62
- Table 8–7, “Unary Operators,” on page 8-62
- Table 8–8, “Binary Operators,” on page 8-63
- Table 8–9, “Keyword Functions,” on page 8-64

Expression Datum Rules

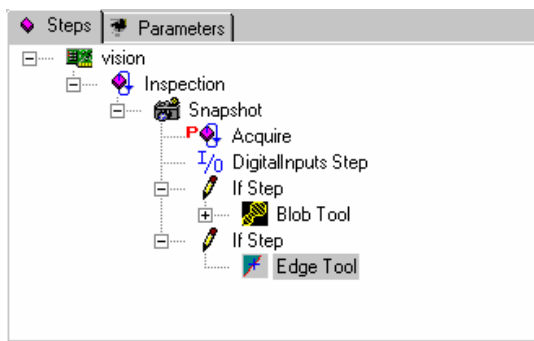
The Expression Datum evaluates an expression and sets its value based on that evaluation. Essentially, an expression accesses Datum values in the Job Tree and performs some function or calculation. When constructing an expression, you must comply with the following rules:

- Steps and Datums in the tree are accessed using symbolic names. The symbolic name is uniquely assigned and created by the step or datum when it is inserted in the Job Tree. You cannot change this symbolic name. To view symbolic names, right-click on the Job Tree and select **Show Symbolic Names**. As a result, each step in the Job Tree shows a string Name : SymName. Click the **Parameters** tab to view the extended names and symbolic names associated with each datum.
- You can access any scalar or string datum in the Job Tree. This includes steps executed after a preceding step whose expression has been edited.
- While step symbolic names are case-sensitive, functions are not.
- Some datums in the tree have fields, which access parts of a datum. Point, for example, contains field x and field y. Other datum types such as Angle, Distance, Double, Integer, and Expression are directly accessible. The two most common datums whose fields you will access are Point (x, y, angle) and Line (a, b, c).
- There is no assignment operator (=). The = sign is always used as a comparison operator.

Expression In Flow Control

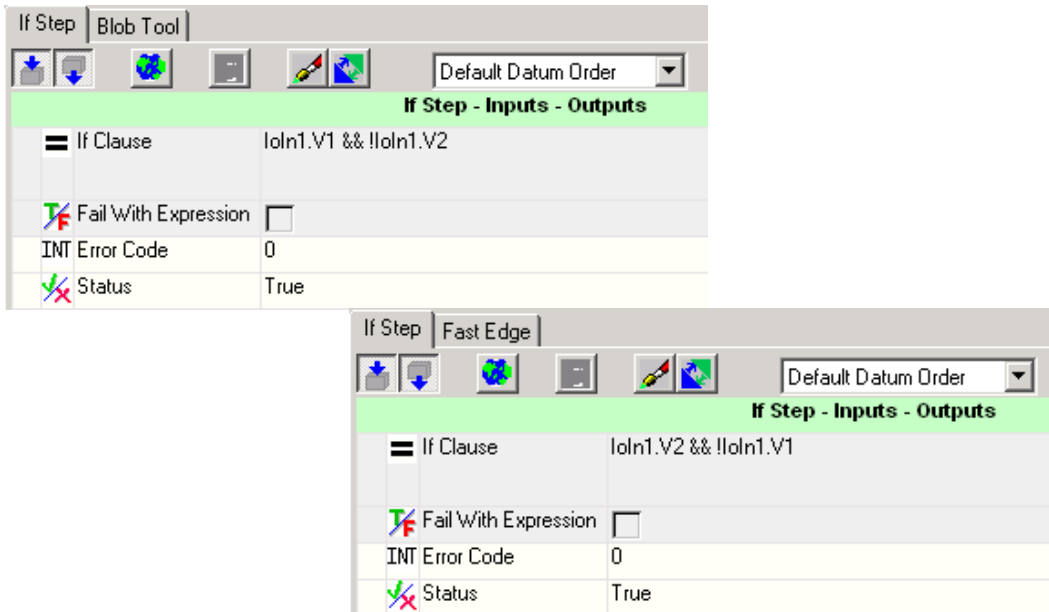
The If step uses an expression IfClause for flow control. The IfClause expression is evaluated and the step optionally executes its children based on the expression. For example, you can create a Job that reads the I/O using DigitalInputs, and then executes a particular branch of the tree based on the I/O states using If steps, as shown in Figure 8–28.

FIGURE 8–28. Flow Control Tree with If Step



In the execution of this Job Tree, the inspection first takes a picture with Snapshot, and then samples the DigitalInputs. The If Clause in each If Step can be set up to read a particular input signal, as shown in Figure 8–29.

FIGURE 8–29. If Step Expressions



The clauses in each step are set up to get the value of Virtual Input 1 and 2 and to execute the Blob Tool only when 1 is high (1) and 2 is low (0), or to execute the Edge Tool only when 2 is high (1) and 1 is low (0). The use of the symbolic name allows access to Virtual Input 1 and 2. When the DigitalInputs step is executed, it samples the I/O and stores the I/O states in a set of integer datums accessible by expressions. The expressions also use the && symbol, which evaluates an AND of two sub-expressions, and the ! symbol, which evaluates the NOT of a sub-expression.

Expression As Result Mechanism

DigitalOutputs, Inspection, and VarAssign Steps use the Expression Datum as a result mechanism. Each step contains an Expression Datum, evaluates the expression, and uses the evaluation as a result.

DigitalOutputs represents all output I/O signals from the vision device, including physical and virtual I/O. In this step, you create expressions for any of the I/O points, and each expression is subsequently evaluated. The signal is set when the expression is non-zero, and is cleared when the expression is zero. Typically, the expression indicates an individual step failure through I/O (e.g., Step.Status). It can also be used as a signal generator to the host PC (virtual I/O), or to a PLC (physical I/O). In this case, the expressions are set to 0 or 1.

The Inspection Step contains an expression that evaluates the pass/fail status of the inspection. By default, the inspection combines the pass/fail results of all contained steps as its own pass/fail status. Therefore, when a step fails, the inspection fails. However, you can override this mechanism and provide your own expression to evaluate the pass/fail status. This expression is evaluated after all contained steps have executed.

The VarAssign Step uses an Expression Datum to evaluate an expression, and sets an Output Value datum to the result. It executes calculations and stores the results in memory to be used by other expressions.

Typically, an expression is a function that sets a result based on a computation or value. The if function in the expression language uses the format `if(expr1,expr2,expr3)`. When `expr1` evaluates to non-zero, the result of the function is `expr2`. Otherwise, it is `expr3`.

Expression In Measurements

The Expression Datum can evaluate measurements. For example, you can use an expression to access the parts of a Point or Line (either in pixel or calibrated space) and check distances. The Point and Line datums each contain three distinct fields, which can be used within an expression:

- Point — (x,y, and angle)
- Line — (a, b, and c)

The function `World` allows you to use the calibrated value of these fields (or of any angle, area, or distance) in calibrated world space. For example, `World(Point.x)` is a sub-expression that returns the value of `Point.x` in calibrated world space.

Expression In String Comparisons

The Expression Datum evaluates string comparisons. For example, the Data Matrix and BarCode Tools read symbols and output a corresponding string value. An expression can compare the resulting value against a known good value, search for a substring within the value, or check the length of the string. The functions StrComp, InStr, and StrLen perform these functions respectively. Both StrComp and InStr take two string parameters and a third integer value to indicate case-sensitivity.

CHAPTER 9

Automatic Identification and Symbol Quality Verification

This chapter discusses the Symbology Tool, OCRTrainableFont Tool, IntelliText OCR, and Symbol Quality Verification.

Symbology Tool

Other Steps Used

None.

Theory of Operation












The Symbology Tool locates and decodes both 1D and 2D encoded symbols. It searches the specified ROI looking for an enabled symbology type and attempts to decode it. If successful it will continue to process the ROI until the ROI is completely traversed, the maximum symbol count has been met, or the advanced timeout feature has been reached.

The symbology tool can be trained to optimize performance for the symbology type decoded. Once trained, the tool will only read the symbology trained, and depending on the symbology may also discriminate other specifications for performance. For this reason training is only recommended for tools that will be used to read symbols of the same symbology, size, and polarity.

Settings

- **Train String Only** — After configuring the symbology tool and setting its match string enabled to enabled, when you select the train button in train and tryout, only the match string field will be updated. None of the other parameters will be modified. However, you will remain in a trained state.
- **Match String Enable** — When enabled, the match string will be compared against the decoded string of a symbol during a run to determine if the step is run successfully. The result datum will be true if a symbol is successfully decoded and the decoded string matches the match string value when this box is checked. During training, the decoded string from the training symbol will be placed in the match string value box only when this property is enabled.
- **Use WildCard ? in Match String** — When enabled '?' can be used in the match string as a placeholder wildcard match and any character will be considered a match for that position.
- **GS1 Format Enable** — When set to enabled, GS1 validation checking will be done on decoded symbols. Symbols that fail this check will cause a GS1 status failure and set the GS1 Format text output datum indicating the error.

Symbols that pass the check will provide formatted GS1 text and convert the identified application fields into output datums (see example below).


	Decoded Text	01006141419999961750123110123ABC21SC50
	GS1 Format Text	(01)00614141999996(17)501231(10)123ABC(21)SC50
	PT Center Point	(994.000,601.500)
	Angle	0.000
	Height	227.000
	Width	226.000
	Symbol Results	<Click to Display Values>
	GS1_ApplD01	00614141999996
	GS1_ApplD17	501231
	GS1_ApplD10	123ABC
	GS1_ApplD21	SC50



Omron Microscan is a GS1 Solution Partner with a Certified GS1 Bar Code Professional on staff. This allows us to provide you with the background knowledge necessary to implement GS1 Standards successfully in your application.

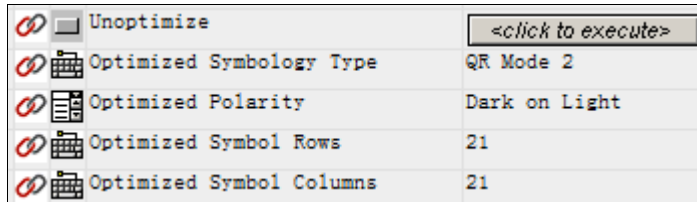
Visit www.gs1us.org for more information about why to use a GS1 Solution Partner.

- **Match String** — The entered string will be compared against each decoded string during each run when match string is enabled. This datum will be filled in after a successful train only when enable match string is enabled while training.
- **Timeout (ms)** — Maximum time the tool spends searching for a symbol. When a symbol is not found and decoded by this time, this step fails and the program execution continues.
- **Max Symbol Count** — This sets the maximum number of symbols that can be decoded within a single ROI.
- **Enabled Symbols** — Only enabled symbols will be located within the ROI:
 - Data Matrix ECC 200
 - Data Matrix Non-ECC 200
 - QR Code 2005
 - Aztec Code

- Code 39
 - Code 128
 - BC412
 - Interleaved 2 of 5
 - Codabar
 - UPC/EAN
 - Code 93
 - Pharmacode
 - GS1 DataBar
 - Postal Symbolologies
 - PDF417 / MicroPDF417
 - Composite
 - DotCode
- **Auto Teach Options** — Selects the functionality used when the new master pin on a smart camera is used in conjunction with a triggered read to teach the step. When set to train step the step is trained whereas when set to learn match string the match string is learned from the decode.
 - **Train** — The Train button saves information relevant to the target symbol (if decoded) to allow higher readability for subsequent similar symbols. This mode of operation remains active until the call is made to disable it and revert to normal operation. Note that it un-optimizes the system once the system is trained. You can subsequently attempt to optimize the system after being trained.

 - **Optimize** — The Optimize button saves information relevant to the target symbol (if decoded) to allow quicker reading and greater consistency for subsequent similar symbols. This mode of operation remains active until the call is made to disable it and revert to normal operation. Note that it disables all symbolologies other than the one that has been optimized.



- When it is optimized, it displays some of the properties of the optimized symbol, and also changes the datum to “Un-optimize” to indicate what will happen if the button is clicked again.



- Graphics Level** — Selects the amount of graphics displayed at the completion of step execution:
 - Show None: No graphics are drawn.
 - Show ROI Only: Only the ROI will be drawn.
 - Show Graphics Except ROI: Data Matrix locating graphics and decoded text will be displayed, but not the ROI.
 - Show Graphics: ROI and Data Matrix locating graphics and any decoded text will be displayed.

Results

- Error Code** — Reports errors encountered when locating or decoding.
- Status** — The status is true when a symbol is found and decoded and additional enabled tests pass. These additional tests include the match string and GS1 format checking.
- Read Status** — The read status is true when any symbol is found and decoded regardless of additional tests.
- GS1 Status** — The GS1 Status is true when symbols decoded adhere to the GS1 format checking.
- Symbol Count** — Indicates the number of symbols decoded within the ROI.
- Symbology Type** — String containing text for the symbology type(s) decoded in the ROI.

- **Decoded Text** — String containing the decoded text of the symbols decoded. When multiple symbols are decoded, enumerated output datums are also provided for each symbol.
- **Center Point** — This is the value of the center location of the decoded symbol. When multiple symbols are decoded enumerated output datums are also provided for each symbol.
- **Angle** — This is the value of the center location of the decoded symbol. When multiple symbols are decoded enumerated output datums are also provided for each symbol.
- **Height** — This is the value of the height of the decoded symbol. When multiple symbols are decoded enumerated output datums are also provided for each symbol.
- **Width** — This is the value of the width of the decoded symbol. When multiple symbols are decoded enumerated output datums are also provided for each symbol.
- **Symbol Results** — This is a structure containing additional symbology decode details.

Advanced Symbology Settings

BC412

- **Check Character Output Status** — When enabled, the check character character is read and compared along with the symbol data. When disabled, symbol data is sent without the check character.
- **Fixed Symbol Length Status** — When enabled, Visionscape will check the symbol length against the symbol length field. If disabled, any length will be considered valid.
- **Symbol Length** — When enabled, the check character is read and compared along with the symbol data. When disabled, symbol data is sent without the check character.

Codabar

- **Start and Stop Match Status** — When disabled, Visionscape will decode Codabar symbols whether or not the start and stop characters are the same.

When enabled, Visionscape will not decode Codabar symbols unless the start and stop characters are the same.

- **Start and Stop Output Status** — When disabled, the start and stop characters will *not* be present in the data output of the decoded symbol. When enabled, the start and stop characters *will* be present in the data output of the decoded symbol.
- **Large Inter character Gap Status** — When disabled, the spaces between characters, or the “inter character gap”, are ignored during the decode process. **Note:** If the inter character space is large enough to be considered a margin, the symbol will not decode, regardless of this parameter’s setting.
- **Fixed Symbol Length Status** — When disabled, Visionscape will accept any Codabar symbol provided it doesn’t exceed the system’s maximum capabilities. When enabled, Visionscape will reject any Codabar symbol that doesn’t match the fixed length.
- **Symbol Length** — This is the value against which all Codabar symbol lengths will be compared.
- **Check Character Type** — When disabled, Visionscape will not perform any character checking calculations on decoded Codabar symbols. When set to **Mod 16**, Visionscape will perform a modulus 16 check character calculation on the symbol. If the symbol does not pass this calculation, it will not be decoded. When set to **NW7**, Visionscape will perform an NW7 modulus 11 check character calculation on the symbol. If the symbol does not pass this calculation, it will not be decoded. When set to **Both**, Visionscape will perform both the Mod 16 and NW7 modulus 11 check character calculations on the symbol. If the symbol does not pass either calculation, it will not be decoded.
- **Check Character Output Status** — When this field is disabled and a check character calculation is enabled, Visionscape will strip the verified check character from the symbol data output. This condition must be accounted for if a fixed length is also being used. When enabled, Visionscape will output the check character as part of the symbol data. This condition must be accounted for if a fixed length is also being used.

Code 39

- **Check Character Status** — When enabled, outputs the status of the Check Character along with symbol data.

- **Check Character Output Status** — When enabled, the check character is read and compared along with the symbol data. When disabled, symbol data is sent without the check character. **Note:** With **Check Character Output Status** and an **External** or **Serial** trigger option enabled, an invalid check character calculation will cause a No Read message to be transmitted at the end of the read cycle.
- **Large Intercharacter Gap Status** — When enabled, Visionscape can read symbols with gaps between symbol characters that exceed three times (3x) the narrow element width.
- **Fixed Symbol Length Status** — When enabled, Visionscape will check the symbol length against the symbol length field. If disabled, any length will be considered valid.
- **Symbol Length** — Specifies the exact number of characters that Visionscape will recognize (this does not include start and stop and check character characters). Visionscape ignores any symbology that does not match the specified length.
- **Full ASCII Set Status** — Standard Code 39 encodes 43 characters; zero through nine, capital “A” through capital “Z”, minus symbol, plus symbol, forward slash, space, decimal point, dollar sign, and percent symbol. When **Full ASCII Set** is enabled, Visionscape can read the full ASCII character set, from 0 to 255.

Code 93

- **Fixed Symbol Length Status** — When disabled, Visionscape will accept any Code 93 symbol provided it doesn’t exceed the system’s maximum capabilities. When enabled, Visionscape will reject any Code 93 symbol that doesn’t match the fixed symbol length.
- **Symbol Length** — This is the symbol length value against which all Code 93 symbols will be compared.

Code 128

- **Fixed Symbol Length Status** — When enabled, Visionscape will check the symbol length against the symbol length field. If disabled, any length will be considered a valid symbol.
- **Symbol Length** — This specifies the exact number of characters that Visionscape will recognize (this does not include start, stop, and check

character characters). Visionscape ignores any symbol not having the specified length.

- **EAN Status** — When this field is disabled, Visionscape will not check any Code 128 labels for conformance to EAN requirements, or perform any special formatting. When enabled, Visionscape can read symbols with or without a function 1 character in the first position. If a symbol has a function 1 character in the first position, it must conform to EAN format. Symbols that conform to EAN format will also be subject to the special output formatting options available in this command. **Note:** Code 128 status must be enabled for EAN status to be active. If EAN status is required, Visionscape will only decode symbols that have a function 1 character in the first position and that conform to EAN format. All symbols read will be subject to the special output formatting options available in this command. **Note:** Code 128 status must be enabled for EAN status to be active.
- **Output Format** — Allows you to set Code 128 Output Format to Standard or Application.
- **Application Record Separator Status** — When enabled, an EAN separator will be inserted into the output between fields whenever an EAN-conforming symbol is decoded and EAN output formatting applies.
- **Application Record Separator Character** — This is an ASCII character that serves as an EAN separator in formatted EAN output.
- **Application Record Brackets Status** — If an EAN-conforming symbol is decoded and EAN formatting applies, this feature places bracket characters around the application identifiers in the formatted output.
- **Application Record Padding Status** — This feature causes Visionscape to pad variable-length application fields with leading zeroes. This is not done for the last field of a symbol.

Composite

- **Separator Status** — Allows the user to distinguish between the main and **Supplemental** symbols. Separates the linear and the composite component.
- **Separator Character** — Allows the user to change the separator character from a comma to a new character.

GS1 DataBar

- **DataBar Expanded Fixed Symbol Length Status** — When enabled, Visionscape will check the symbol length against the symbol length field, minus the embedded check character. If disabled, any length would be considered valid.
- **DataBar Expanded Symbol Length** — Specifies the exact number of characters that Visionscape will recognize (this does not include start, stop, and check character characters). Visionscape ignores any symbol not having the specified length.

Interleaved 2 of 5

- **Check Character Status** — An error correcting routine in which the check character is added.
- **Check Character Output Status** — When enabled, a check character character is sent along with the symbol data for added data security.
- **Symbol Length 1** — The **Symbol Length # 1** field is one of two fields against which the decoded symbol is compared before accepting it as valid or rejecting it.
- **Symbol Length 2** — The **Symbol Length # 2** field is one of two fields against which the decoded symbol is compared before accepting it as valid or rejecting it.
- **Guard Bar Status** — Whenever **Guard Bar** is enabled, the presence of guard bars (also called “bearer bars”) is required for decoding to take place.
- **Range Mode Status** — When **Range Mode** is disabled, Visionscape checks the value of the symbol length against the values set in **Symbol Length # 1** and **Symbol Length # 2**. If the symbol length does not match either of the preset values, then it is rejected as invalid. When **Range Mode** is enabled, **Symbol Length # 1** and **Symbol Length # 2** are combined to form a range of valid symbol lengths. Any symbol length that does not fall into this range is rejected as an invalid symbol. Either of the preset symbol length values in the **Symbol Length # 1** and **Symbol Length # 2** fields can form the start or end of the range.

PDF417

- **Symbol Length** — When enabled, the PDF417 symbol must contain the same number of characters as the symbol length setting before it can be considered a good read. Visionscape will ignore any symbol not having the specified length.
- **Codeword Collection** — Allows the user to determine whether Visionscape will attempt to decode the PDF417 symbol only from the information provided in a single image or in multiple images.

MicroPDF417

- **Fixed Symbol Length Status** — When enabled, Visionscape will check the symbol length against the symbol length field. If disabled, any length will be considered valid.
- **Symbol Length** — When enabled, the MicroPDF417 symbol must contain the same number of characters as the symbol length setting before it can be considered a good read. Visionscape will ignore any symbol not having the specified length.

Pharmacode

- **Symbol Length** — Specifies the exact number of bars that must be present for Visionscape to recognize and decode the Pharmacode symbol.
- **Minimum Bars** — Sets the minimum number of bars that a Pharmacode symbol must have to be considered valid.
- **Bar Width** — If set to **Mixed**, Visionscape will autodiscriminate between narrow bars and wide bars. If set to **All Narrow**, all bars will be considered as narrow bars. If set to **All Wide**, all bars will be considered as wide bars. If set to **Fixed Threshold**, it will use the fixed threshold value to determine whether the bars are narrow or wide. The **Bar Width Status** setting will be ignored when Visionscape is able to tell the difference between the narrow and the wide bars.
- **Direction** — Specifies the direction in which a symbol can be read.
- **Fixed Threshold Value** — Used when **Bar Width Status** is set to **Fixed Threshold**. Defines the minimum difference in pixels that will distinguish a narrow bar from a wide bar.
- **Background Color** — Allows the user to determine if the background color is white, black, or both.

UPC

- **Supplementals Status** — A supplemental is a 2 to 5 digit symbol appended to the main symbol. When set to **Enabled** or **Required**, Visionscape reads supplemental code data that has been appended to the standard UPC or EAN symbols.
- **Separator Status** — A character can be inserted between the standard UPC or EAN symbology and the supplemental symbology when **Supplementals** is set to **Enabled** or **Required**.
- **Separator Character** — Allows the user to change the separator character from a comma to a new character.
- **Supplementals Type** — Allows the user to select 2 character or 5 character supplements, or both.
- **UPC-E as UPC-A Status** — When disabled, Visionscape will output the version E symbols in their encoded 6-character format. When enabled, Visionscape will format the symbol as either a 12-character UPC-A symbol or an EAN-13 symbol, depending on the state of the EAN status parameter. This formatting reverses the zero suppression that is used to generate the symbol in the UPC specification.

OCRTrainableFont Tool

This tool reads labels and marks and returns string results. Any mark and label symbol can be trained incrementally at the time it is first seen, or offline. Characters and symbols can have any shape or size, as long as they can be mapped to an ASCII character. There are no limits on the number of characters; however, character sets must be combined in groups of at most 45. Each set of 45 characters constitutes a single font. Execution time is proportional to the number of fonts that read the input mark or label.

At the Step level, the OCRTrainableFont Tool encapsulates the user interface elements to configure the Segment Agent, the FeatExtract Agent, the classifier inputs (Font files), and confidence thresholds.

Font files are stored into text files and will be installed in a “Fonts” directory at install time inside the Vscape\Jobs directory. A font file consists of the following files:

- A fontname.nnd file contains runtime data necessary for OCR to read labels and marks using that Font.
- A fontname.nna file contains the data necessary to incrementally train a font.
- A fontname.nnc file contains the data defining the Font alphabet (ASCII characters) and statistics information necessary to support confidence levels.

By default, a special Font called new (i.e., new.nnd) is always available and is necessary both for creating new fonts and for use at runtime by the OCR tool itself.

The OCRTrainableFont Tool provides both runtime and training capability for the OCR algorithm:

- Runtime — The tool is responsible for reading a mark or label within an ROI using one or more Fonts and reporting results, including the string read and confidence levels for each character read.
- Training — Fonts can be trained using the OCR Tool in AutoVISION Software. Fonts trained in AutoVISION can be used by the OCRTrainableFont Tool.

Theory of Operation

When an OCRTrainableFont Tool is inserted into a Visionscape Job, the settings that control the behavior of the Segment Agent and the FeatExtract Agent, as well as the font parameter settings, are available on the tool's property page. The settings on the property page can modify the way in which the OCRTrainableFont Tool functions:

- If ROI Contains is set to **Two or More Characters**, see the properties shown in Figure 9–1.
- If ROI Contains is set to **Single Character**, see the properties shown in Figure 9–2.

FIGURE 9–1. OCRTrainableFont Tool Properties Page (Two or More)

The screenshot shows the OCRTrainableFont Tool Properties Page. At the top, there is a title bar 'OCRTrainableFont Tool' and a toolbar with icons for file operations, a 'Default Datum Order' dropdown, and a 'Snapshot' button. Below the toolbar is a green header 'OCRTrainableFont Tool - Inputs'. The main area is a table of properties:

InputBuffer	Snapshot.SnapOutputBuffer
Match String Enable	<input type="checkbox"/>
Match String	N/A
ROI Contains	Two or More Characters
Polarity	Dark Characters
Adaptive Thresholding	<input checked="" type="checkbox"/>
Threshold	1
Threshold Bias	0
Edge Energy Threshold	20
Discard Boundary Characters	<input checked="" type="checkbox"/>
Allow Overlapped Characters	<input checked="" type="checkbox"/>
Allow Segmented Characters	<input type="checkbox"/>
Min Character Area (pixels)	40
Limit Character Width	<input type="checkbox"/>
Limit Character Height	<input type="checkbox"/>
Selected Font(s)	OCRTFDemo
Set Unknown Characters to:	?
Pass On No Data	<input type="checkbox"/>
Check Character Confidence	<input checked="" type="checkbox"/>
Minimum Confidence	70
Limit the Number of Characters	<input type="checkbox"/>
Graphics Level	Show ROI Only

FIGURE 9–2. OCRTrainableFont Tool Properties Pages (Single)

OCRTrainableFont Tool

Default Datum Order

OCRTrainableFont Tool - Inputs - Output

InputBuffer	Snapshot.SnapOutputBuffer
Match String Enable	<input type="checkbox"/>
Match String	N/A
ROI Contains	Single Character
Character Fills Entire ROI	<input type="checkbox"/>
Polarity	Dark Characters
Adaptive Thresholding	<input checked="" type="checkbox"/>
Threshold	1
Threshold Bias	0
Edge Energy Threshold	20
Discard Boundary Segments	<input checked="" type="checkbox"/>
Collect All Character Segments	<input type="checkbox"/>
Selected Font(s)	<none>
Set Unknown Characters to:	?
Pass On No Data	<input type="checkbox"/>
Check Character Confidence	<input checked="" type="checkbox"/>
Minimum Confidence	70
Limit the Number of Characters	<input type="checkbox"/>
Graphics Level	Show ROI Only

Table 9–1 lists the OCRTrainableFont Tool properties in alphabetical order, and indicates in which property page the property appears (either Single Character or Two or More Characters).

TABLE 9–1. OCRTrainableFont Tool Settings (Alphabetized)

Property	Description
Adaptive Thresholding (Single Character) (Two or More Characters)	When this option is checked, a Sobel operation will be performed within the search area to determine the pixel value that separates a light from a dark pixel. Otherwise, the value given in the Threshold setting is used. By default, adaptive thresholding is enabled.
Allow Overlapped Characters (Two or More Characters)	When this option is checked, the boundaries of the characters found by the tool can overlap. This is useful for fonts that allow the tops of uppercase characters to extend over the next lowercase character in a row. Default: Enabled
Allow Segmented Characters (Two or More Characters)	When this option is checked, the tool will combine adjacent disconnected parts when searching for a character's boundary. This should be enabled when locating dot-matrix characters. Default: Disabled
Character Fills Entire ROI (Single Character)	Uses the ROI perimeter as the character's bounding box. Use this option when two characters touch each other because the tool will not separate contiguous segments. Ensure that the ROI fits tightly around the character. Default: Disabled
Check Character Confidence (Single Character) (Two or More Characters)	When enabled, the confidence level found by the font reading process for each character will be checked against a user input level. (See Minimum Confidence description.) Default: Enabled
Collect All Character Segments (Single Character)	When this option is checked, the tool combines all segments in the ROI into a single character. The ROI can fit loosely around the character, but different characters must not touch. Default: Enabled

TABLE 9–1. OCRTrainableFont Tool Settings (Alphabetized) (continued)

Property	Description
Discard Boundary Characters (Two or More Characters)	When this option is checked, characters found touching the ROI will be discarded. Default: Enabled
Discard Boundary Segments (Single Character)	(This option is displayed only when the Character Fills Entire ROI option is disabled.) When this option is checked, segments of characters found touching the ROI will be discarded. Default: Enabled
Edge Energy Threshold (Single Character) (Two or More Characters)	Defines the pixel value at which a pixel in a Sobel Edge Enhancement is considered to be an edge pixel. This property is only used when Auto Thresholding Enabled is enabled. Range: 0 to 255; Default: 20
Graphics Level (Single Character) (Two or More Characters)	Enables various graphics options at runtime. Default: Show Basic Graphics
InputBuffer (Single Character) (Two or More Characters)	Allows selection of the buffer to work on from the list of currently available buffers. The default buffer will be the output buffer of its originator. This is usually the output buffer of the closest enclosing Snapshot but can also be the buffer of any step that generates an output image.
Limit Character Height (Two or More Characters)	This allows you to specify a minimum and maximum height for the characters found. Default: No height limits
Limit Character Width (Two or More Characters)	This allows you to specify a minimum and maximum width for the characters found. Default: No width limits

TABLE 9–1. OCRTrainableFont Tool Settings (Alphabetized) (continued)

Property	Description
Limit the Number of Characters (Single Character) (Two or More Characters)	When enabled, the tool will return only a limited number of characters. The characters with the maximum confidence level up to that limit (see Maximum Number of Characters description) will be returned. Default: Disabled
Min Character Area (pixels) (Two or More Characters)	(This option is displayed only when you disable the Allow Segmented Characters option.) This setting specifies the minimum size of a character, thereby eliminating segments that are too small to be characters by themselves. Default: 40
Minimum Confidence (Single Character) (Two or More Characters)	(This option is displayed only when the Check Character Confidence property is enabled.) You can enter a confidence level that the tool must find for each character. Any character that cannot be matched to at least this input level will be returned as the unknown character. Default: 70%
Pass On No Data (Single Character) (Two or More Characters)	When enabled, the status will report passed even if a character is not recognized. When disabled, if any character is reported as unknown, the status will be false.
Polarity (Single Character) (Two or More Characters)	Allows selection of the symbol type for which the tool will search. Choices are Dark Characters or Light Characters, with a default of Dark Characters.
ROI Contains (Single Character) (Two or More Characters)	Provides a choice between Single Character and Two or More Characters. When you select the Single Character option, only one character boundary will be found within the search area. The boundaries of the character will be defined so that they tightly enclose all blobs found. Default: The tool searches for multiple characters.

TABLE 9–1. OCRTrainableFont Tool Settings (Alphabetized) (continued)

Property	Description
Selected Font(s) (Single Character) (Two or More Characters)	This is a list box containing the names of all fonts that have been stored on the system. Whenever a new font is created and trained, its name will appear as an option in this list. You can select from this list one or more fonts that are to be used when reading characters from an image.
Set Unknown Characters To (Single Character) (Two or More Characters)	This is a character that will be returned from the font reading process when no match can be made within the selected font. Default: “?”
Threshold (Single Character) (Two or More Characters)	This value is the threshold above which a pixel's value must be for it to be considered a light pixel. If the Adaptive Thresholding property is enabled, this value is ignored, but the calculated threshold will be displayed. Default: 25
Threshold Bias (Single Character) (Two or More Characters)	This value will be added to the Threshold value before a comparison is made to determine whether a pixel is light or dark. Default: 0

OCR Font Training

1. In FrontRunner, click Editor.
2. Select the OCRTrainableFont tool.
3. Click Show Custom Properties for this Step (Figure 9–3). The custom editor is displayed, as shown in Figure 9–4. The custom editor is used for all font creation and training.

FIGURE 9–3. Show Custom Properties for this Step Button

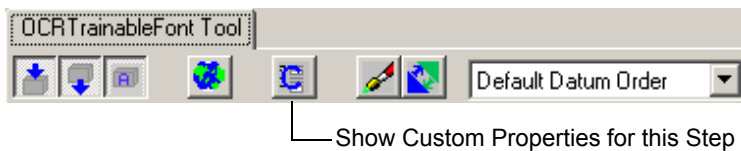
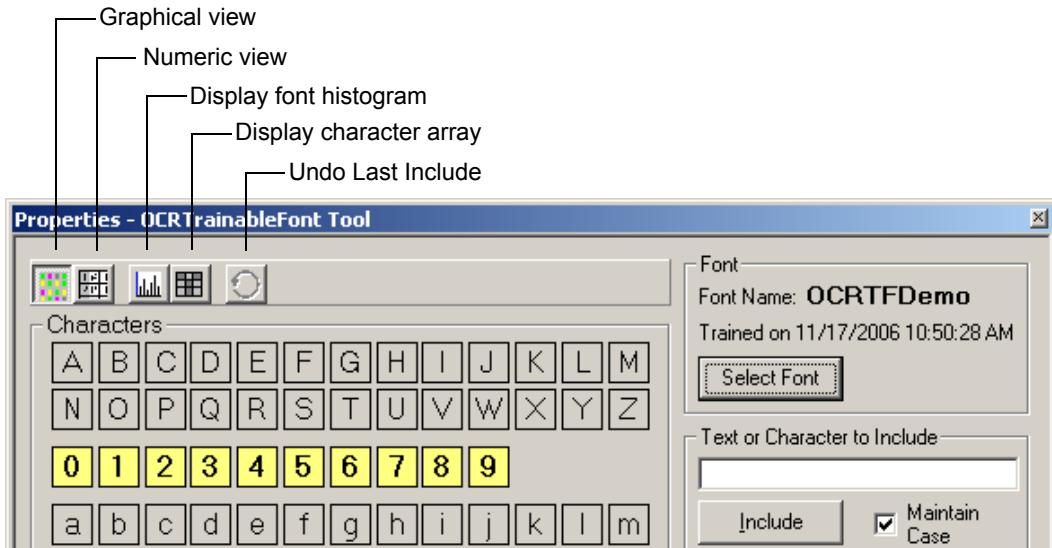


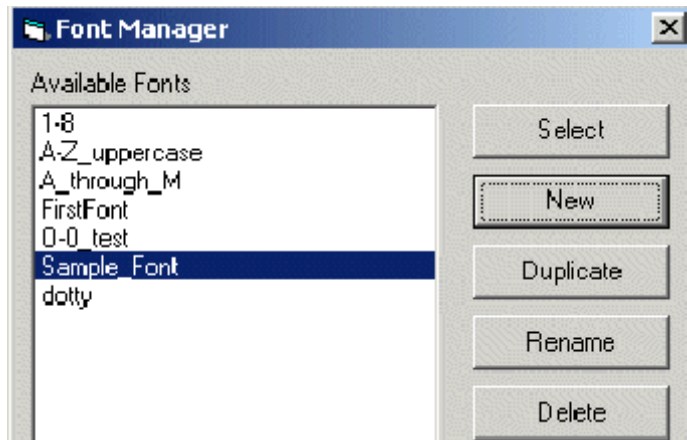
FIGURE 9–4. Custom Editor



4. Minimize the FrontRunner Editor (not the custom editor)
5. In the custom editor, click **Select Font** to select a new font to be trained.

The Font Manager dialog box is displayed, as shown in Figure 9–5.

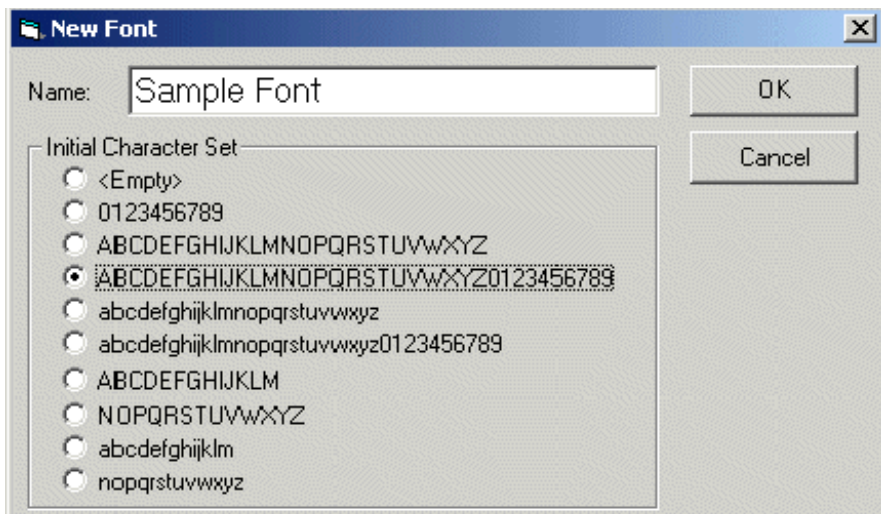
FIGURE 9–5. Font Manager Dialog Box



6. Click New.

The New Font dialog box is displayed, as shown in Figure 9–6.

FIGURE 9–6. New Font Dialog Box



7. Type a name for the new font.
8. Select the initial character set (characters may be deleted or added later).

9. Click OK to close the New Font dialog box.

The Font Manager is displayed again.

10. Highlight the newly-created font and click **Select**.
11. In FrontRunner, click **Acquire New Image** to capture a new image.

Acquiring a new image clears the image of all overlay graphics.
12. Size the ROI to completely surround the characters to be trained, as shown in Figure 9–7.

FIGURE 9–7. Sizing the ROI



13. In the custom editor, type the characters to be trained into the “Text or Characters to Include” text box (Figure 9–7).

Note: Do not type spaces.

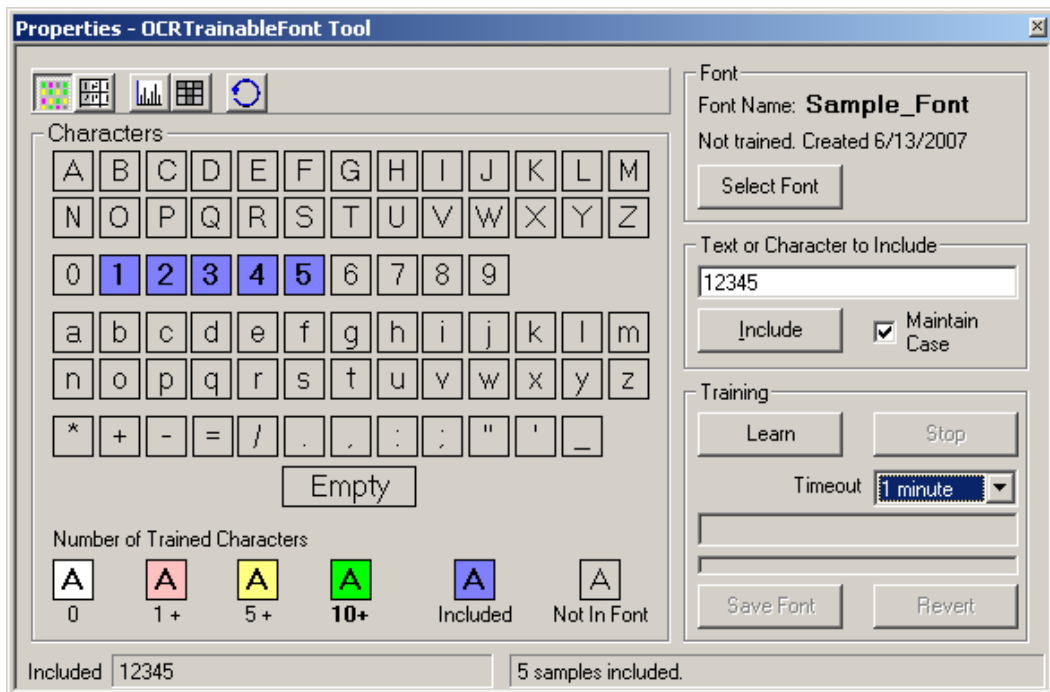
In the image in Figure 9–7, the ROI completely surrounds the characters “12345”. The red lines defining the ROI should not touch the characters to be included.

The Maintain Case check box forces the “Text or Character to Include” to uppercase if the font consists of uppercase characters (ABC). For example, if you type “klm”, the characters will be converted to “KLM” when you click the Include button. Maintain Case also forces characters to lowercase if the font consists of lowercase characters (abc).

14. Click Include.

Recently included characters will be highlighted in blue. Characters remain highlighted until the font is trained and saved. The most recently included characters are displayed in the status bar, as shown in Figure 9–8

FIGURE 9–8. Recently Included Characters Highlighted

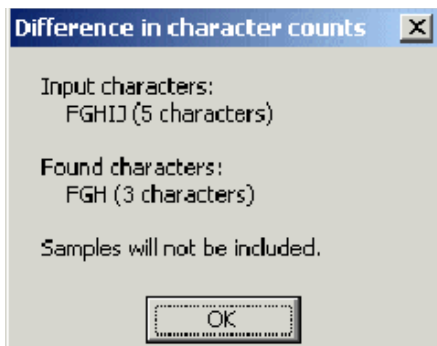


- If the characters do not train well, right click and select Undo Include.

To undo the inclusion of the last set of characters, right click somewhere in the dialog and select the option Undo Include [characters]. There is one level of undo.

Undo is available even after a font is trained.

- If fewer characters than expected are found, a message box appears, as shown in Figure 9–9. Resize the ROI or make adjustments to the OCRTF tool.

FIGURE 9–9. Fewer Characters Found

- If more characters are found than expected, only the found characters are included.
15. Repeat steps 11 - 14 to include numerous character samples in the font.

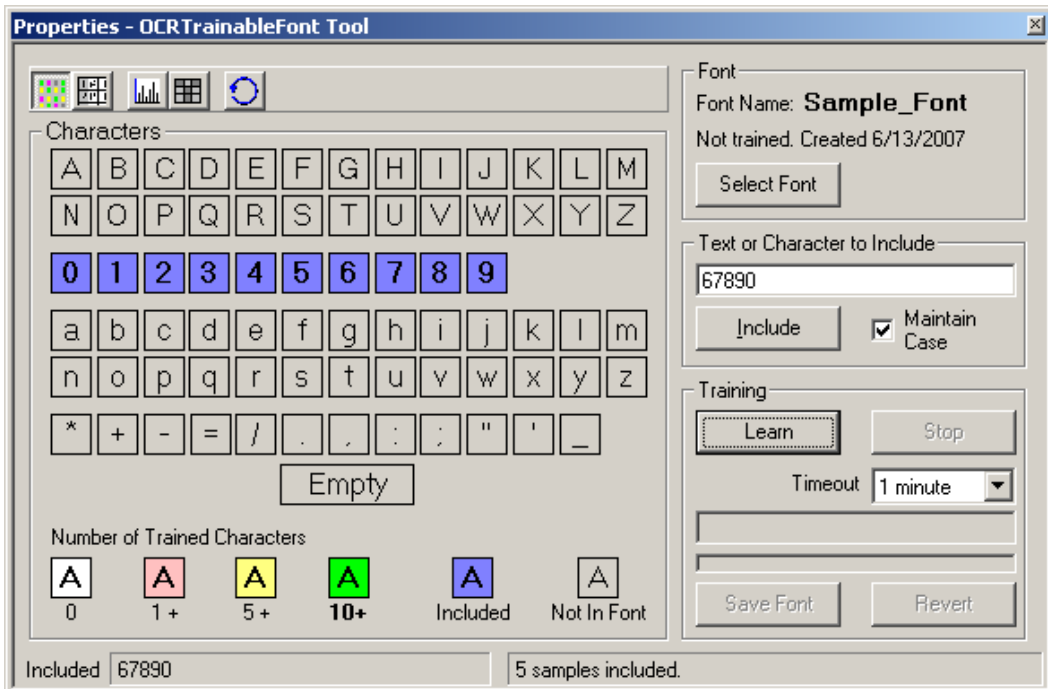
As characters are included, graphics appear in the image for each character.

FIGURE 9–10. Graphics for Each Character

16. Click Acquire New Image to clear the image graphics.

The custom editor is updated to indicate all of the included characters, as shown in Figure 9–11.

FIGURE 9–11. Custom Editor Updated



17. Click **Learn** to train the font using the included characters.

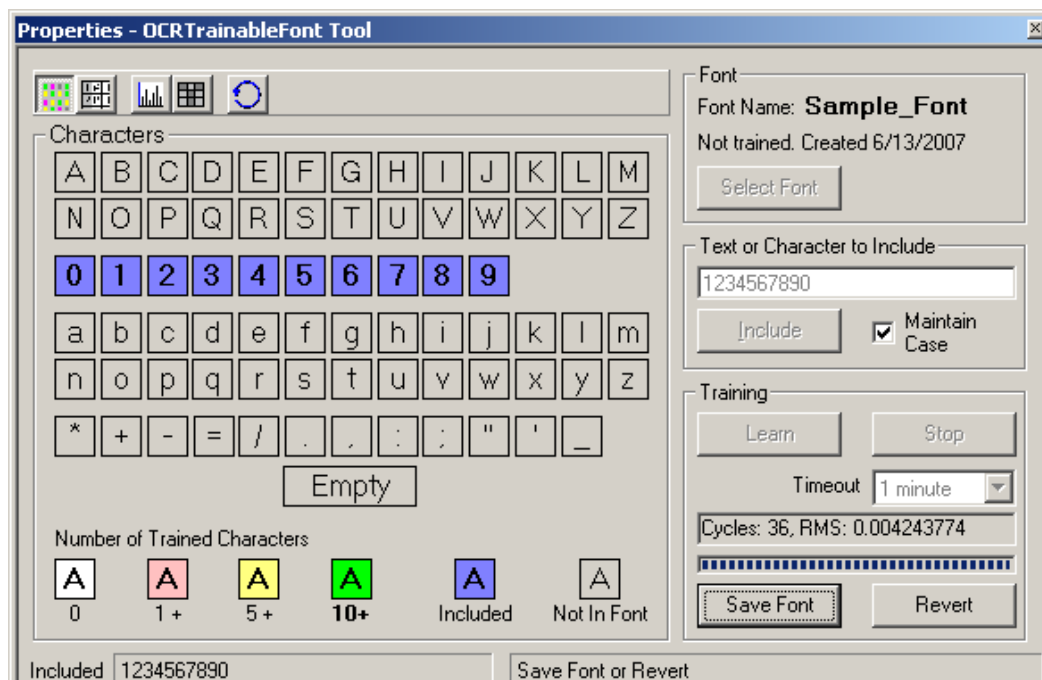
You may stop training, if desired.

You may train at any time as you include characters.

18. When training is complete, click **Save Font** or **Revert**, as shown in Figure 9–12.

The time required for training increases as more samples are added. The default timeout is 1 minute; select shorter or longer training periods from the Timeout drop-down list.

FIGURE 9–12. Click Save Font



Clicking **Save Font** saves the new training. The train date and time are displayed beneath the font name, as shown in Figure 9–13.

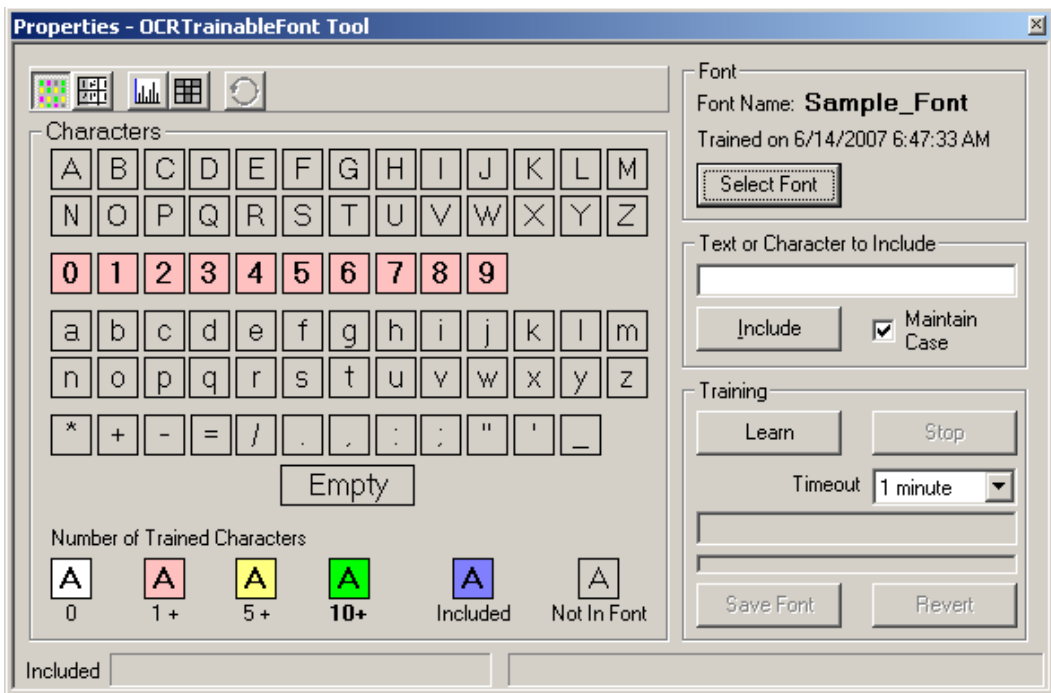
FIGURE 9–13. Font Name and Train Date and Time

Font Name: **Sample_Font**
Trained on 2/12/2003 11:48:57 PM

Clicking **Revert** abandons the training. If the font had been trained previously, it will revert to its previous trained state.

After training, characters are highlighted using red, yellow, and green to indicate the number of trained samples for each character.

FIGURE 9-14. After Training



When the mouse pointer rests over a character, a tool tip pops up indicating the number of trained samples for the character. In the image above, the tool tip shows that 6 samples have been trained for the character “A”.

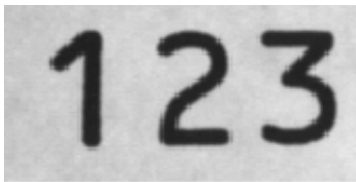
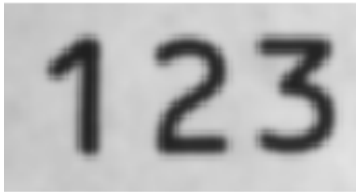
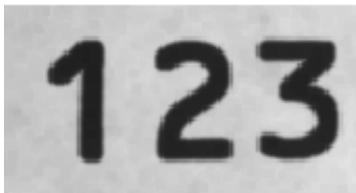
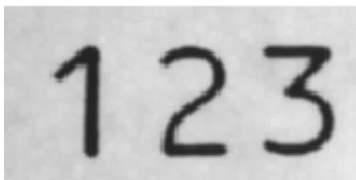
19. Defocus the lens slightly and repeat steps 11 - 18.

A font should be trained using samples that vary in quality. When fewer samples than desired are available, the set of samples can be re-used several times during training.

To re-use the character set after the first round of training, defocus the lens slightly to render the characters a bit fuzzy. If the lens is not easily accessible, image processing may be used to change the appearance of the characters.

This method simulates the variability expected in large sample sets of characters.

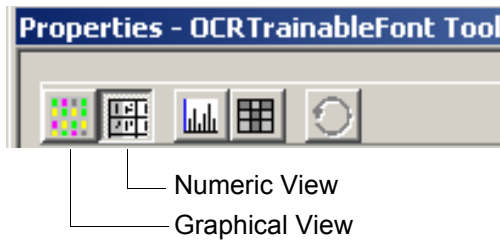
FIGURE 9–15. Defocusing an Image

**Sharp Focus** - Original Image**Defocused** - Simulate using MeanLP filter**Thicker Characters** - Simulate using
GrayMorph-Dilate Dark-1**Thinner Characters** - Simulate using
GrayMorph-Erode Dark-1

Another alternative to image processing is adjusting the Edge Energy datum for the OCRTF step. Changing this parameter is similar to changing focus to produce sharper or softer character edges.

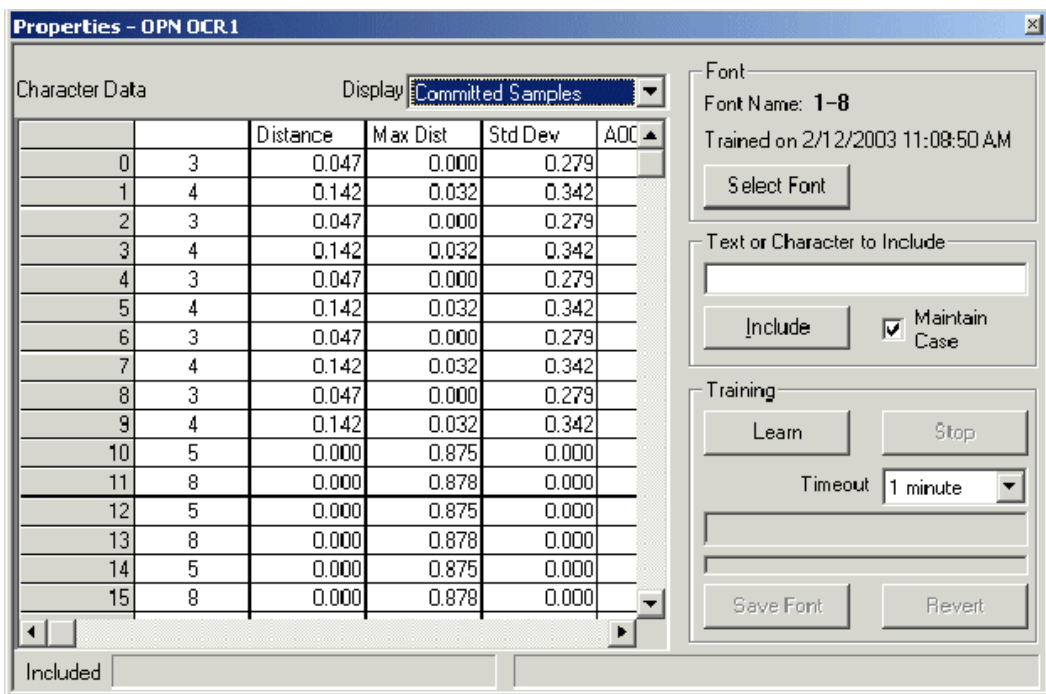
Numeric View

FIGURE 9-16. Numeric View Button



After you click **Numeric View**, the constants associated with each character sample are displayed in a grid, as shown in Figure 9-17.

FIGURE 9-17. Constants for Character Samples

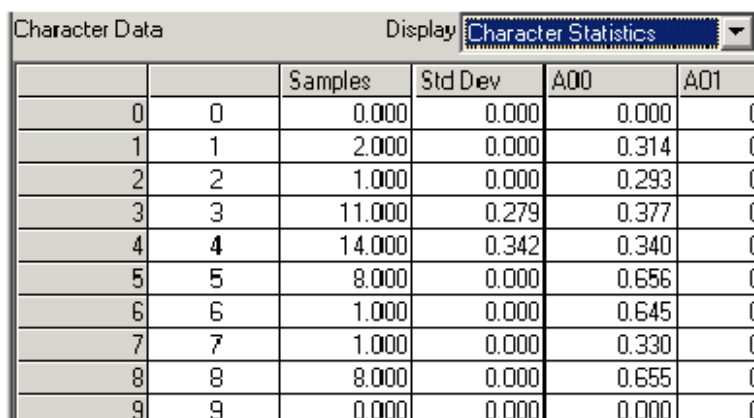


Committed Samples corresponds to included characters.

Individual samples may be deleted from the list by right clicking on the grid and choosing Delete.

Selecting **Character Statistics** displays the training statistics for all characters in the font, as shown in Figure 9–18.

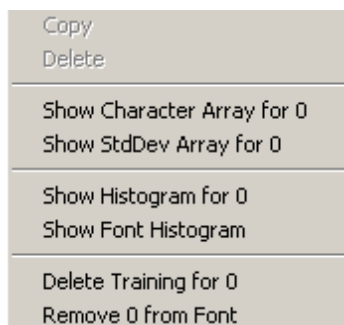
FIGURE 9–18. Training Statistics for All Characters



		Samples	Std Dev	A00	A01
0	0	0.000	0.000	0.000	(
1	1	2.000	0.000	0.314	(
2	2	1.000	0.000	0.293	(
3	3	11.000	0.279	0.377	(
4	4	14.000	0.342	0.340	(
5	5	8.000	0.000	0.656	(
6	6	1.000	0.000	0.645	(
7	7	1.000	0.000	0.330	(
8	8	8.000	0.000	0.655	(
9	9	0.000	0.000	0.000	(

Character-Related Features

FIGURE 9–19. Character Features Menu

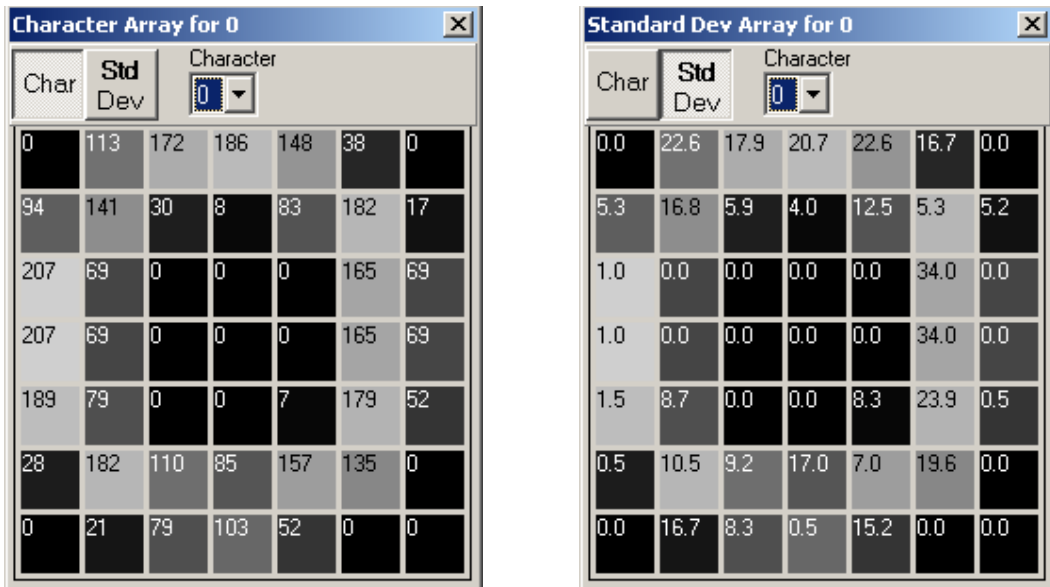


This menu is displayed when you right click on a character in the Graphical View.

Show Character Array and Show StdDev Array

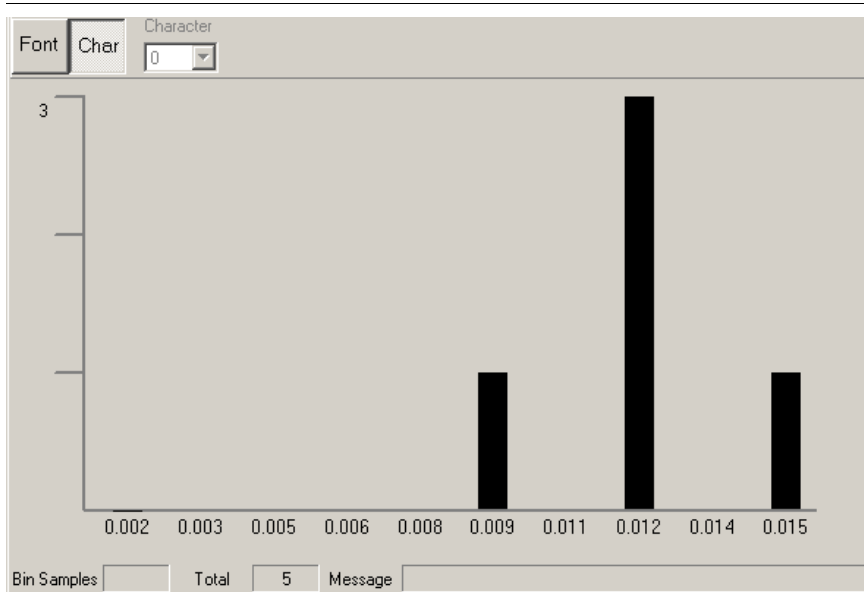
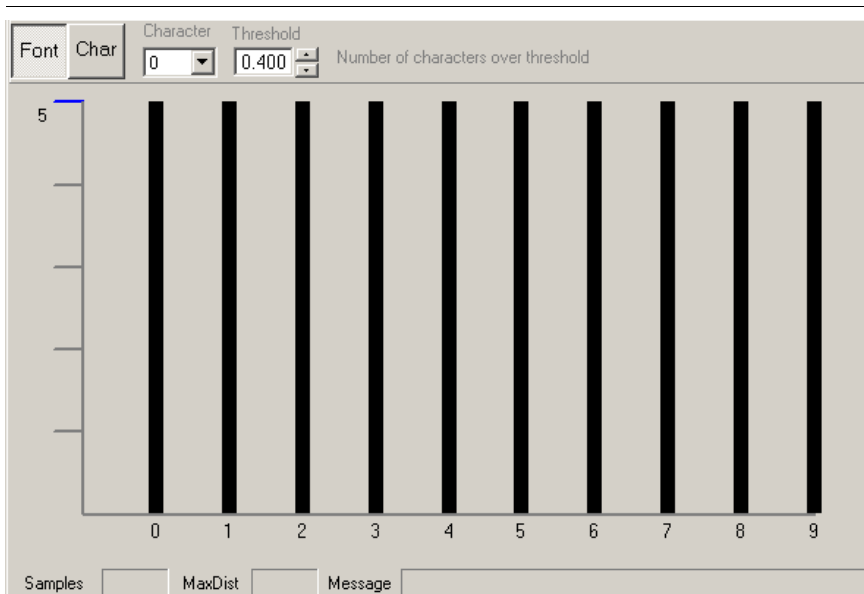
You can display constants for each character by selecting Show Character Array or Show StdDev Array, as shown in Figure 9–20. These displays are useful for low-level debugging and font testing only. These displays may not be helpful for an operator.

FIGURE 9–20. Results of Show Character Array and Show StdDev Array



Show Histogram and Show Font Histogram

You can display a histogram for a character (see Figure 9–21), or show font histogram information (see Figure 9–22).

FIGURE 9–21. Results of Show Histogram for a Character**FIGURE 9–22. Results of Show Font Histogram**

Delete Training and Remove from Font

The Delete Training feature removes all trained samples for a character from the font. This feature is useful if a particular character should be retrained by itself. You will be prompted regarding deleting all samples for a character.

Remove from Font deletes all training for a character and removes it completely from the font. You will be prompted regarding excluding the character from the font.

Training Tips

- The camera field of view should be configured so that the characters read by the OCRTrainableFont tool have a character width of 25 pixels or larger.
- Given an initial set of sample parts, divided the samples into two subsets. Use one subset for training and the other subset for reading tests.
- For OCRTrainableFont, the image background should be relatively free of noise.
- Be certain to train using samples of varying quality. The samples should be representative of the character print/mark quality expected during production runs of the printed characters.
- Train frequently. A font may be trained at any time.
- If the value of the RMS error increases significantly after a new training cycle, consider deleting recently added samples.
- Changes should be made to the trained font whenever a camera position is moved. If the change in camera position or lens focus is minor, it may be necessary to add no more than a few characters samples.
- Note that certain characters are more prone to be confused for other characters. Examples include B/8, G/6, and numeral 0/ letter O. During testing, confirm that these characters are distinguished from one another accurately.
- If the camera lens is inaccessible for defocusing, use software methods to simulate variations in character quality.
- Note that each OCRTF step can use multiple trained fonts.
- If training takes longer than a minute, try dividing a font into two or more smaller fonts. Click Select Font to display the Font Manager dialog, then click Duplicate to create a copy of the font. Remove characters from the font

copy to reduce the size of the character set. A font with characters A - Z can be split into fonts with A - M and N - Z. Make sure all required fonts are selected in the OCRTrainableFont Tool.

- Until Save Font is clicked, font changes may be abandoned. Close the OCRTF training dialog and click the “No” button when prompted to save changes.

Tips for Marking OCR Fonts

- Use either the numeral 0 or the letter O, but not both. If a human operator has trouble distinguishing characters quickly, the vision system will also have difficulty.
- Unlike machine-readable symbols such as Data Matrix, OCR does not have built-in error correction. Misreads are possible. For example, “813” may be reported as “B13”. If possible, include a checksum character in the printed text.

Results

- *Status* — Set to true after a successful execution of the step.
- *Number of Characters Found* — Total number of character objects found, whether or not they were successfully decoded.
- *Output String* — The set of characters found within the ROI placed in order from the top left most character and scanning to the right, then down. Characters that are found but not decoded will be represented by the character defined by **Set Unknown Characters To**.
- *Minimum Character Confidence* — The lowest match level of the characters successfully decoded.
- *Mean Character Confidence* — The mean match level of the characters successfully decoded. This excludes the confidence level of any characters that were excluded by the 'Minimum Confidence' parameter.
- *Maximum Character Confidence* — The highest match level of the characters successfully decoded.
- *OCRTF Character Results* — Contains a vector of the character confidence values for the successfully decoded characters only. The results are in the order of the characters in the output string skipping unknown characters.

IntelliText OCR

IntelliText Overview

The IntelliText OCR tool provides better character segmentation stability on noisy backgrounds and has a matchcode option that allows regular expressions to be used for matching text that has been read by the system.

IntelliText OCR can be used on the emulator, MicroHAWK MV, HAWK MV-4000, and GigE Cameras in combination with FrontRunner. Fonts can be trained in AutoVISION only, but font files trained in AutoVISION can easily be used in FrontRunner.

IntelliText OCR improves consistency and readability of a variety of substandard marks, such as inkjet marks or marks on noisy backgrounds. Significant improvements over standard OCR can be seen in the following areas:

- Improved invariance to background noise and uneven illumination;
- The ability to read rotated OCR text at any angle;
- The ability to handle fixed tilts and slants in characters;
- Characters can be scaled up or down to make them the optimal size for reading;
- The ability to create a variable match string using regular expressions;
- The ability to set character width and height to resolve segmentation issues.

IntelliText Binarization

Binarization Overview

OCR segmentation methods are implemented in the binary image domain. To handle uneven illumination conditions or ROIs with background variations, an advanced statistical method is used by default.

Note: Due to the wide variety of applications and image conditions, this automatic parameter selection for the binarization algorithm is not always optimal and can introduce instability.

For this reason, fixed binarization parameters can be used for optimal consistency. These parameters are “Binary Mean Factor” and “Binary Dynamic Range”.

Parameter Usage

When the “Binary Threshold Mode” is configured for “Fixed Template”, the “Binary Mean Factor” and the “Binary Dynamic Range” parameters are fixed and defined by the associated datum in the tool.

The recommended procedure for modifying the binarization parameters for optimization is first to modify the mean factor with a midrange value and then tune the dynamic range and mean factor as needed to optimize for the application.

For the “Binarization Mean Factor”, a high value will result in high background rejection on good contrast symbols, but a low value will be needed for lower contrast marks. The “Binary Dynamic Range” should be tuned only after a proper “Binarization Mean Factor” value has been selected.

IntelliText OCR User Interface in FrontRunner

In FrontRunner, the output binary image can be viewed as an output buffer by double-clicking the tool. The segmentation results are also drawn in the output buffer.



IntelliText OCR Tool Inputs

Selected Font

This datum is the currently selected font for character identification.

Character Polarity

This field selects the polarity type of the characters to detect in the ROI where “Dark Characters” refers to black characters on white background and “Light Characters” refers the white characters on a black background.

String Matching Method

This datum selects the method of string matching to apply to the “Output String” during tool execution.

Basic

When matching method is set to Basic, the tool will perform basic matching on the result.

Regular Expression (TRE)

When set to Regular Expression (TRE) matching the tool will test the result string against the TRE regular expression provided in the match string field. Regular expression syntax used can be found on the TRE website.

<http://laurikari.net/tre/documentation/regex-syntax/>

Examples:

Example expiration date string matching regex “EXP (JUN|FEB) [0-9]{2} [0-9]{4}”

“EXP JUN 12 1998“ - match

“EXP JUN 30 1999“- match

“EXP FEB 12 1998“ - match

“EXP JVN 12 1998“ - fail

“EXP JUN 12 98“ - fail

“EXP JUN 12 199B” - fail

“LXP JUN 12 1998” - fail

Add anchor character(s) for string start and end position(s)

“^EXP (JUN|FEB) [0-9]{2} [0-9]{4}\$”

Match String

This datum is the reference string used to match against the tool “Output String” when the tool is run. The match method used is defined by the “String Matching Method” datum and the output result is provided by the “Match Status” output datum.

Unknown Confidence Threshold

This datum defines the confidence threshold used to replace detected characters with the character defined by the “Unknown Character” datum.

Unknown Character

This datum defines the character used to replace detected characters that fail to meet the threshold defined in the “Unknown Confidence Threshold” datum.

Scaling Factor

This datum is used to change the sampling interval of the ROI. By reducing the scaling factor the ROI pixels will be sampled using a bilinear interpolation prior to execution of the tool. The IntelliText OCR tool is optimized for detecting characters that are in the range of 25-50 pixels in height. For text under high resolution, particularly high resolution dot print, this can improve detection by reducing scale. Because reducing the sampling introduces some loss of information, it is preferable to configure the optical setup accordingly. Scaling is provided for applications that may have other, unrelated requirements making high resolution necessary. Scaling may also provide some speed enhancements when using larger ROIs and relatively low character counts.

Minimum Character Width

This datum defines the minimum width of detected characters in pixels.

Maximum Character Width

This datum defines the maximum width of detected characters in pixels.

Minimum Character Height

This datum defines the minimum height of detected characters in pixels.

Maximum Character Height

This datum defines the maximum height of detected characters in pixels.

Binary Threshold Mode

This datum defines the threshold mode of operation used in detecting characters.

Auto

This mode uses threshold parameters that are automatically determined for detection. For optimal consistency under challenging environments use Fixed Template with application-determined parameters.

Fixed Template

This mode uses fixed binarization parameters defined by the “Binary Mean Factor” datum and the “Binary Dynamic Range” datum for processing. Refer to the [IntelliText Binarization](#) section for further detail.

Match Test String

This datum is a test datum that is useful in evaluating match strings. FrontRunner will update the “Match Test String Result” datum when this datum is modified. This result is the match result using the current “String Matching Method” and comparing the “Match Test String” with the configured “Match String”.

Process Default Fonts

This enables / disables processing of default fonts that may exist in the font file selected. Detected characters are run against any default fonts provided to find the best character match. This can significantly impact speed and error rates. It is recommended that users train fonts for their application and disable this during application deployment to achieve best performance.

Binarization Mean Factor

See the [IntelliText Binarization](#) section for details about this parameter.

Binarization Dynamic Range

See the [IntelliText Binarization](#) section for details about this parameter.

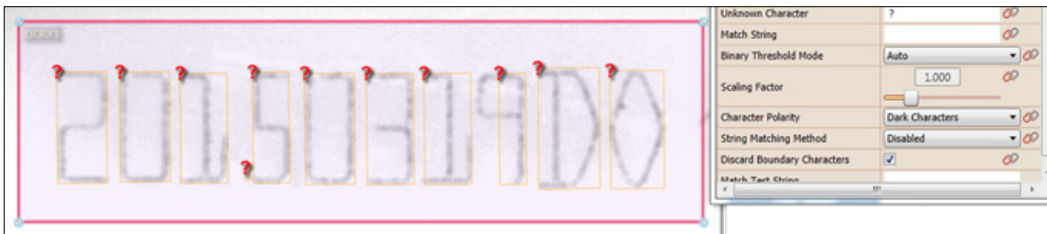
Discard Boundary Characters

This mode eliminates any binary objects that are touching the boundary ROI. This is needed in certain applications where items may extend into the ROI. This is true for UPC symbols that have bars extending into the human readable area.

Optimization Examples

Binarization Usage

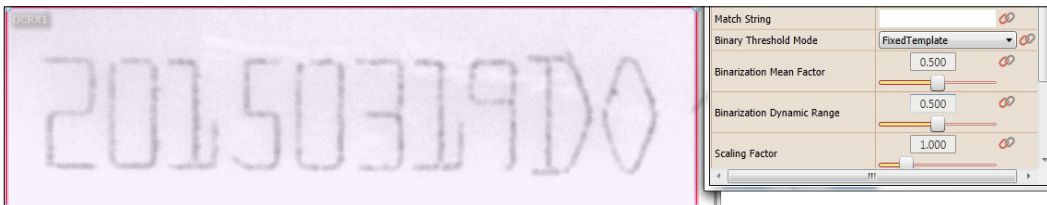
Consider the following low contrast image image. In this example, binarization parameters will be adjusted to optimize performance for this type of exposed mark. It is worth noting that in this example the characters are quite tall – around 80 pixels – but because they are generally continuous, a reasonable segmentation is achieved. This is an ideal binarization example because it is easy to visualize the effects. Reducing the scaling will also work well on this mark as shown in the “Scaling Usage” example. For optimal performance, the mark should be scaled to around 30 pixels in height and be optimized as shown below.



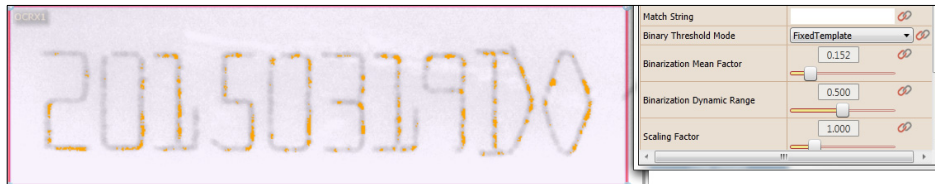
The first step is to change Threshold Mode to Fixed Threshold. This will show the binary buffer used for processing as an overlay on the ROI. It will use the current fixed template parameter settings. If you immediately click back to “Auto” you will see the result of the current Auto settings.



The above segmentation is close, but the characters are a bit eroded. Improvements can be made by switching to Fixed Template and tuning the binarization parameters. Initially, the default parameter values (0.5, 0.5) provide a completely blank binarization image.



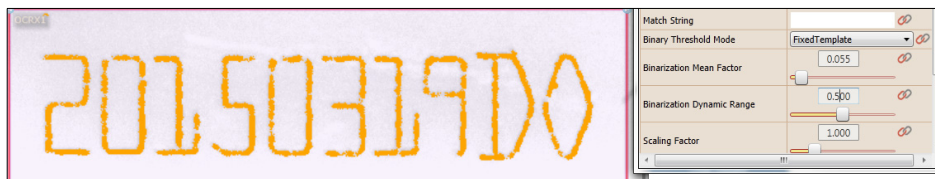
The next step is to start reducing the “Binarization Mean Factor”. Around a value of 0.2 to 0.15 characters will start to appear on this particular mark.



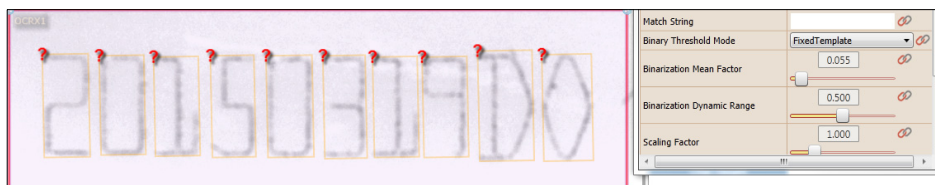
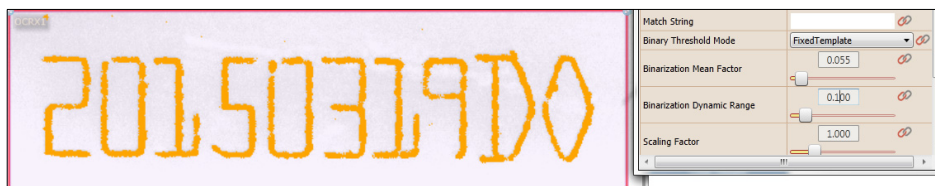
By continuing to reduce the Binarization Mean Factor, noise will start showing up around 0 on this particular sample.



The next step is to raise the threshold again until most of the background noise and objects are eliminated. Note that the characters might start to erode a bit again, but are still well-formed.



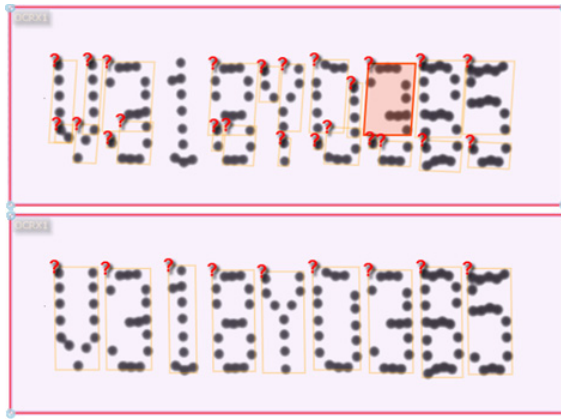
The “Binary Dynamic Range” is then lowered, having the effect of filling in the countour around the character. The result is a well-segmented mark.



Scaling Decrease Usage

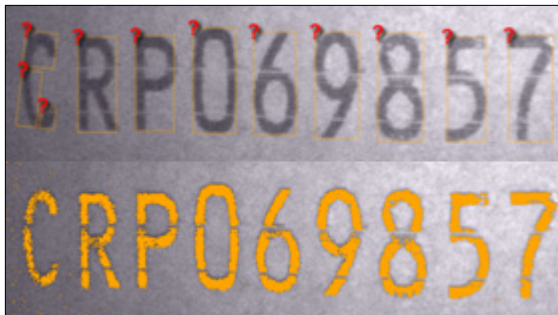
Example 1

Consider the following large dot mark. The mark has good contrast and does not require binarization optimization, but it does fail segmentation. This particular mark is 130 pixels high, causing the text segmentation to identify entire characters improperly. By reducing the scaling to 0.5, proper segmentation becomes possible. The 0.5 scaling reduces the height to 65 pixels and segmentation is successful, although ideally this symbol would be captured at a lower resolution for optimal performance.



Example 2

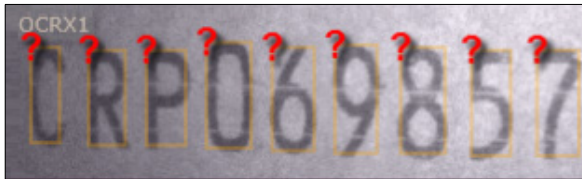
Consider the following large mark with image noise and a wide dynamic change across the ROI. The automatic binarization selection works well, but the low signal-to-noise ratio on the left side of the mark causes the 'C' to fail. This could be optimized by fixing the binarization.



As this mark is approximately 80 pixels high, it is rather large for optimum text segmentation. Because the mark is so large, the algorithm has detected segments from the 'C' on different lines of characters.

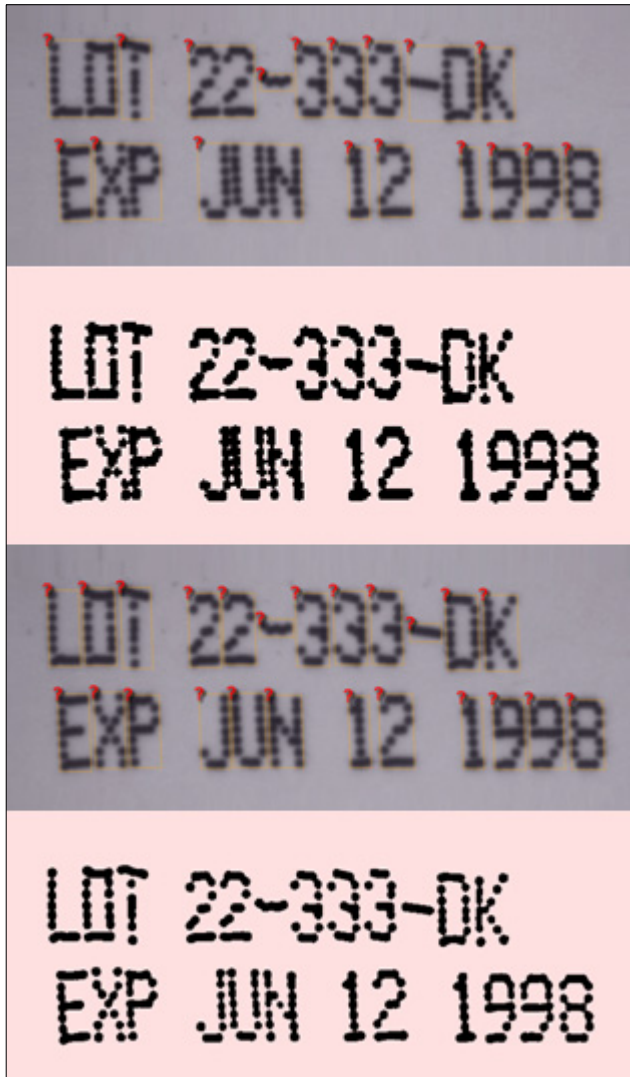


By modification of the scaling, the segmentation problem is resolved. Note that this low contrast mark may experience some instability from image-to-image with Auto binarization and would likely benefit from a Fixed Template setting.



Scaling Increase Usage

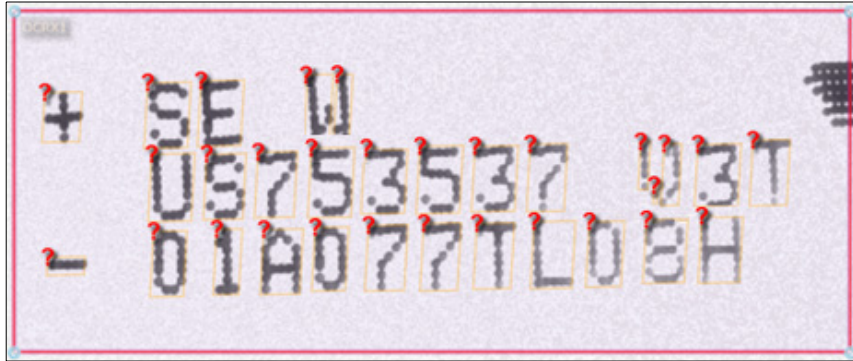
Consider the following dot mark with close and partially touching characters. The mark is at reasonable resolution but the close characters are creating segmentation problems. Character thresholds can be modified to separate them, but this can be challenging, particularly with the JUN segment. Setting the scaling to 2.0 in combination with over-exposing the image creates a good separation that binarizes well and subsequently segments properly and consistently.



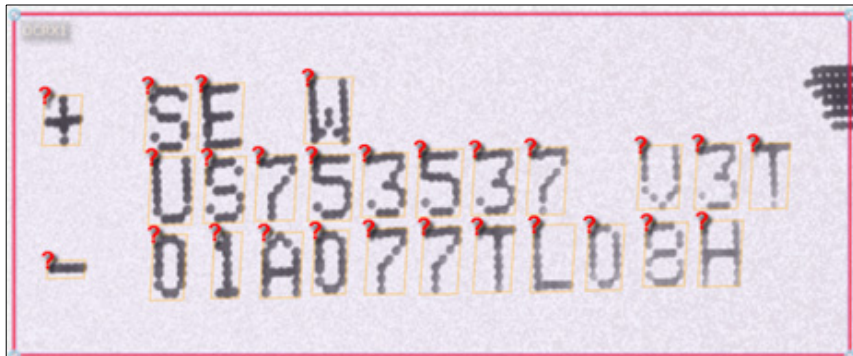
Character Threshold Usage

Example 1

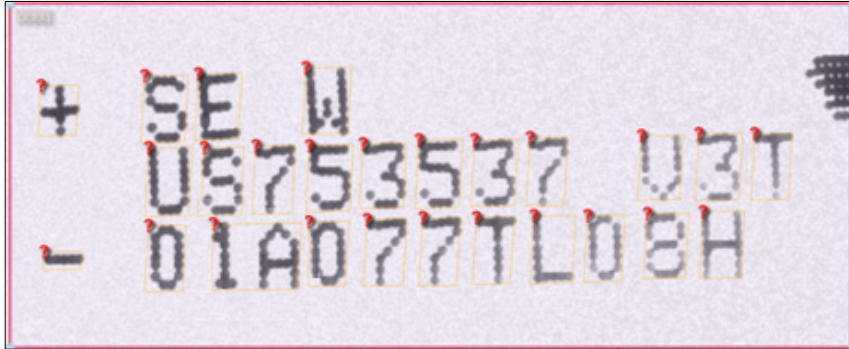
Consider the following segmentation problem. The mark mostly segments well, but splits a couple characters up because of poor horizontal connection between the dots.



In this example, the maximum character width is about 20 pixels. The segmentation problem can be resolved by increasing the minimum width from 1 to 15 pixels.

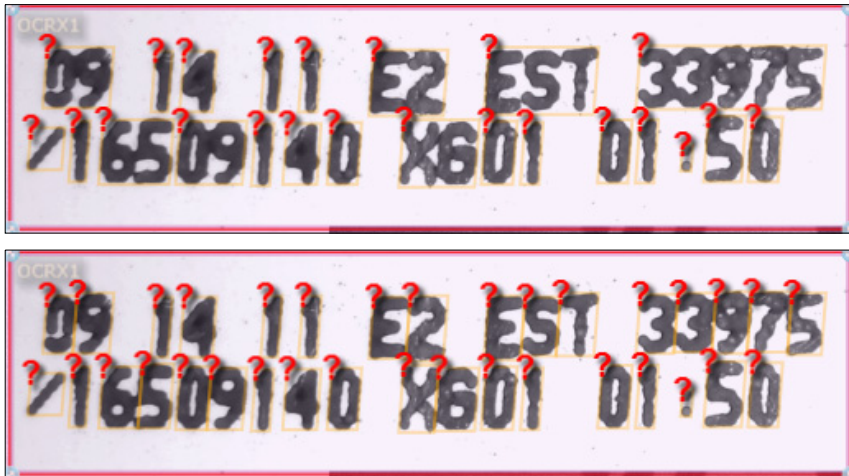


If threshold is set too high, there is a risk of combining characters as shown below.



Example 2

In this example, touching characters can be segmented by properly, reducing the maximum width to a value closely matching the nominal width of the wide characters.



IntelliText OCR Tool Outputs

Status

This output datum is true if the tool has run properly.

Output String

The output datum represents the output string of the detected characters. Characters below the “Unknown Confidence Threshold” are replaced by the “Unknown Character”.

Number of Characters Found

The output datum represents the number of characters detected.

Match Status

This is the result of the matching of the “Output String” against the “Match String” when the tool is run using the methodology defined by “String Matching Method”.

Minimum Character Confidence

The minimum confidence seen in the characters detected for the font selected.

Maximum Character Confidence

The maximum confidence seen in the characters detected for the font selected.

Mean Character Confidence

The mean confidence of all characters detected for the font selected.

Match String Test Result

This datum is a test datum that is useful in evaluating match strings. FrontRunner will update this datum result when the “Match Test String” datum is modified. This result is the match result using the current “String Matching Method” and comparing the “Match Test String” against the “Match String”.

Symbol Quality Verification Tool

Other Steps Used

None.

Theory of Operation

Verification in the context of Visionscape is the process of grading the quality of a 1D or 2D Auto ID symbol on an interval scale according to an agreed or specified standard. The verification/grading process is a measure of the quality of the mark from the point of view of readability. It should not be confused with the process for checking the content of the Auto ID symbol.

Auto ID symbol verification may be employed to serve one or more of a number of overlapping purposes. The overall goal is to ensure that a symbol can be reliably recognized and decoding at the point of reading. The requirement for verification may arise from requirements of standards and regulations (for example with respect to pharmaceutical labels); from commercial standards specified by trading partners (for example when a retailer desires that incoming goods can be scanned in a receiving area); or from the requirement to maintain traceability within a manufacturing process or product life cycle.

In all cases it is good practice to perform symbol quality verification as close to or as soon after the marking process as possible so that out of tolerance or bad symbols can be detected quickly and the marking process adjusted before a significant number of unreadable or lower quality marks are created.

The Verification Tool provides a graded result (0.0 – 4.0). It can therefore be used to indicate trends and apply a safety margin to a marking process. This is not true of go/no-go “verification” performed using a reading device which cannot detect symbols that are close to the edge of un-readability. It is an axiom that symbols never improve after marking and that a symbol that is just readable at the point of marking is only one scratch or smear away from illegibility.

Choice of Standard

The Verification Tool supports a range of formal standards and also allows the user to create custom standards by selecting individual measurements from the formal standards and applying custom thresholds and criteria.

Formal standards are appropriate when verification is applied to confirm that symbols and marks meet the requirements regulatory standards or the requirements of trading partners. The customized measurement capability may be appropriate for supporting “in-house” traceability requirements and monitoring particular marking processes.

The applicability of the formal standards is as described below.

ISO/IEC 15415

This standard is defined as a “bar code print quality test specification” for two dimensional symbols. The standard states that its purpose is to “provide symbol producers and their trading partners with universally standardized means for communicating about the quality of multi-row bar code and two-dimensional matrix symbols”. The Verification Tool implements this specification for the Data Matrix symbology. ISO 15415 is designed for and is typically applied for high contrast marks such as those created by conventional printing purposes on paper labels or similarly prepared surfaces. The manner in which low contrast values and substrate variation is penalized in the grading process means that this standard is not usually appropriate for direct part marks (DPM).

AIM DPM/ISO 29158

AIM DPM / ISO 29158 was developed to address the need to measure the quality of direct parts marks such as those produced directly on component surfaces by such means as dot peen or laser marking. The measurements process and the grading criteria used by the AIM DPM / ISO 29158 standard take into account the expected lower contrast and substrate variability.

ISO/IEC 15416

The ISO 15416 standard provides an agreed methodology for grading and predicting the readability of 1D symbols. It is the equivalent of ISO 15415 in that it assumes that the symbol will be of high contrast and that the substrate will be relatively uniform.

Setup

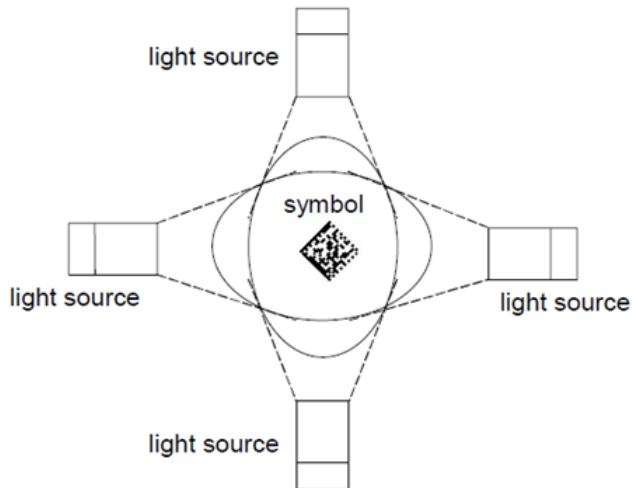
The standards each provide a recommended or reference geometric, optical, and lighting configuration. The user is encouraged to consult the latest revision of the standard to ensure compatibility. In broad terms the goals of the standards are to specify measurement configurations that are consistent with appropriate or likely reading scenarios. The recommended geometries can be summarized as follows.

All Standards

Camera optical axis positioned perpendicular to the inspection area.

ISO/IEC 15415

The specified reference lighting geometry (illustrated) uses four light sources at 45 degrees to produce uniform illumination across the inspection area. The standard allows alternative geometries of 30 degree and diffuse on-axis lighting where appropriate.



AIM DPM/ISO 29158

Lighting options: On axis (90), Diffuse on axis (D), Low angle 30 degrees – Four directions (30Q), Two directions (30T), Single direction (30S).

AIM DPM/ISO 29158 requires symbols to be aligned within ± 5 degrees of the sensor rows and columns.

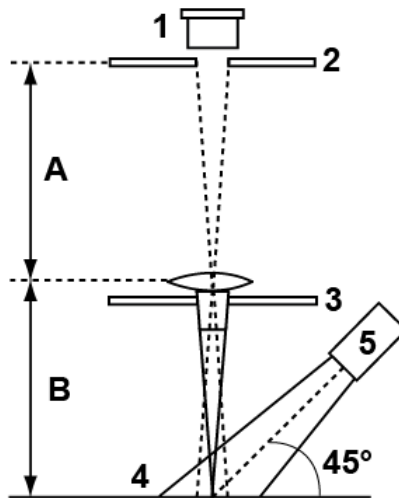
ISO/IEC 15416 (1D Symbols)

Below is the required setup for 1D verification as detailed in the ISO 15416 specification.

The correct ISO 15416 symbol verification setup includes an illumination source that is uniform across the symbol area at a 45 degree angle to the surface, and in a plane containing the illumination source that is perpendicular to the surface and parallel to the bars of the 1D symbol. The configuration also includes a camera whose axis is perpendicular to the symbol surface.

The light reflected from a circular sample area of the symbol surface is collected in a cone formation with a vertex angle of 15 degrees, centered on the perpendicular to the surface, through a circular measuring aperture with a diameter of 1:1 magnification which is equivalent to that of the sample area (the area containing the 1D symbol).

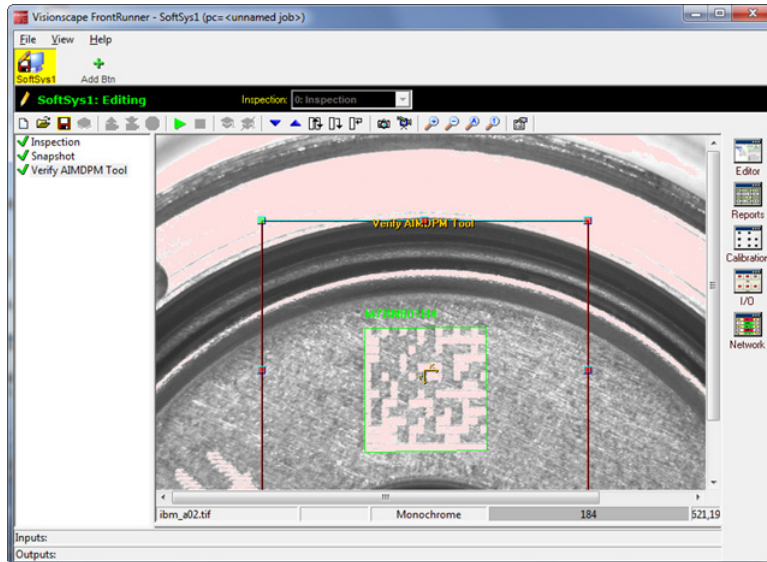
The angles of the configuration described above minimize specular reflection and maximize diffuse reflection from the symbol. This configuration is intended to provide a reference for measurement consistency.



- 1 = Image Sensor
- 2 = Aperture at 1:1 Magnification (measurement A = measurement B)
- 3 = Light Baffle
- 4 = Symbol Surface
- 5 = Illumination Source

Process and Settings

Visionscape implements the Verification Tool as three separate steps that can be selected from the Analysis Tools tab in the Inset Step dialog.

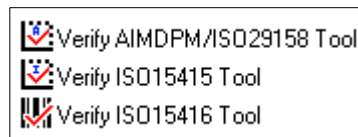


The following steps and settings are common to each of the three verification methods.

In each case the sample/target must be placed under the camera at an optimum focus obtained using the Image tab or the appropriate elements of the Setup dialog in Visionscape. The symbol should be imaged so as to provide the required resolution and quiet zones. Once this is done the Verification Tool should be specified and the region of interest (ROI) adjusted to encompass the symbol and its quiet zone. The standard to be applied is selected by inserting the appropriate verification tool in Visionscape.



**Provide the Necessary
Resolution and Quiet Zone**



Available Verification Tools













All three standards require that the illumination wavelength and geometry be included in the verification report. These quantities are entered in the Verification Settings portion of the parameter page.

 	Dimension Units	mil
 	Auto Aperture Size (%)	50
 	Aperture Size (mil)	6
 	Wavelength (nm)	640
 	Light Angle	90

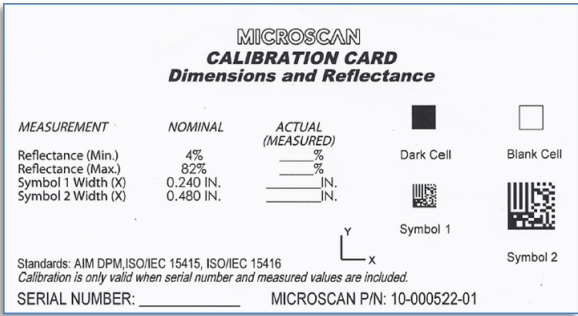
This section also includes the aperture specification. In the context of verification the aperture setting is the size of the averaging circle that used to calculate pixel grayscale (reflectance) values by the respective verification algorithms. The “auto” values implement an aperture relative to the size of the symbol being inspected. These values may be overridden by specifying a dimension for the spot size.

The calibration dialog is used to enter the reflectance values and dimensions of the reflectance/dimensional calibration target. These are required by the calibration algorithm to generate a relationship between grayscale values and reflectance values and for dimensional calibration that will relate pixel dimension to real world units.

The dialog also includes a maximum exposure (duration) value that represents the maximum exposure time that will be used by the calibration routine. If the calibration process cannot meet the dynamic range requirements for verification without exceeding this threshold it will fail. If the exposure cannot be increased due to the application environment more illumination must be applied. This value should be carefully specified for on-line grading of moving parts.

 	Card Calibration	<click to execute>
 	- Calibration Card Rmin (%)	4,000
 	- Calibration Card Rmax (%)	82,000
 	- Calibration Symbol 1 Dimension (in)	0,240
 	- Calibration Symbol 2 Dimension (in)	0,480
 	- Maximum Exposure (us)	32000

The calibration process is performed by presenting one of the two symbols on the calibration card and pressing the “calibrate” button. During the process the system will adjust the camera settings to produce and optimum range of grey scales and identify the symbol on the calibration card and perform a dimensional calibration. The calibration will be saved with the job but will be reset to un-calibrated if the exposure or gain settings are changed manually by the user.

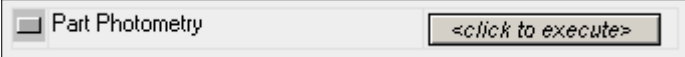


Unique Settings

The ISO 15416 Verification Settings dialog contains a Signal Compensation option that can be used to compensate for uneven illumination. In some application environments where compliant verification is not required this can provide a grading method without needing the highly uniform illumination the standard requires. Selection of this option will mean that the result obtained is not standard compliant.



The calibration process for AIM DPM/ISO 29158 is a two step process. The first step uses the calibration card to obtain a reference for the dimension and reflectance and should be done once the application environment is setup or changed. This environment includes any changes focal distance or illumination. After the calibration step using the calibration card a second step should be performed using the actual part as the reference. AIM DPM/ISO 29158 allows for a variety of parts and substrates to be inspected beyond common printed materials. This may require the camera to be adjusted to allow the proper image for grading. This second step will adjust the camera exposure to take account of the reflectivity of the part and should be repeated every time a part with substantially different reflectivity is introduced.



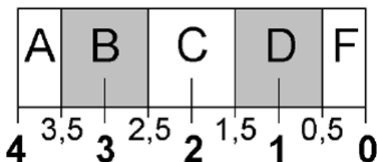
Verification Reports

The two 2D symbol verification tools produce similar reports. The most significant differences are in the grade assignments of the values of each measurement and the details of calculation of each quantity.

ISO 15415 Verification Report					
Status	Fair				
	Grade	Value	Units		
Overall Grade	C			Grade Reported	2.0/3.2 pixels/640/90
Symbol Contrast	C	52	%	Calibrated	FALSE
Modulation	A			Cal Symbol 1 Size	0.24
Reflectance Margin	A			Cal Symbol 2 Size	0.48
Fixed Pattern Damage	A			Cal RMin(%)	4.00
Axial NonUniformity	A	3		Cal RMax(%)	82.00
Grid NonUniformity	A	7		Damage %	0
Unused Error Correction	A	100			
Print Growth	A	6	X	Rows	10
		0	Y	Cols	10

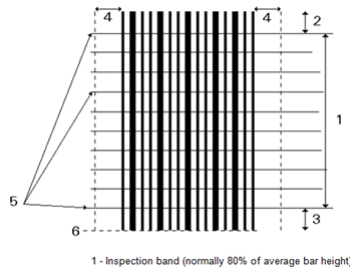
AIMDPM Verification Report					
Status	Good				
	Grade	Value	Units		
Overall Grade	A			Grade Reported	DPM4.0/19/640/90
Cell Contrast	A	72	%	Calibrated	TRUE
Cell Modulation	A			Cal Symbol 1 Size	0.24
Reflectance Margin	A			Cal Symbol 2 Size	0.48
Minimum Reflectance	A	82			
Fixed Pattern Damage	A				
Axial NonUniformity	A	1		Damage %	0
Grid NonUniformity	A	9			
Unused Error Correction	A	100		Rows	22
Print Growth	B	-18	X	Cols	22
		-10	Y		

Operating normally without custom verification enabled the overall grade is the worst of the individual measurements and is formally reported as a number grade using the following range table.



Reporting for 1D symbols is similar except that the report dialog contains information for each of the 10 individual scans that are part of the measurement process. In this case the overall value of each scan is the worst measurement within that scan and the overall grade is the average of the ten scans. The same translation between letter and number grades is applicable.

Verification Measurements



Location of the 10 Scan Profiles Specified by the ISO Standard

The process for creation of the individual measurements is complex and the reader is referred to the current standards for a formal definition. The short form of the common parts of the process is as follows.

As a first stage the symbol is located and the threshold between on and off cells (or lines and spaces) is established. The symbol is then averaged using an aperture and then binarized using the threshold. The reference decode algorithm that is specified in the standard is then applied. In the case of the 2D standards this process may be applied recursively until a successful decode is achieved. In all cases if a successful decode is not achieved at this stage the Decode measurement gets an F (fail) and the verification process is terminated. Since the reference decode algorithm is relatively primitive the Verification Tool may report a failed decode when the Symbolology Tool (with an X mode decoder) reports success.

The outline of the other measurements is as follows.

2D Symbols

- **Symbol Contrast (AIM DPM/ISO 29158)** — The difference in percent of the center of the distribution of the light cells of the Data Matrix versus the center of the distribution of the dark cells.
- **Symbol Contrast (ISO 15415)** — The difference in percent of the maximum and minimum reflectivity in the inspection area.
- **Modulation** — A measurement of the uniformity of the contrast of the dark areas and the light areas of the Data Matrix
- **Reflectance Margin** — Degree to which the cells are correctly distinguishable as black or white in comparison to the threshold.

- **Fixed Pattern Damage** — Measurement of non-uniformity in the quiet zone and in the locator and clock tracks
- **Axial Non-Uniformity** — A measurement of the difference between the printed size of the rows and columns in a matrix.
- **Grid Non-Uniformity** — This measurement is a delta of the difference of the measured grid in relation to the ideal grid formed from the four sides of the Data Matrix.
- **Unused Error Correction** — The amount of error correction remaining after applying the reed Solomon error correcting algorithm and successfully decoding the symbol expressed as a percentage of the error correction contained within the symbol.

1D Symbols

- **Edge Determination** — Percentage value of minimum edge contrast. Edge Contrast is the difference between the bar reflectance and space reflectance of two adjacent elements.
- **Symbol Contrast** — Is the difference between the highest and the lowest reflectance values in a scan reflectance profile
- **Min Reflectance** — Percentage value of reflectance of darkest bar.
- **Min Edge Contrast** — Percentage value of minimum edge contrast. Edge Contrast is the difference between the bar reflectance and space reflectance of two adjacent elements.
- **Modulation** — The ratio of the minimum edge contrast to symbol contrast.
- **Defects** — Irregularities found within elements and the quiet zones measured in terms of element reflectance non-uniformity
- **Decodability** — The proportion of the available margin before improperly decoding characters within the symbol. This measurement is based off the individual bar and space elements within a character and their deviations from thresholds used to identify characters.
- **Quiet Zone** — Fails if there is a violation in the quiet zone region before the leading or trailing bar. The required quiet zones are symbology dependent and can be found in the symbology specifications.

Custom Verification

	Custom Verification	<input checked="" type="checkbox"/>
	Set Selected Parameters with Same Grade	<input checked="" type="checkbox"/>
	Good Grade	3,000
	Fair Grade	2,000
	-- Edge Determination	<input checked="" type="checkbox"/>
	-- Decode	<input checked="" type="checkbox"/>
	-- Symbol Contrast	<input checked="" type="checkbox"/>
	-- Minimum Reflectance	<input checked="" type="checkbox"/>
	-- Minimum Edge Contrast	<input checked="" type="checkbox"/>
	-- Modulation	<input checked="" type="checkbox"/>
	-- Defects	<input checked="" type="checkbox"/>
	-- Decodability	<input checked="" type="checkbox"/>
	-- Quiet Zone	<input checked="" type="checkbox"/>

Visionscape allows the user to override or modify the measurement selection and thresholds in the standards based verification process by enabling the custom verification option.

Once this is selected the user can deselect measurements which will not then be part of the overall grading process.

In addition the user may move the boundary between Good/Fair and Fair/Poor levels to cover the required grade range. In the example shown, which might be appropriate for laser DPM marking on a moving part with a low contrast background, cell contrast and axial non uniformity have been de-selected since these measurements are expected to have low grades. The remaining measurements are those which are appropriate for monitoring the laser marking process. In addition the good/fair/poor boundaries have been adjusted to reflect tolerance of fixed pattern damage (due to an irregular substrate) and print growth.

Results

A comprehensive set of verification results are available for transmission using the output step.

The available results for the two 2D tools (ISO 15415 and AIM DPM/ISO 29158) which are listed below include the overall status, final grade (i.e grade expressed as a number), the grade report string (i.e the formal result grade number/spot size/wavelength/lighting configuration), the individual measurement grades, where appropriate the corresponding numeric value (i.e contrast value) and symbol dimensions.








































INT Error Code	0
Status	False
Report Grade	0.0/0.0 pixels/640/90
INT Grade Value	0
Verification Standard	ISO/IEC 15415
Symbology Type	Data Matrix ECC200
Decoded Text	N/A
Calibration Status	Uncalibrated
INT Decode Grade	0
INT Contrast Grade	0
INT Modulation Grade	0
INT Reflectance Margin Grade	0
INT Fixed Pattern Damage Grade	0
INT Axial Nonuniformity Grade	0
INT Grid Nonuniformity Grade	0
INT Unused Error Correction Grade	0
INT Symbol Contrast Value (%)	0
INT Axial Nonuniformity Value (x100)	0
INT Grid Nonuniformity Value (x100)	0
INT Unused Error Correction Value (x100)	0
DB _{0.80} Contrast Uniformity (x100)	0,000
DB _{0.80} Rmin (%)	0,000
DB _{0.80} Rmax (%)	0,000
INT Global Threshold	0
Symbol Polarity	Light on Dark
Symbol Size	0x0
DB _{0.80} Cell Size (pixel)	0,000
INT Print Growth X (%)	0
INT Print Growth Y (%)	0
P.T Symbol Corner 1	(853,000,222,000)
P.T Symbol Corner 2	(853,000,222,000)
P.T Symbol Corner 3	(853,000,222,000)
P.T Symbol Corner 4	(853,000,222,000)

ISO 15415

INT Error Code	0
Status	False
Report Grade	DPM0.0/0.0 pixels/640/90
INT Grade Value	0
Verification Standard	AIM DPM/ISO 29158
Symbology Type	Data Matrix ECC200
Decoded Text	N/A
Calibration Status	Uncalibrated
INT Decode Grade	0
INT Cell Contrast Grade	0
INT Cell Modulation Grade	0
INT Reflectance Margin Grade	0
INT Fixed Pattern Damage Grade	0
INT Axial Nonuniformity Grade	0
INT Grid Nonuniformity Grade	0
INT Unused Error Correction Grade	0
INT Cell Contrast Value (x100)	0
INT Axial Nonuniformity Value (x100)	0
INT Grid Nonuniformity Value (x100)	0
INT Unused Error Correction Value (x100)	0
DB _{0.80} Contrast Uniformity (x100)	0,000
Symbol Polarity	Light on Dark
Symbol Size	0x0
INT Mean Light	0
DB _{0.80} Cell Size (pixel)	0,000
INT Print Growth X (%)	0
INT Print Growth Y (%)	0
P.T Symbol Corner 1	(1107,000,43,000)
P.T Symbol Corner 2	(1107,000,43,000)
P.T Symbol Corner 3	(1107,000,43,000)
P.T Symbol Corner 4	(1107,000,43,000)

AIM DPM / ISO 29158

The 1D verification tool (ISO 15416) makes similar results available with the addition that results are available for each of the 10 scan profiles generated by the grading process.

 INT	Error Code	0
 Status	Status	True
 "A"	Report Grade	4.0/6.0 pixels/640/90
 DBL 0.00	Overall Grade (Average Final Grade)	4,000
 Good Status	Good Status	True
 Fair Status	Fair Status	False
 Poor Status	Poor Status	False
 "A"	Verification Standard	ISO/IEC 15416
 "A"	Symbology Type	Code 128
 "A"	Decoded Text	39123439
 "A"	Calibration Status	Uncalibrated
 DBL 0.00	Average Edge Determination Grade	4,000
 DBL 0.00	Average Decode Grade	4,000
 DBL 0.00	Average Contrast Grade	4,000
 DBL 0.00	Average Minimum Reflectance Grade	4,000
 DBL 0.00	Average Minimum Edge Contrast Grade	4,000
 DBL 0.00	Average Modulation Grade	4,000
 DBL 0.00	Average Defects Grade	4,000
 DBL 0.00	Average Decodability Grade	4,000
 DBL 0.00	Average Quiet Zone Grade	4,000
 INT	Final Grade - Scan Line 1	4
 INT	(1)Edge Determination Grade	4
 INT	(1)Decode Grade	4
 INT	(1)Contrast Grade	4
 INT	(1)Minimum Reflectance Grade	4
 INT	(1)Minimum Edge Contrast Grade	4
 INT	(1)Modulation Grade	4
 INT	(1)Defects Grade	4
 INT	(1)Decodability Grade	4
 INT	(1)Quiet Zone Grade	4
 INT	(1)Symbol Contrast Value (%)	100
 INT	(1)Rmin (%)	0
 INT	(1)Rmax (%)	100
 INT	(1)Minimum Edge Contrast Value (%)	100
 INT	(1)Modulation Value (x100)	100
 INT	(1)Defects Value (x100)	0
 INT	(1)Decodability Value (x100)	99
 DBL 0.00	(1)Quiet Zone Start Value (X-dim)	10,000
 DBL 0.00	(1)Quiet Zone Stop Value (X-dim)	10,000

Further Reading

A very clear description of the ISO 15416 grading process is available in “The Layman's Guide to ANSI, CEN and ISO/IEC Linear Bar Code Print Quality Documents” that is readily available from AIM at http://www.aimglobal.org/?page=bc_resources.

This chapter provides Optical Character Verification tool details.

OCV Inspection

The three OCV Tools and their supporting steps inspect codes such as component ID and Date/Lot. The print may be either pre-print or On-line. Individual printed features are referred to as symbols. The OCV inspection methods inspect the quality of the individual printed symbols.

Quality checks include:

- **Contrast** — A contrast value is calculated for the symbol and compared to a user-selectable contrast minimum. The symbol fails if the calculated value is less than the minimum. Contrast refers to the difference between symbol and background.
- **Sharpness** — A sharpness value is calculated for the symbol and compared to a user-selectable sharpness minimum. The symbol fails if the calculated value is less than the minimum. Sharpness is an indication of symbol border definition or symbol crispness.
- **Breaks** — The current symbol data is compared to the trained symbol data. If a break appears in the current data that was not in the trained data, the symbol fails.
- **Initial Residue** — Initial residue of the symbol is basically a count of those pixels that differ between the trained template and the current image. A binary residue template is created that contains On pixels only where a difference occurs between the current image data and the trained symbol template. The sum of the On pixels in this residue template is the initial residue value. If this value is greater than the user-selectable maximum, the symbol fails. Refer to Figure 10–1.

FIGURE 10–1. Initial Residue Examples



Template



Symbol



Initial Residue

- **Final Residue Total Count** — After performing a set of morphological operations on the residue image, the final residue is calculated again as the number of On pixels in the residue image. If this value is greater than the user-selectable maximum, the symbol fails. Increasing final residue % accepts symbols of lower quality.
- **Final Residue Largest Blob** — After performing a set of morphological operations on the residue image, the largest blob is found in the residue image. If the area of this blob is greater than the user-selectable maximum, the symbol fails.
- **For the font based tools, OCFontTool and OCVRuntimeTool, Runtime ID Checking** — When a symbol is trained and added to an OCFont, it is compared to other symbols already in the OCFont.

Note: OCFont boxes must be the same size in order to utilize Runtime ID tests.

If the two symbols being compared are found to be similar, tests are set up that verify that the correct symbol is present at runtime. If it cannot be determined that the correct symbol is present at runtime, the symbol fails the inspection.

Additional Filters

- **Character Expansions** are useful when dealing with print from a dot matrix printer or any print that is broken up in segments. The broken print is filtered so that it becomes solid by expanding the segments until they come together. Dilations expand each segment. Then, erosions decrease the size of the character in every direction except the direction in which the segments have connected. Dilations and erosions work together to make the segments solid without making the character fatter.
- **Filter Bright Defects** is useful when dust or other material settles on the print and appears brighter than the print in the image. This filter eliminates the bright specks and allows proper inspection of the print.

Brief Descriptions

- **OCVFont** — An OCVFont step is a container of one or more FontSymbol steps. The OCVFont contains a default FontSymbol that is used only for setting default parameters (parameters that any FontSymbol will inherit when inserted into the OCVFont). One or more OCVFonts are required for font-based OCV. OCVFonts are created and modified using the Custom Properties dialog box of the OCVFontTool or OCVRuntimeTool. OCVFonts are stored separately from the inspection Job file in the Vscape\Jobs\Fonts folder.
- **FontSymbol** — A FontSymbol is a collection of template images, settings, and tolerances that inspect a character or logo at runtime.
- **OCVFontTool** — An OCVFontTool uses an OCVFont to learn the layout, which means that it determines which characters from the OCVFont are in which locations in the FOV. Once the layout is learned, the OCVFontTool expects to find these symbols at the same locations during inspection. It uses the data from the FontSymbols in the OCVFont to verify the quality of the characters being inspected.
- **OCVRuntimeTool** — An OCVRuntimeTool uses an OCVFont (called the Master Font) to learn the layout, which means that it determines which characters from the OCVFont are in which locations in the FOV. Once the layout is learned, the OCVRuntimeTool creates a new OCVFont (called a Runtime Font) by training a new FontSymbol at each layout position, using the current image data. The OCVRuntimeTool expects to find the symbols at the same locations during inspection. It uses the data from the Runtime Font to verify the quality of the characters being inspected. The OCVRuntime Tool compensates for day-to-day changes in On-line print and helps minimize false rejects. The OCVRuntime Tool can be used when inspecting Date/Lot codes.
- **OCVFontlessTool** — An OCVFontlessTool does not require an OCVFont. Instead, it determines the location of characters in the FOV using a blob-analysis technique. Then, it stores training data for each character location as an OCVSymbolStep. The OCVFontlessTool expects to find the symbols at the same locations during inspection. It uses the trained data to verify the quality of the characters being inspected. The OCVFontless Tool checks symbol quality and not symbol correctness. The OCVFontless Tool can be used on Date/Lot codes when only symbol quality is a concern. You should *not* use the OCVFontless Tool to inspect Component ID codes. Table 10–1 contains usage hints.

Important Note

When placing the ROI around the code to inspect, be sure to leave clean area on either side of the code. This is called a quiet zone.

TABLE 10–1. Usage Hints

Use This Tool...	When You Want To...
OCVFontTool	Inspect for code quality and correctness. Ensure that code quality is always measured against the Font Library created by the Programmer.
OCVFontless	Inspect for code quality only.
OCVRuntime	Inspect for code quality and correctness. Inspect on-line printing.

- **AutoFind** — An AutoFind can optionally be used by any of the OCV Tools. This step determines the location of the layout at runtime. An AutoFind can be set up to use 1-Pin (no rotation) or 2-Pins (rotation). The Pins can be set up by selecting which layout positions to use.

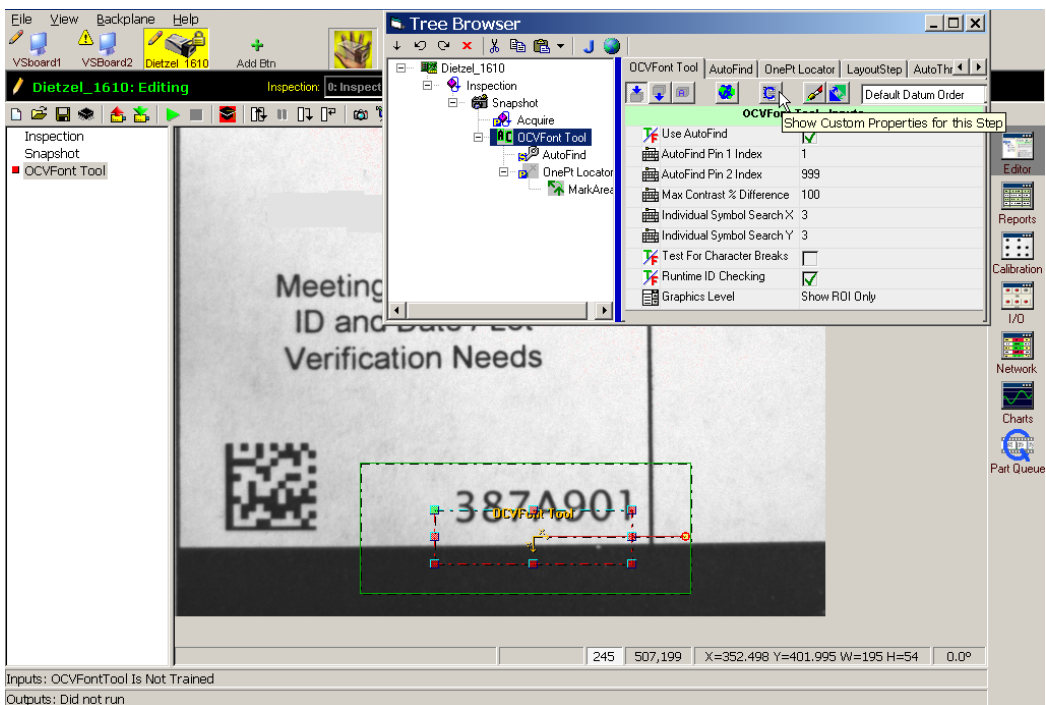
Custom Properties — Create/Modify OCVFonts (Library)

When Visionscape is first loaded, there are no fonts on the system. Fonts have to be created in order to perform font based OCV inspection. Fonts are stored in the Vscope\Jobs\Fonts folder. OCVFont files have the extension “.ocv”. The location of the stored fonts is not modifiable so that all Visionscape applications can locate the fonts in a single folder.

You can use the LayoutStep of the font based OCV tool to select a font for training and inspection from a list of available fonts on the system.

Custom Settings

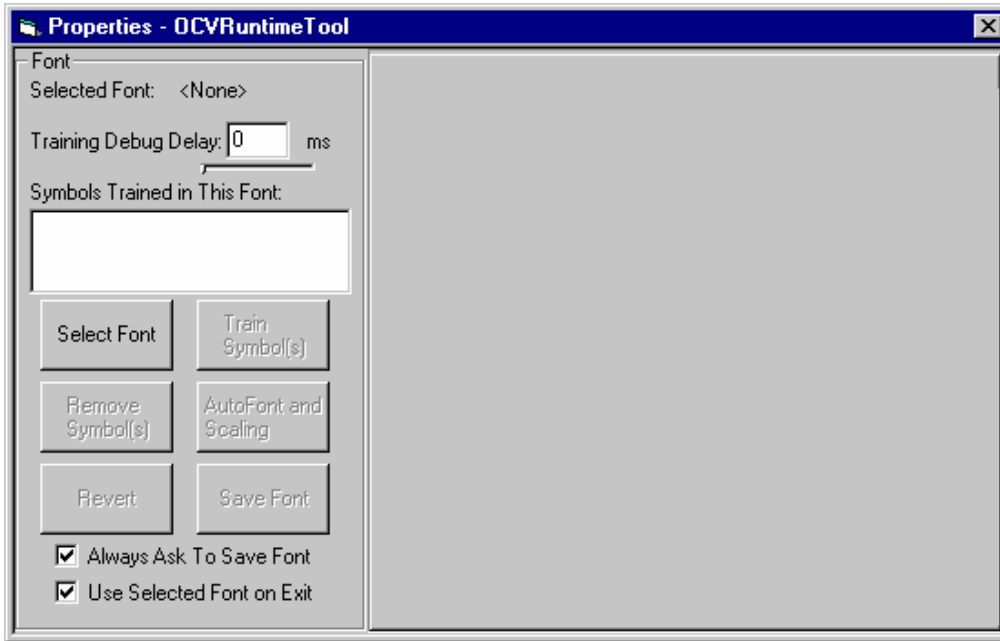
FIGURE 10–2. Custom Settings Dialog Box



The Custom Settings dialog box for the font based OCV tool creates and modifies fonts on the system. With a font based OCV tool selected as the current tool from the FrontRunner editing screen, clicking on the blue C found in the editor toolbar button brings up the Custom Properties page.

Main Custom Properties Dialog

FIGURE 10–3. Main Custom Properties Dialog Box



The right side of the Custom Properties dialog box displays the properties for the selected font. When no font is selected, no properties are displayed.

The left side of the Custom Properties dialog box displays the name of the selected font and the names of the symbols currently trained in that font. Setting the “Training Debug Delay” to a non-zero value causes the system to display detailed information during the training and scaling of symbols.

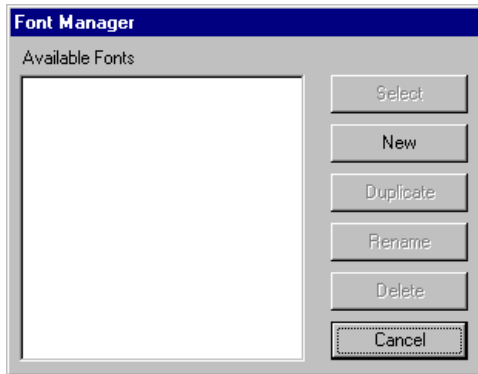
Buttons

- **Select Font** — Displays the “Font Manager” dialog (see “Font Manager Dialog Box” on page 10-9) and allows for a font to be selected for training or modification.
- **Train Font** — Initiates the training of an OCVFont (see “Training Fonts” on page 10-11). This is a change from previous Visionscape versions where you were required to train the OCVFont using the **Train** button in Visionscape.

- **Remove Symbols** — Displays the “Remove Symbols” dialog (see “Remove Symbol Dialog” on page 10-16) and allows for symbols to be easily removed from the selected font.
- **AutoFont and Scaling** — Instructs the Visionscape system to determine automatically the best font (from all fonts in the Vscape\Jobs\Fonts folder) for use on the current image (see “Automatic Font Selection and Scaling Dialog” on page 10-16).
- **Revert** — Reads in the last saved version of the selected font. This allows you to undo all changes since the last save.
- **Save Font** — Saves any changes made to the selected font.
- **Always Ask To Save Font** (checked, by default)
 - If this box is **checked** when the custom properties dialog box is closed or **Select Font** is clicked, you are asked if any changes should be saved.
 - If this box is **not checked**, you need to remember to save changes or they will be lost when the dialog box is closed or **Select Font** is clicked.
- **Use Selected Font on Exit** (checked, by default)
 - If this box is **checked** when the custom properties dialog box is closed, the OCVFont currently active in the custom properties dialog box becomes the selected OCVFont for the font based OCV tool that is being trained in Visionscape.
 - If this box is **not checked**, no change is made to the selected OCVFont for the tool being trained in Visionscape.

Font Manager Dialog Box

FIGURE 10–4. Font Manager Dialog Box

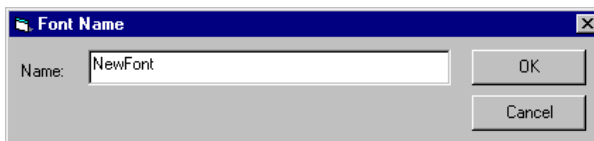


The “Available Fonts” list is the list of all OCVFonts found in the Vscape\Jobs\Fonts folder. OCVFont files have the extension “.ocv”.

Buttons

- **Select** — When clicked, this button returns you to the Main Custom Properties dialog box with the font selected in the “Available Fonts” list as the selected font.
- **New** — When clicked, this button prompts you to enter a name for the new font, as shown in Figure 10–5.

FIGURE 10–5. Font Name Dialog Box

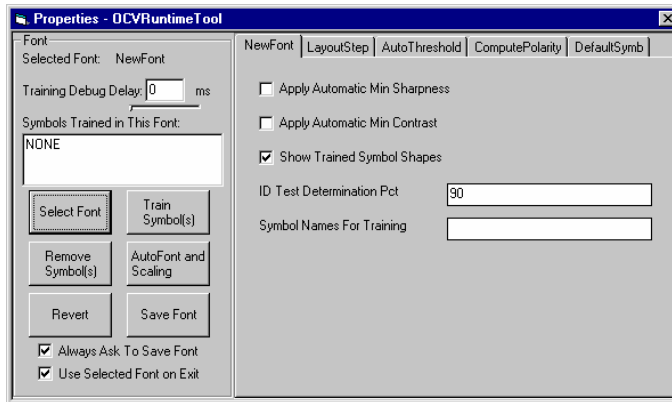


You must enter a unique name for the new font. The font name should be made up of alphanumeric characters and no more than 40 characters long. If an OCVFont with the name “DefaultFont.ocv” does not exist in the fonts folder, Visionscape will create one and give it the standard default property values. If an OCVFont with the name “DefaultFont.ocv” already exists in the fonts folder, it will not be overwritten. The values of all font properties are copied from “DefaultFont.ocv” to the new font. This allows new fonts to have customized settings based on your requirements.

- **Duplicate** — When clicked, this button prompts you for a name for the new font. You must enter a unique name for the new font. The font that is selected in the “Available Fonts” list is then copied, and the copy is given the name you provided.
- **Rename** — When clicked, this button prompts you for a name for the new font. You must enter a unique name for the new font. The font that is selected in the “Available Fonts” list is then renamed with the name you provided.
- **Delete** — When clicked, this button deletes the font that is selected in the “Available Fonts” lists from the fonts folder.
- **Cancel** — When clicked, this button returns you to the Main Custom Properties dialog box with no change to the selected font.

Training Fonts

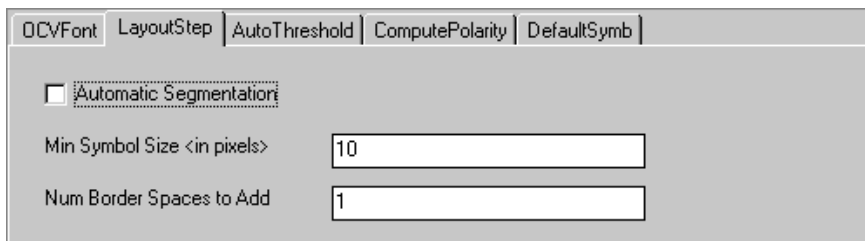
FIGURE 10–6. OCV Font (Library) Properties Dialog Box



When a new font is added and selected for training using the Font Manager dialog, it needs to be trained before it can be used by a font based OCV tool. First, the OCVFont shape needs to be positioned over the symbols to be trained.

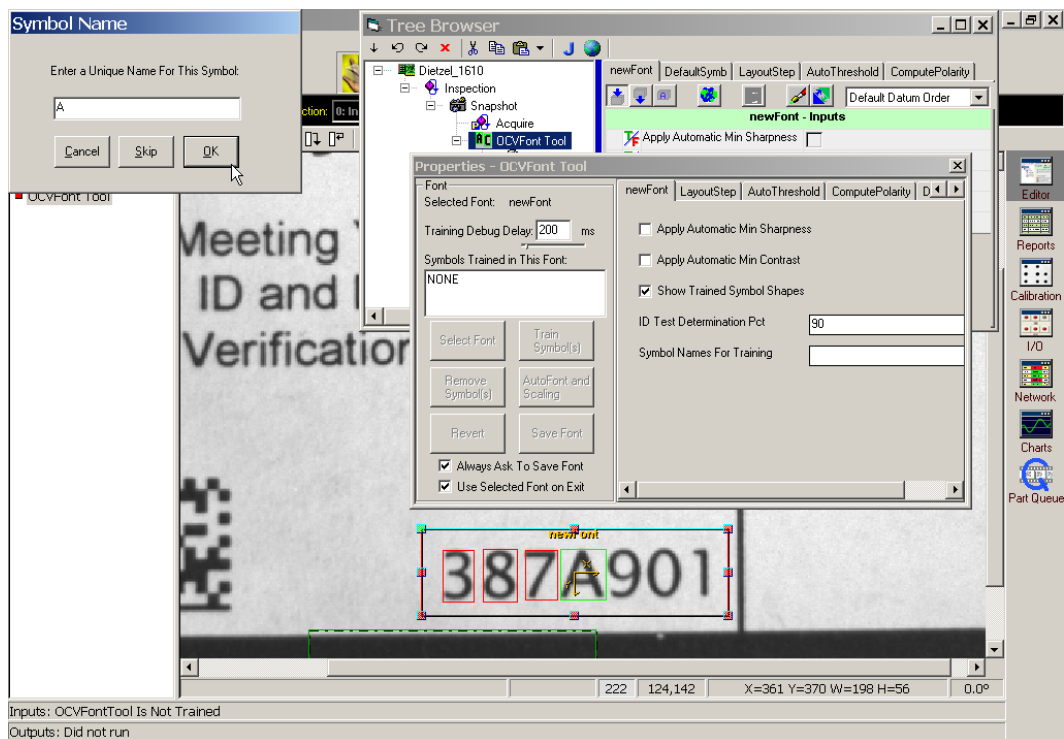
Second, select the LayoutStep tab and enable the Automatic Segmentation option, as shown in Figure 10–7.

FIGURE 10–7. LayoutStep Properties Page

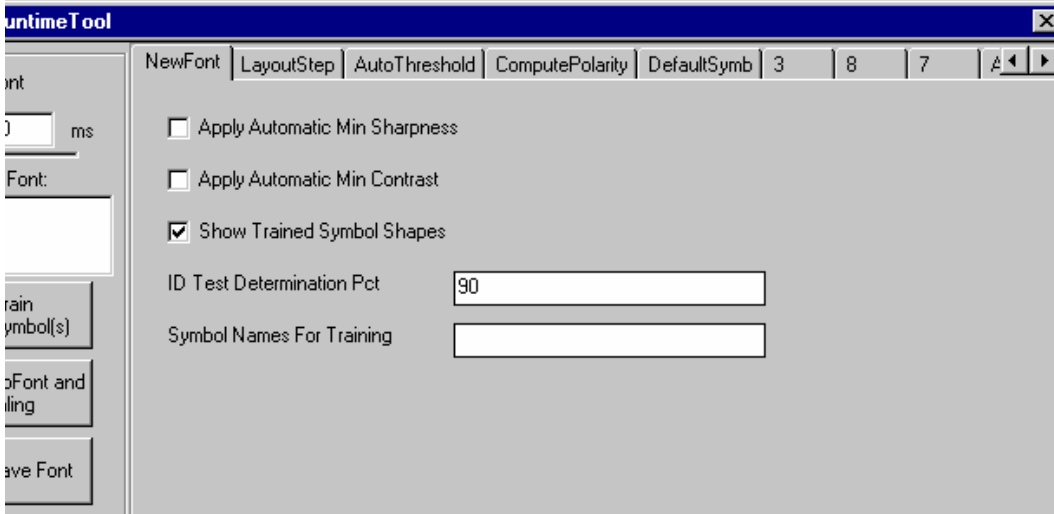


Clicking **Train Font** on the Custom Properties dialog box initiates the training. You are prompted to name each symbol found in the train ROI.

FIGURE 10–8. Prompt to Enter Unique Name for the Symbol



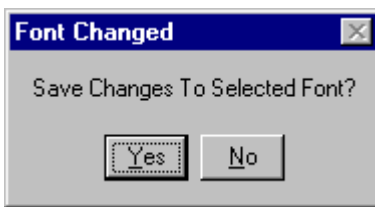
When training is complete, the right hand side of the Custom Properties dialog box is modified to contain a tab for each symbol that was added to the OCVFont, as shown in Figure 10–9.

FIGURE 10–9. Tabs for Each Added Symbol

Note: There are many options for training OCVFonts. The example in Figure 10–9 is the quickest way to train a font. For more details on training OCVFonts and the properties and settings involved, see “OCVFont” on page 10-23.

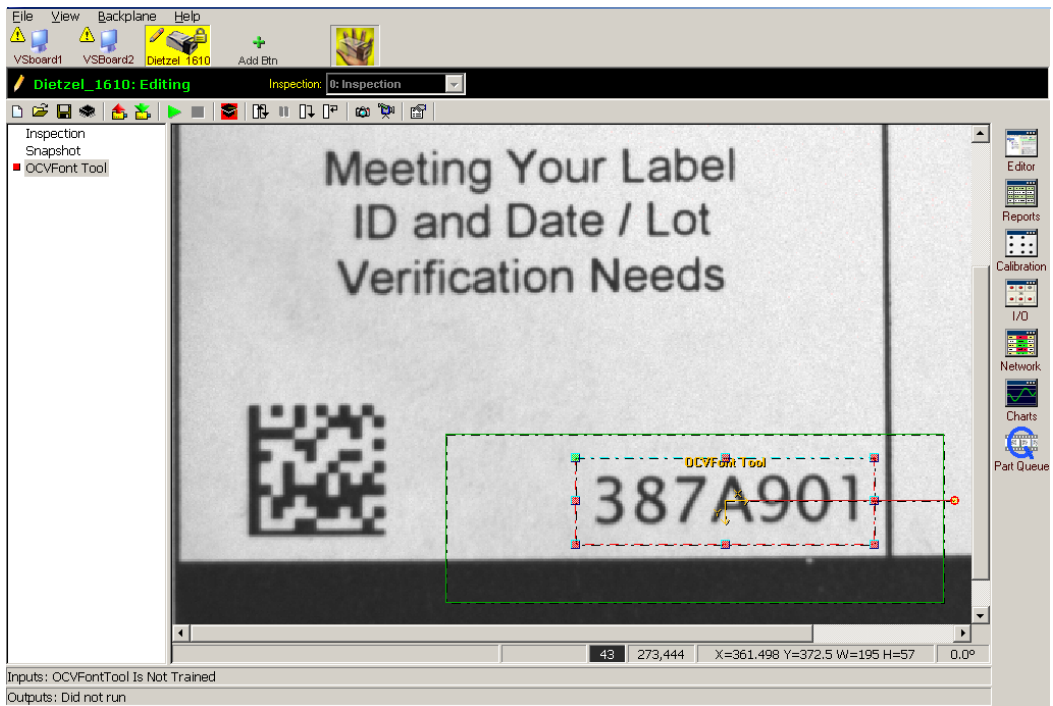
Training the OCVFontTool

To train the OCVFontTool, the Custom Properties dialog box must first be closed. If changes have not been saved, you are asked whether the changes should be saved. Clicking **Yes** saves the changes; clicking **No** loses any changes that were made.

FIGURE 10–10. Prompt to Save Changes

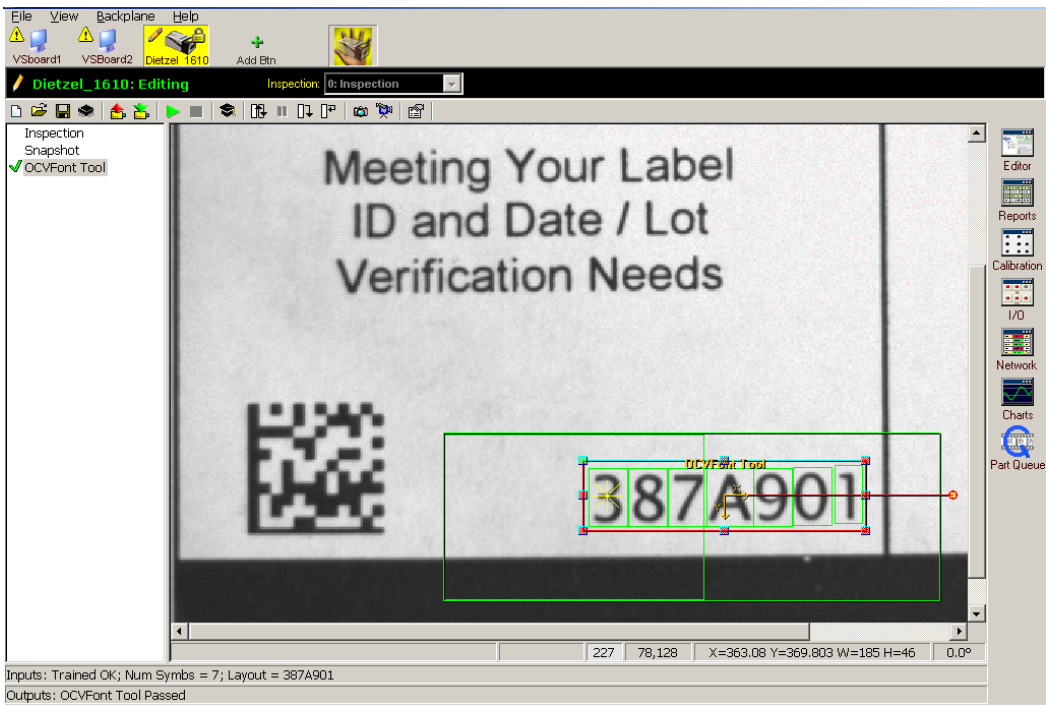
To train the OCVFontTool, the tool shape must be placed around the characters that are going to be inspected.

FIGURE 10–11. Position Tool ROI Around Characters to be Inspected



Click **Editor** to display the OCVFontTool’s datum page. Clicking on the “LayoutStep” tab in the datum page will display all properties for the LayoutStep. The correct font needs to be selected from the “Selected Font” datum’s list of available fonts.

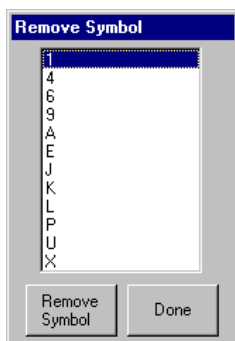
FIGURE 10–12. OCVFont Tool Training Completed



Clicking **Train** causes the tool to find all characters within the ROI that are trained as symbols in the selected font. The tool sets up its inspection “Layout” and is then ready to run. The inspected “layout” is shown in the input information bar under the image.

Remove Symbol Dialog

FIGURE 10–13. Remove Symbol Dialog Box

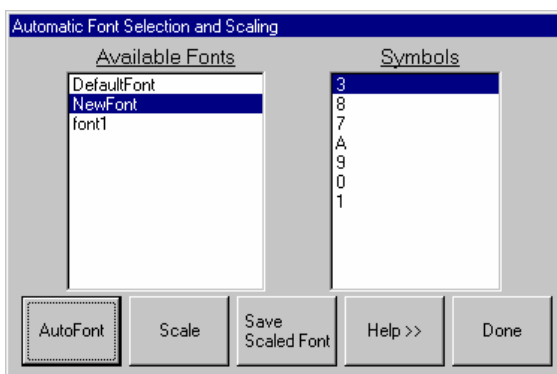


The Remove Symbol dialog box deletes symbols from the Custom Properties selected font. By selecting a symbol name from the list and clicking **Remove Symbol**, you are able to remove the selected symbol from the font. The **Done** button returns you to the Custom Properties Main dialog box.

Automatic Font Selection and Scaling Dialog

Figure 10–14 displays the Automatic Font Selection and Scaling dialog box.

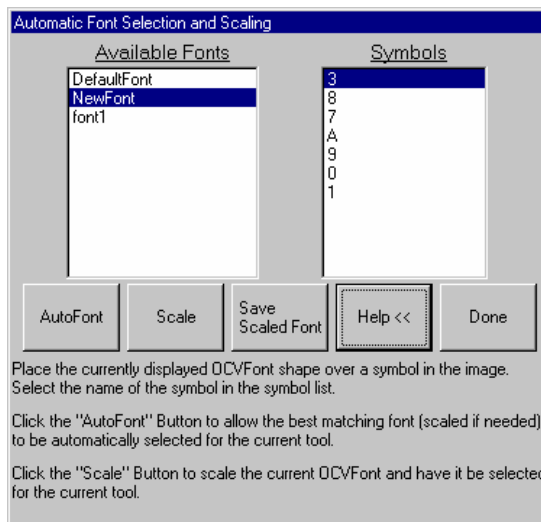
FIGURE 10–14. Automatic Font Selection and Scaling Dialog Box



Buttons

- **Done** — Returns you to the Custom Properties Main dialog box when you have finished with AutoFont and Scaling.
- **Help** — Displays or removes help information from the Automatic Font Selection and Scaling dialog, as shown in Figure 10–15.

FIGURE 10–15. Help Displayed



Automatic Font Selection — The “AutoFont” Button

The automatic font selection and scaling feature allows the system to scan through all of the OCVFonts in the Vscape\Jobs\Fonts folder to determine which one will work best with the current image data. When scaling an OCVFont is required to make it the best match, the system determines the proper scaling factors to use to create a scaled version of the OCVFont. This scaled version of the OCVFont will be created at the end of the automatic selection process. The name of the scaled OCVFont will reflect the change in width and height used to perform the scaling. By default, scaled OCVFonts are stored in the Job file as part of the associated font based tool. These scaled OCVFonts can be stored on the disk using the Save Scaled Font button.

The Automatic Font Selection and Scaling dialog box has two lists:

- The list on the left hand side of the dialog box is a font list, containing the names of all the OCVFonts in the Vscape\Jobs\Fonts folder.
- The list on the right hand side of the dialog box is a symbol list, containing the names of all the FontSymbols found in the OCVFont that is currently selected in the font list.

Choosing a Symbol

Select a symbol from the symbol list. This symbol will determine the best font. It is important to select a complex, uniquely shaped character. For example, a 5 would be better than a 0 or a 1. The character should appear in the current image and be crisply formed and printed (i.e., no smudges or blurring).

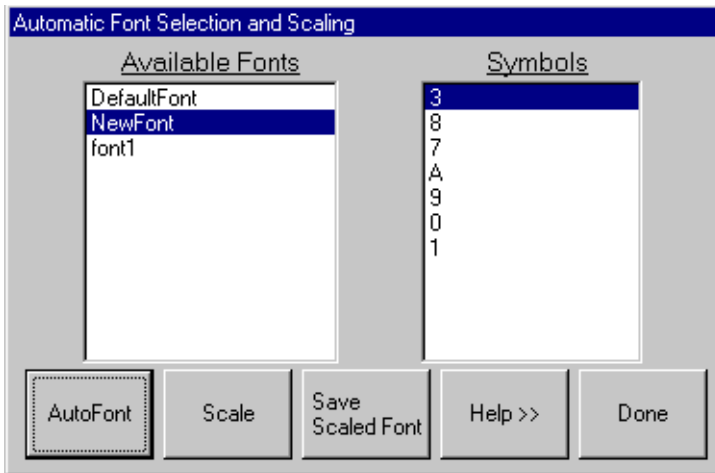
Positioning the OCVFont Shape

The OCVFont shape of the currently selected font sets up the automatic font selection and scaling process. This shape needs to be positioned and sized around a character in the current image that matches the character selected in the symbol list. It is important that the shape be positioned and sized very tightly over the selected character (do not leave any border). This ensures that the system will not mistake any part of other characters as being part of the selected character.

Note: You may find it easier if the trained symbol shapes are not displayed. Click **Done** and uncheck **Show Trained Symbol Shapes** (see Figure 10–9, “Tabs for Each Added Symbol,” on page 10-13).

Performing the Automatic Font Selection and Scaling

FIGURE 10–16. Ready to Perform Automatic Font Selection and Scaling



Once a character has been selected and the OCVFont shape has been correctly sized and positioned around that character in the image, the system is ready to perform the automatic font selection and scaling. Click **AutoFont** to start the process.

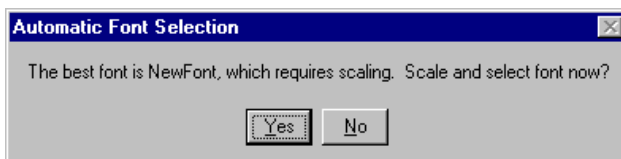
During the font selection process, each OCVFont that is in the Vscape\Jobs\Fonts folder is tested. The first part of the test determines if a symbol with the name of the selected character is trained in the OCVFont. If not, the process continues to the next OCVFont. If the symbol is in the font, the system will create several scaled versions of the template based on the size of the symbol in the font and the size of the OCVFont shape. Each scaled template is assigned a score value after it is compared to the actual image data inside the OCVFont Box. If the score value for any scaled template is better than any previous score values, that score value is stored as the **BestScore**, along with the name of the font that the template originated from and the scaling factors used to derive the scaled template.

When the “Train Debug Delay” property on the Custom Properties Main dialog box is set to a non zero value, the scaled templates and match scores are displayed in the upper left corner of the screen.

After all OCVFonts have been tested, the OCVFont that is associated with the **BestScore** is considered to be the font that will work best with the current image data. When an OCVFont has been automatically selected, a message box appears

to display the name of the best matching OCVFont and whether or not it requires scaling to match the current image.

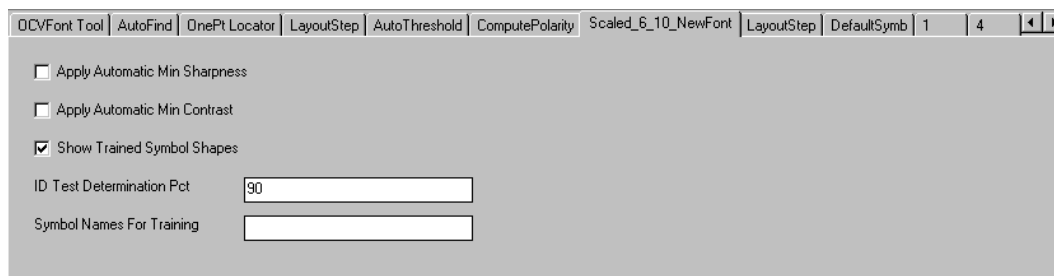
FIGURE 10–17. Name of Best Matching OCVFont



The dialog box also asks if the chosen font should be automatically selected into the current font based OCV Tool.

- Choosing No results in no scaling being done to the best matching OCVFont.
- Choosing Yes results in the OCVFont being scaled and the resulting scaled font becoming the selected font of the OCV Tool.

FIGURE 10–18. Scaled_6_10_NewFont Embedded in OCVFontTool



If none of the OCVFonts were able to match the current image data with at least a 20% score, then no best font is found and an error message is displayed.

Font Scaling — The “Scale” Button

The font scaling feature is useful when you already know which font needs to be used, but the FOV has changed. Font scaling allows re-sampling of all FontSymbol train data to match the current FOV. A scaled version of the OCVFont will be created and the name of the scaled OCVFont will reflect the change in width and height used to perform the scaling. By default, scaled OCVFonts are stored in the Job file as part of the associated font based tool. These scaled OCVFonts can be stored on the disk using the **Save Scaled Font** button.

Choosing a Symbol

Choose the OCVFont that needs scaling from the font list. Select a symbol from the symbol list. This symbol will determine the changes in width and height that are needed to perform the font scaling. The character should appear in the current image and be crisply formed and printed (i.e., no smudges or blurring).

Positioning the OCVFont Shape

The OCVFont shape of the currently selected font sets up the font scaling process. This shape needs to be positioned and sized around a character in the current image that matches the character selected in the symbol list. It is important that the shape be positioned and sized very tightly over the selected character (do not leave any border). This ensures that the system will correctly calculate the changes in width and height.

Note: You may find it easier if the trained symbol shapes are not displayed. Click **Done** and uncheck Show Trained Symbol Shapes (see Figure 10–9, “Tabs for Each Added Symbol,” on page 10-13).

Performing Font Scaling

Once a character has been selected and the OCVFont shape has been correctly sized and positioned around that character, the system is ready to perform the font scaling. Click **Scale** to start the process. The system compares the trained width of the selected FontSymbol with the width of the OCVFont box and calculates the required change in width to scale the FontSymbol in X. The system then compares the trained height of the selected FontSymbol with the height of the OCVFont box and calculates the required change in height to scale the FontSymbol in Y. Then, a new OCVFont is created and given the name of the source OCVFont with the addition of the change in width and change in height values. For example, OldFont_2_-5 indicates that the OCVFont named “OldFont” was scaled by increasing the width of the symbols by 2 and decreasing the height of the symbols by 5. Each symbol that is in the source OCVFont is then scaled and added to the new OCVFont.

When the “Train Debug Delay” property on the Custom Properties Main dialog box is set to a non zero value, the scaled templates and other FontSymbol training details are displayed in the upper left corner of the screen.

OCVFont

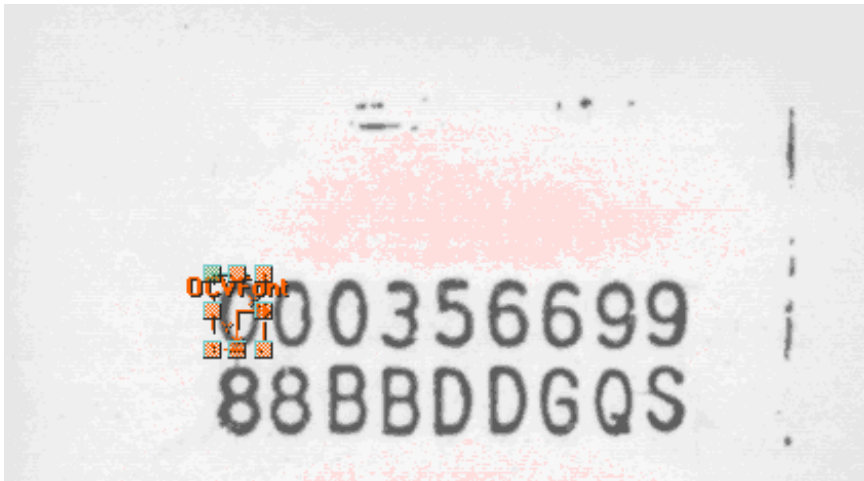
This step is a container of one or more FontSymbol steps. It trains and groups a set of characters of a particular font style and size.

The OCVFont contains a default FontSymbol that is used only for setting default parameters (parameters that any FontSymbol inherits when inserted into the OCVFont). One or more OCVFonts are required for font based OCV.

Creating FontSymbols

As a container step, the OCVFont step creates FontSymbol steps. Creating FontSymbol steps can be accomplished by individual training or automatic segmentation using the Custom Properties dialog box of the OCVFontTool or OCVRuntimeTool.

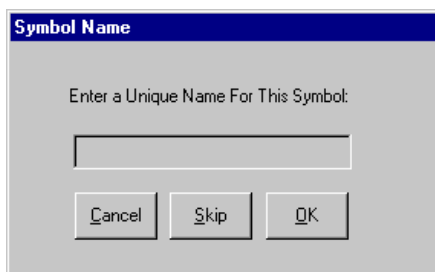
FIGURE 10–19. OCVFont — Example 1



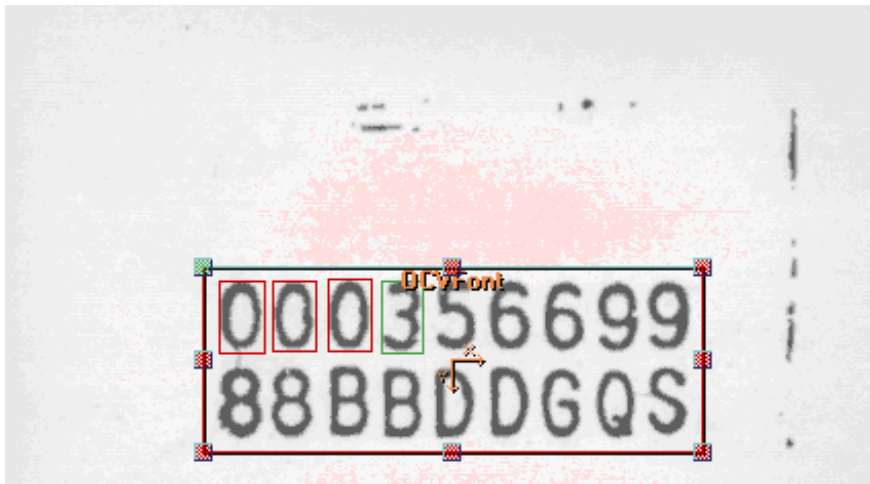
By default, Visionscape is designed such that you perform individual training of characters. This is to activate runtime ID checking of special characters like O, 0, B, 8, D, etc. ID checking requires that these symbol boxes be the same size.

Individual character training requires that the OCVFont ROI be placed close around a single character in the image, leaving a 1-2 pixel border, as shown in Figure 10–19. This box should not include any portion of the adjacent characters. The minimum character width we recommend is 20 pixels. When **Train Font** is clicked, the Symbol Name dialog box is displayed, asking for a unique name for the symbol, as shown in Figure 10–20.

FIGURE 10–20. Symbol Name Dialog Box



Clicking **Cancel** or **Skip** aborts the training of this FontSymbol. When a unique name is entered and **OK** is clicked, a FontSymbol is created, and templates (created from the ROI area of the image) and default parameters are stored in that FontSymbol. The OCVFont shape must be placed around the next character to train it. This process continues until all characters in the image have been trained as FontSymbols and added to the OCVFont, as shown in Figure 10–21. Only one example of a given character needs to be trained.

FIGURE 10–21. OCVFont — Example 2

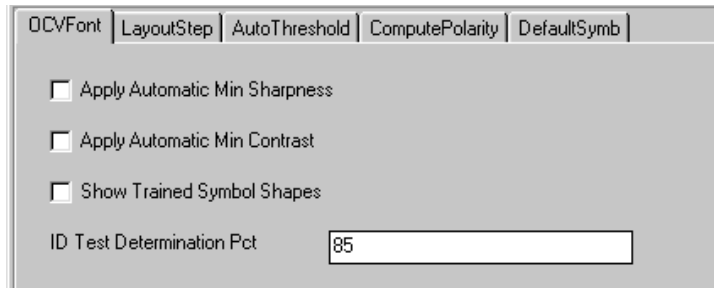
The Automatic Segmentation feature can be enabled from the Custom Properties dialog box. The Automatic Segmentation setting can be found on the Layout Step property tab for the selected font. Automatic segmentation training requires that the OCVFont shape be placed around all the characters in the image that are going to be added as FontSymbols in the OCVFont. Then, when **Train Font** is clicked, a green box appears in the image over one of the characters. A dialog box is displayed, asking for a unique name for this symbol. Clicking **Cancel** aborts the training of this FontSymbol and ends the automatic segmentation training. Clicking **Skip** aborts the training of this FontSymbol and moves on to the next character in the image. Only one example of a character needs to be trained.

When a unique name is entered and **OK** is clicked, a FontSymbol is created, and templates (created from the ROI area of the image) and default parameters are stored in that FontSymbol. The settings for the FontSymbol can be found in the FontSymbol section located on page 10–30. The green box changes to a red box and a new green box appears over the next character in the image. This process continues until all characters in the image have been trained as FontSymbols and added to the OCVFont or the process is canceled.

OCVFont Tab

When an OCVFont is selected in the Custom Properties dialog box, the OCVFont tab displays the current settings for that OCVFont.

FIGURE 10–22. OCVFont Properties Page



- **Apply Automatic Min Sharpness** — When enabled (disabled by default), as FontSymbols are trained and added to this OCVFont, a minimum tolerance for sharpness is calculated for the FontSymbol. This value is 65% of the sharpness value calculated using the trained grayscale template of the FontSymbol.
- **Apply Automatic Min Contrast** — When enabled (disabled by default), as FontSymbols are trained and added to this OCVFont, a minimum tolerance for contrast is calculated for the FontSymbol. This value is 50% of the contrast value calculated using the trained grayscale template of the FontSymbol.
- **Show Trained Symbol Shapes** — When enabled (disabled by default), the shapes of all FontSymbols that are part of this OCVFont are displayed whenever the shape for this OCVFont is selected in the buffer view.
- **ID Test Determination Pct** — When a FontSymbol is trained as part of an OCVFont, it is compared against all of the FontSymbols already in the OCVFont. When FontSymbols are found to be similar, special tests are set up to check for the presence of the correct symbol at runtime.

ID Test Determination Pct adjusts the level at which symbols are similar enough to require special runtime tests. This property defaults to 90% with a minimum of 10% and a maximum of 100%. Smaller percentages cause more symbols to be flagged as similar, while larger percentages cause less symbols to be flagged as similar.

Examples — When two symbols are found to be 75% similar and the value of this property is 85%, no special tests are set up for runtime ID checking; if two symbols are found to be 90% similar and the value of the property is 85%, a special test is set up for runtime ID checking.

LayoutStep Tab

The **LayoutStep** of the OCVFont is used for automatic segmentation of the image, when enabled. When an OCVFont is selected in the Custom Properties dialog box, the LayoutStep tab displays the current settings for that OCVFont's LayoutStep.

FIGURE 10–23. LayoutStep Properties Page

- **Automatic Segmentation** — When enabled (disabled by default), the training of the OCVFont causes the image to be segmented using blob analysis. A dialog box is displayed, asking for a unique name to give the FontSymbol before training the FontSymbol for each position found.
- **Min Symbol Size <in pixels>** — Adjusts the minimum size that a blob must be in order to be considered a symbol.

Default: 10 pixels
Range: 5 to 256 pixels

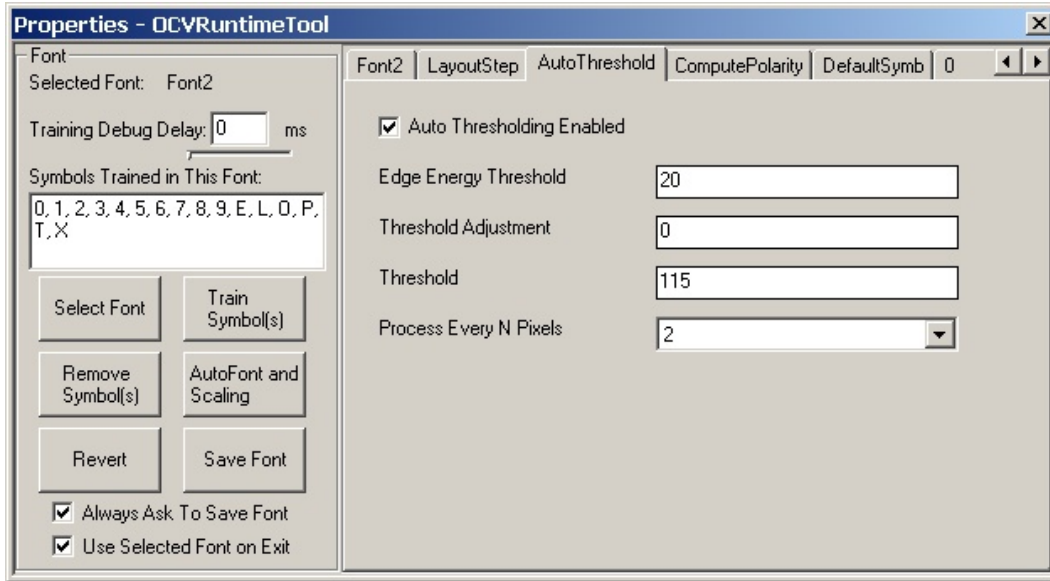
- **Num Border Spaces to Add** — Determines how many pixels to allow between actual character pixels and the edge of the box that defines the FontSymbol.

Default: 1 pixel
Range: 0 to 19 pixels

AutoThreshold Tab

The AutoThreshold step of the OCVFont is part of the LayoutStep that is used for automatic segmentation of the image, when enabled, as shown in Figure 10–24.

FIGURE 10–24. AutoThreshold Properties Page



- **Auto Thresholding Enabled** — Enables (default) and disables the automatic thresholding. When enabled, a threshold is calculated using the ROI of the step. This calculation uses edge detection to determine foreground and background information. The calculated threshold is displayed in **Threshold**. Although it is called **AutoThreshold** step, the Auto portion can be disabled. When disabled, no calculation is done. The threshold used by the step is whatever value is in **Threshold**. The **Edge Energy Threshold** and **Threshold Adjustment** properties are not used when Auto is disabled.
- **Edge Energy Threshold** — Defines the pixel value at which a pixel in a Sobel Edge Enhancement is considered to be an edge pixel. This property is only used when **Auto Thresholding Enabled** is enabled.

Default: 10

Range: 0 to 255

- **Threshold Adjustment** — Used to offset or bias the dynamically calculated threshold, when Auto is enabled.

Default bias: 0

Range: -255 to 255

- **Threshold** — Displays the dynamically calculated threshold when Auto is enabled. When Auto is disabled, the value of this property is the threshold that is used by the step.

Default 135

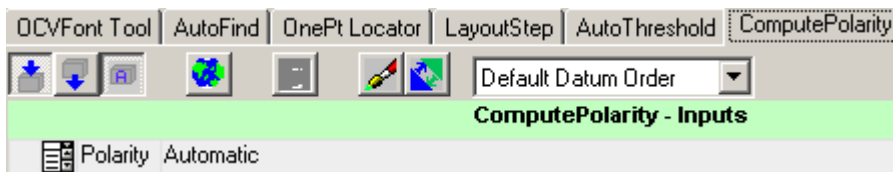
Range: 0 to 255

- **Process Every N Pixels** — As N increases, the accuracy of the threshold calculation may decrease, but the processing time for this step is reduced.

ComputePolarity Tab

The ComputePolarity step of the OCVFont is part of the LayoutStep that is used for automatic segmentation of the image, when enabled, as shown in Figure 10–25.

FIGURE 10–25. ComputePolarity Properties Page



- **Polarity** — Allows the step to be set up to always return Light_On_Dark, always return Dark_On_Light, or return an automatically determined polarity. The default setting is Automatic.

DefaultSymb Tab

The DefaultSymb step of the OCVFont sets default parameters that any FontSymbol trained and added to the OCVFont receives.

FontSymbol

A FontSymbol is a collection of template images, settings, and tolerances that inspect a character or logo at runtime. FontSymbols are created during the training of an OCVFont. They are used by the OCVFontTool and OCVRuntimeTool steps to learn the layout at train time and inspect the layout at runtime.

FontSymbols are trained when they are added to an OCVFont.

FIGURE 10–26. FontSymbol Properties Page

3 8 7 A 9 0 1

Num of ON Pixels in Template 147

Polarity Dark on Light

Legibility (%) 25.0

Allowed Movement in X (+/-) 50

Allowed Movement in Y (+/-) 50

Residue Limit Units Percentage

Initial Residue Limit 100.0

Final Residue Method Total Residue Area

Final Residue Limit 15.0

Final Residue Max Blob Size 10

Maximum Flaw Size 1

☒ Appearance Flaw Break Test

Min Appear. Flaw Break Size 2

Sharpness Limit Units Gray Level

Minimum Allowed Sharpness 0

Contrast Limit Units Gray Level

Minimum Allowed Contrast 0

☒ Auto Threshold Enabled

Auto Threshold Adjustment 0

Manual Threshold 135

Edge Energy Threshold 20

Character Expansions 0

☐ Filter Bright Defects

Bright Defect % Range 0

Output Mask Type Mask Template

Apply To All FontSymbols

Apply To Default Symbol

TABLE 10–2. Links to Property Descriptions

For Information About...	Go To...
Allowed Movement in X (+/-)	page 10–32
Allowed Movement in Y (+/-)	page 10–32
Appearance Flaw Break Test	page 10–34
Apply To All FontSymbols	page 10–37

TABLE 10–2. Links to Property Descriptions

For Information About...	Go To...
Apply To Default Symbol	page 10–38
Auto Threshold Adjustment	page 10–36
Auto Threshold Enabled	page 10–36
Bright Defect % Range	page 10–37
Character Expansions	page 10–37
Contrast Limit Units	page 10–35
Edge Energy Threshold	page 10–37
Filter Bright Defects	page 10–37
Final Residue Limit	page 10–33
Final Residue Max Blob Size	page 10–34
Final Residue Method	page 10–33
Initial Residue Limit	page 10–32
Legibility (%)	page 10–32
Manual Threshold	page 10–36
Maximum Flaw Size	page 10–34
Min Appear. Flaw Break Size	page 10–35
Minimum Allowed Contrast	page 10–36
Minimum Allowed Sharpness	page 10–35
Num of ON Pixels in Template	page 10–31
Output Mask Type	page 10–37
Polarity	page 10–31
Residue Limit Units	page 10–32
Sharpness Limit Units	page 10–35

- **Num of ON Pixels in Template** — Displays the number of foreground pixels in the trained binary template.
- **Polarity** — Allows the step to always train with polarity Light_On_Dark, always train with polarity Dark_On_Light, or train using an automatically calculated polarity. The default setting is Automatic.

- **Legibility (%)** — Passes/fails the symbol based on this minimum correlation percentage. The symbol fails inspection when the correlation percentage is less than this value.

Default: 25%

Range: 0% to 100%

- **Allowed Movement in X (+/-)** — Sets the maximum number of pixels that a symbol can move in the X-axis (relative to other symbols) from its trained position.

Default: 50 pixels (any movement is allowed)

Range: 0 to 50 pixels. The maximum of 50 pixels comes from the parent OCVTool setting “Individual Symbol Search X”, which limits the search range in X to a maximum of 50 pixels in either direction.

- **Allowed Movement in Y (+/-)** — Set the maximum number of pixels that a symbol can move in the Y-axis (relative to other symbols) from its trained position.

Default: 50 pixels (any movement is allowed)

Range: 0 to 50 pixels. The maximum of 50 pixels comes from the parent OCVTool setting “Individual Symbol Search Y”, which limits the search range in Y to a maximum of 50 pixels in either direction.

- **Residue Limit Units** — Inputs the residue limits in either a maximum pixel count value or a percentage value (percentage value is based on the number of On pixels in the trained template). The default value is Percentage. When the value of this property changes, the values of **Initial Residue Limit** and **Final Residue Limit** are changed to match the selected units.
- **Initial Residue Limit** — Provides a quick check of the character quality and correctness. The initial residue calculation is done before any image processing is performed on the residue image.

When the system looks at the symbol being inspected, it determines the residue of the symbol, which is a count of those pixels that differ between the trained template and the current image. Based on the value of this property, the system determines if the residue is within tolerances. If it is not within tolerances, the symbol fails. Otherwise, the system continues on with the rest of the inspection procedure.

When **Residue Limit Units** is set to Percentage:

Default: 100.0%

Range: 0.0% to 100.0%

The value of this property is the smallest percentage of residue pixels (relative to the trained On pixel count) in the inspected image that will make the symbol fail the inspection.

When **Residue Limit Units** is set to Pixels:

Default: symbol size

Range: 0 to symbol size

The value of this property is the smallest count of residue pixels in the inspected image that will make the symbol fail the inspection.

This property is good for catching smudges that are aesthetically poor, but would pass after all inspection operations are performed on it. A property value of 100% or symbol size means initial residue is ignored.

- **Final Residue Method** — Selects between three algorithms for final residue analysis:
 - **Total Residue Area** — This is the default. This choice counts all On pixels in the residue image and use the value in **Final Residue Limit** (pixel or percent) as the tolerance.
 - **Max Residue Blob** — Only counts the pixels in the largest blob of the residue image and use the value in **Final Residue Max Blob Size** as the tolerance.
 - **Both** — Performs both methods.
- **Final Residue Limit** — Sets the amount of objectionable residue that is to be deemed passable when **Final Residue Method** is set to **Total Residue Area** or **Both**. Final residue calculation is done after the image processing on the residue image that is associated with **Maximum Flaw Size**.

An assignment of 0% (residue pixel count = 0) means that no residue is passable. An assignment of 100% (residue pixel count = symbol size) means that objectionable residue as large as the area of the prototype itself is passable.

When **Residue Limit Units** is set to Percentage:

Default: 15.0% (meaning a 15% variation is acceptable)

Range: 0.0% to 100.0%

The value of this property is the smallest percentage of residue pixels (relative to the trained On pixel count) in the inspected image that makes the symbol fail the inspection.

When **Residue Limit Units** is set to Pixels:

Default: 15% of the symbol size

Range: 0 to symbol size

The value of this property is the smallest count of residue pixels in the inspected image that makes the symbol fail the inspection.

Note: Determining the proper value for **Final Residue Limit** is a subjective decision; the higher the quality of the character/symbol desired, the lower the **Final Residue Limit** should be.

- **Final Residue Max Blob Size** — Used when **Final Residue Method** is set to Max Residue Blob or Both. A blob analysis is performed on the residue image and the largest blob is found. If this blob has an area that is greater than the value of this property, the symbol fails the inspection.

Default: 10

Range: 1 to 512

- **Maximum Flaw Size** — Represents the maximum width in pixels that a discrepancy is allowed to be before it is considered objectionable. The larger the number assigned, the larger a discrepancy is allowed before causing the symbol inspection to fail.

Default: 1 pixels

Range: 0 to 20 pixels

- **Appearance Flaw Break Test** — Determines whether the **FontSymbol** is to inspect for character breaks in the symbol. When enabled, the inspection fails if a break is found in the symbol. When disabled, the inspection ignores breaks in the symbol. This property defaults to enabled.

- **Min Appear. Flaw Break Size** — Is the smallest size break that causes a character break failure.

Default: 2 pixels

Range: 1 to 10 pixels

- **Sharpness Limit Units** — Sets the units for Minimum Allowed Sharpness.
 - When set to “**Gray Level**”, Minimum Allowed Sharpness is an absolute minimum value that the calculated sharpness value must be in order for the inspection to pass.
 - When set to “**Percentage**”, Minimum Allowed Sharpness calculates a percentage of the trained sharpness value, which is then used as an absolute minimum value that the calculated sharpness value must be in order for the inspection to pass.
 - When switched from “**Gray Level**” to “**Percentage**,” Minimum Allowed Sharpness is updated to be the percentage value that corresponds to the gray level value that it previously held.
 - When switched from “**Percentage**” to “**Gray Level**,” Minimum Allowed Sharpness is updated to be the gray level value that corresponds to the percentage value that it previously held.

Default: “Gray Level”

- **Minimum Allowed Sharpness** — This value determines how crisp a symbol must be to pass inspection. It is measured by average edge strength over the entire symbol. Typical edge strengths are from 20 to 80 sharpness units. When Minimum Allowed Sharpness is 0, no sharpness check is performed.

Default: 0

Range: 0 to 256 “Gray Level” or 0 to 100 “Percentage”

- **Contrast Limit Units** — Sets the units for Minimum Allowed Contrast.
 - When set to “**Gray Level**,” Minimum Allowed Contrast is an absolute minimum value that the calculated contrast value must be in order for the inspection to pass.

- When set to “**Percentage**,” **Minimum Allowed Contrast** calculates a percentage of the trained contrast value, which is then used as an absolute minimum value that the calculated contrast value must be in order for the inspection to pass.
- When switched from “**Gray Level**” to “**Percentage**,” **Minimum Allowed Contrast** is updated to be the percentage value that corresponds to the gray level value that it previously held.
- When switched from “**Percentage**” to “**Gray Level**,” **Minimum Allowed Contrast** is updated to be the gray level value that corresponds to the percentage value that it previously held.

Default: “Gray Level”

- **Minimum Allowed Contrast** — The Contrast is the measurement that defines the grayscale foreground to background relationship of the symbol data. To calculate the contrast value, the average gray level value of the background pixels is subtracted from the average gray level of the foreground pixels. Whenever this property has a value of 0, no contrast checks are performed.

Default: 0

Range: 0 to 256 “Gray Level” or 0 to 100 “Percentage”

- **Auto Threshold Enabled** — Enables (default) or disables the automatic calculation of a threshold for binarizing the image at both train and run time. When enabled, the calculated threshold is displayed in **Manual Threshold**. When disabled, no calculation is done. The threshold used for binarizing is whatever value is in **Manual Threshold**. **Edge Energy Threshold** and **Threshold Adjustment** are not used when this property is disabled.
- **Auto Threshold Adjustment** — Used to offset or bias the dynamically calculated threshold, when **Auto Threshold Enabled** is enabled.

Default: 0

Range: -64 to 64

- **Manual Threshold** — Displays the dynamically calculated threshold when **Auto Threshold Enabled** is enabled. When **Auto Threshold Enabled** is disabled, the value of this property is the threshold that is used for binarizing the image.

Default: 135

Range: 0 to 255

- **Edge Energy Threshold** — Defines the pixel value at which a pixel in a Sobel Edge Enhancement is considered to be an edge pixel. This is only used when **Auto Threshold Enabled** is enabled.

Default: 20

Range: 0 to 255

- **Character Expansions** — Useful when dealing with print from a dot matrix printer or any print that is broken up in segments. The more sparse the print, the higher the value of this property should be. This allows for the broken print to become solid by expanding the segments until they come together. Dilations expand each segment. Then, erosions decrease the size of the character in every direction except the direction in which the segments have connected. Dilations and erosions work together to make the segments solid without making the character fatter.

Default: 0

Range: 0 to 9

- **Filter Bright Defects** — When enabled, runtime inspection of the symbol includes a pre-processing step for filtering out any bright defects in the image. The default setting is disabled.
- **Bright Defect % Range** — The value is a percentage that determines the threshold at which the bright defect filter processes. The threshold is calculated by taking this percentage of the range between the binarizing threshold and 255. This means that the binary threshold would be used when **Filter Bright Defects** is enabled.

Default: 0

Range: 0 to 100

- **Output Mask Type** — Used in conjunction with the **DynamicMask** step:
 - **None** — Adds nothing to the mask.
 - **Mask Template (default)** — Only the foreground area of the symbol is added to the mask.
 - **Mask ROI** — The entire area within the symbol's ROI is added to the mask.
- **Apply To All FontSymbols** — Sets the properties of all symbols in the OCVFont to the values currently shown on the page.

- **Apply To Default Symbol** — Sets the properties of the default symbol of the OCVFont to the values currently shown on the page.

The factory default settings work well for most applications. When adjustments to Pass/Fail limits are required, the following settings should be modified first:

- **Final Residue Limit**
- **Maximum Flaw Size**

Increasing Final Residue to 20% allows more variations to be accepted. Changing the Final Residue % has a gradual effect on Pass/Fail. Using a high Final Residue %, such as 50%, on small characters such as - can reduce false rejects.

Increasing the **Maximum Flaw Size** has a pronounced effect on Pass/Fail. Increasing **Maximum Flaw Size** allows more character variations to be acceptable. For many applications, this value should not be set greater than 2.

AutoFind

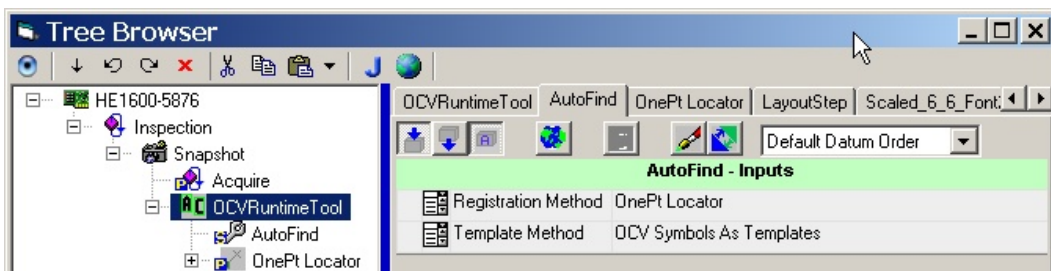
An AutoFind can optionally be used by any of the OCV Tools. This step determines the location of the layout at runtime. An AutoFind can be set up to use 1-Pin (no rotation) or 2-Pins (rotation). The Pins can be set up by selecting which layout positions to use on the OCV Tool properties page.

Training

The AutoFind pin(s) get trained automatically when the OCV Tool is trained. When all characters have been located in the FOV during OCV Tool training, the AutoFind Pin1 Index and AutoFind Pin2 Index properties of the OCV Tool select which characters to use as the find pins. These characters are trained as templates for the pins.

The AutoFindSearchArea box sets up the search regions of the find pins. This box can be moved and sized anywhere in the image, independently of the OCV Tool box. The size of the individual pin search areas is determined by comparing the OCV Tool box to the AutoFindSearchArea box. The position of the individual search areas is determined by the position of the AutoFindSearchArea box.

FIGURE 10–27. AutoFind Properties Page



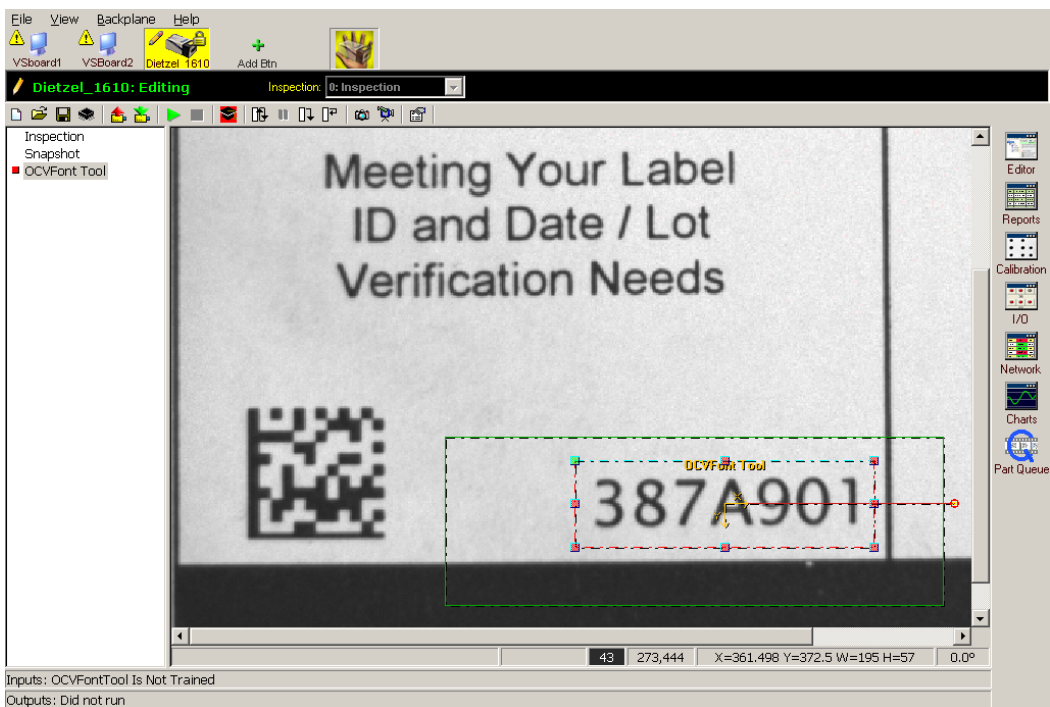
- **Registration Method** — Selects between a 1-Pin Find and a 2-Pin Find. For OCVFontlessTool, the default is 2-Pin. For OCVFontTool and OCVRuntimeTool, the default is 1-Pin. When set to 1-Pin, the locator will not handle any rotation of the characters being inspected. Switching between the registration methods requires re-training the OCV Tool so that the appropriate templates can be set up.

- **Template Method** — Sets the method for training the templates used by the Autofind. When set to “OCV Symbols As Templates”, the Autofind uses symbol positions from the OCV tool's trained layout to automatically train templates for the locator. When set to “User Defined Templates,” you must manually position and size the locator template and search boxes.

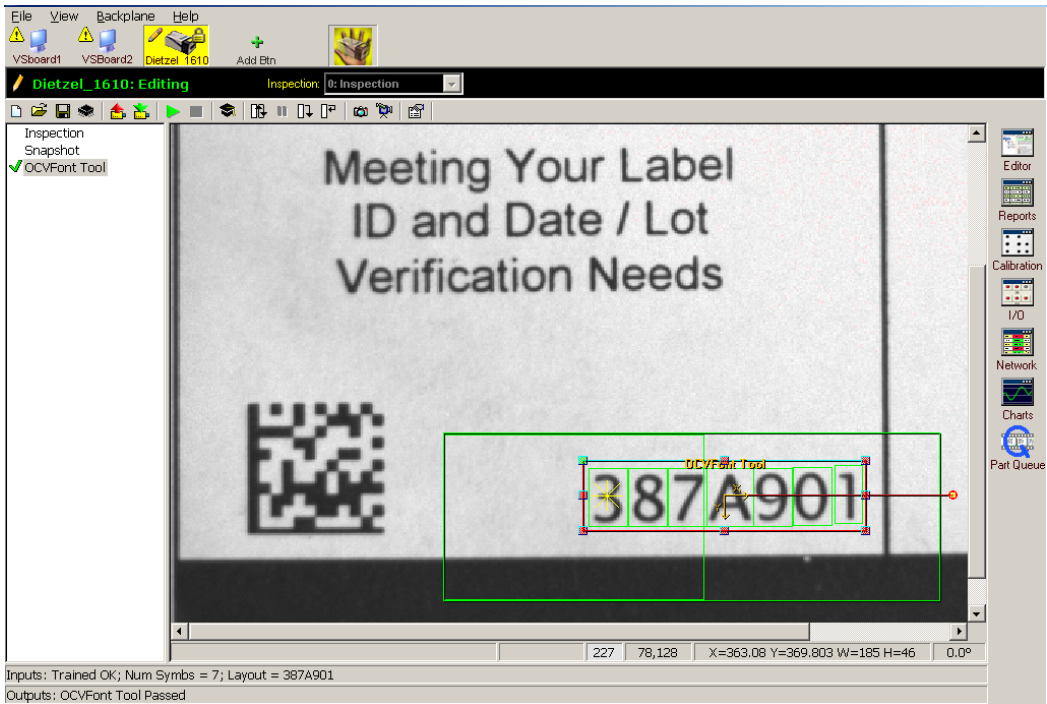
Default: “OCV Symbols As Templates”

To train the OCVFontTool, the tool shape must be placed around the characters that are going to be inspected.

FIGURE 10–28. Tool ROI Positioned Around Characters to be Inspected



Click **Editor** to display the OCVFontTool's datum page. Clicking on the “LayoutStep” tab in the datum page will display all properties for the LayoutStep. The correct font needs to be selected from the “Selected Font” datum's list of available fonts.

FIGURE 10–29. OCVFont Tool Training Completed

Clicking **Train** causes the tool to find all characters within the ROI that are trained as symbols in the selected font. The tool sets up its inspection “Layout” and is then ready to run. The inspected “layout” is shown in the input information bar under the image, as shown in Figure 10–30.

FIGURE 10–30. Inspected Layout Information

Inputs: Trained OK; Num Syms = 7; Layout = 3 8 7 A 9 0 1
 Outputs: OCVRuntimeTool Passed

Troubleshooting

In order for the AutoFind to find the code, the entire code must be contained within the AutoFind Search Area ROI. The AutoFind needs at least 3 pixels space between the code and the AutoFind Search Area ROI. This ensures that the code does not exceed the boundaries that are defined by this ROI. If the inspected code is close to the edge of the AutoFind Search Area ROI, try making the AutoFind Search Area larger to allow the code to be located properly.

In some cases, the AutoFind may have difficulty finding the code even though it is completely contained within the AutoFind Search Area ROI. In these cases, the find pin robustness can be adjusted to make the AutoFind search more thoroughly for the code. To adjust this setting in FrontRunner:

1. Click **Editor**.
2. Select the OnePt Locator or TwoPt Locator that is directly below the AutoFind step.
3. Select the Find Pin 1 tab and change the Robustness to a higher Refinement level.

Note: Maximum Refinement will be the most intensive search, but will increase the processing time.

4. Repeat this procedure for Find Pin 2 if the AutoFind is a TwoPt Locator.

OCVFontTool

This tool uses an OCVFont to learn the layout, which means that it determines which characters from the OCVFont are in which locations in the ROI. Once the layout is learned, the OCVFontTool expects to find these symbols at the same locations in the ROI during inspection. It uses the data from the FontSymbols in the OCVFont to verify the quality and correctness of the characters being inspected.

Training

Training of the OCVFontTool involves placing and sizing the OCVFontTool ROI around the area containing the symbols to be inspected. When Train is clicked, the ROI is scanned for symbol candidates. Symbol candidates are determined by searching for each symbol that is in the selected OCVFont, chosen through the LayoutStep.

Then, the OCVFontTool ROI is reset based on the bounding rectangle of all symbols found and the values of the search extra properties. The AutoFind is trained automatically whenever the OCVFontTool is trained. When the AutoFind Search Area Box is moved and/or sized, it is automatically re-trained, without requiring re-training of the OCVFontTool.

Inspection

If AutoFind is enabled, the pins are located and the OCVFontTool ROI is re-positioned based on the pin locations. Each symbol found during training is expected to be at the same location within the OCVFontTool ROI at runtime. For each symbol position, there are several ways that an inspection can fail. Here are some of the ways that the OCV tool can fail:

- The symbol cannot be located.
- The symbol can fail because the sharpness value is out of tolerance.
- The symbol can fail because the contrast value is out of tolerance.
- The symbol can fail because a break larger than the user-specified size appears in the character.
- The symbol failed an ID Test. It could not be determined that the correct symbol was present.
- The symbol can fail the initial residue check.

- The symbol can fail the final residue check, either or both methods. This residue analysis allows for detection of the following:
 - Symbol has become thicker or thinner
 - Symbol has holes or missing features
 - Symbol holes are filled in
 - Symbol contains additional or stray markings

FIGURE 10–31. OCVFont Tool Properties Page

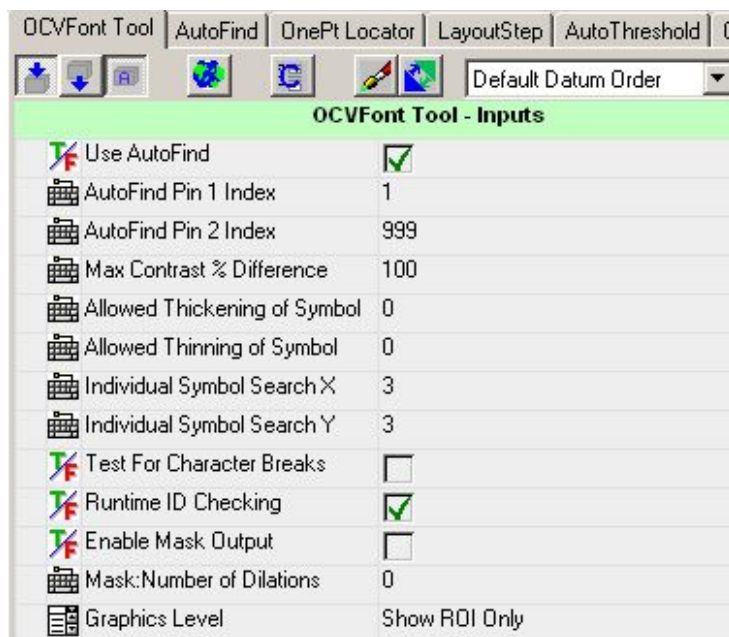


TABLE 10–3. Links to Property Descriptions

For Information About...	Go To...
Allow Thickening of Symbol	page 10–45
Allow Thinning of Symbol	page 10–45
AutoFind Pin 1 Index	page 10–45
AutoFind Pin 2 Index	page 10–46
Enable Mask Output	page 10–47

TABLE 10–3. Links to Property Descriptions

For Information About...	Go To...
Graphics Level	page 10–47
Individual Symbol Search X	page 10–46
Individual Symbol Search Y	page 10–46
Mask: Number of Dilations	page 10–47
Max Contrast % Difference	page 10–46
Runtime ID Checking	page 10–46
Test For Character Breaks	page 10–46
Use AutoFind	page 10–45

- **Use AutoFind** — Enables and disables the locator. By default, the AutoFind is enabled. Switching between enabled and disabled requires re-training the OCV Tool so that the appropriate templates can be set up.
- **Allowed Thickening of Symbol** — Determines the number of pixels that a symbol is allowed to grow along its perimeter. Residue will be ignored if it is found in the region between the edge of the symbol and the set number of pixels away from the edge, in the direction away from the center of the symbol.

Default: 0 pixels
Range: 0 to 10 pixels

- **Allowed Thinning of Symbol** — Determines the number of pixels that a symbol is allowed to shrink along its perimeter. Residue will be ignored if it is found in the region between the edge of the symbol and the set number of pixels away from the edge, in the direction toward the center of the symbol.

Default: 0 pixels
Range: 0 to 10 pixels

- **AutoFind Pin 1 Index** — Allows selection of the symbol position that trains the templates for AutoFind Pin 1. When this property is set to a value less than or equal to 1, the first symbol position is used. When this property is set to a value greater than or equal to the number of trained symbols, the last symbol position is used.

Default: 1, meaning use the first symbol
Range: 1 to n, where n is greater than or equal to the number of trained symbols

- **AutoFind Pin 2 Index** — Allows selection of the symbol position that trains the templates for AutoFind Pin 2 (when the AutoFind is set up as a 2PinFind). When this property is set to a value less than or equal to 1, the first symbol position is used. When this property is set to a value greater than or equal to the number of trained symbols, the last symbol position is used.

Default: 999 (use the last symbol)

Range: 1 to n, where n is greater than or equal to the number of trained symbols

- **Max Contrast % Difference** — Sets the maximum percentage difference between the calculated contrast values for symbols being inspected by the tool. When set to 100%, any contrast difference is acceptable. In order to use this functionality, the symbol contrast check must be performed. If no contrast calculations are performed for the inspected symbols, the calculated percent difference is 0. Otherwise, the smallest contrast from the inspected symbols is divided by the largest contrast from the inspected symbols. This value is then subtracted from 1 to get the percentage difference. If the calculated the difference is larger than the value of “Max Contrast % Difference”, the inspection fails.

Default: 100%

Range: 0 to 100%

- **Individual Symbol Search X** — Determines the width of the search area for individual symbols. This number is doubled and added to the symbol width to get the search width.

Default: 3 pixels

Range: 0 to 50 pixels

- **Individual Symbol Search Y** — Determines the height of the search area for individual symbols. This number is doubled and added to the symbol height to get the search height.

Default: 3 pixels

Range: 0 to 50 pixels

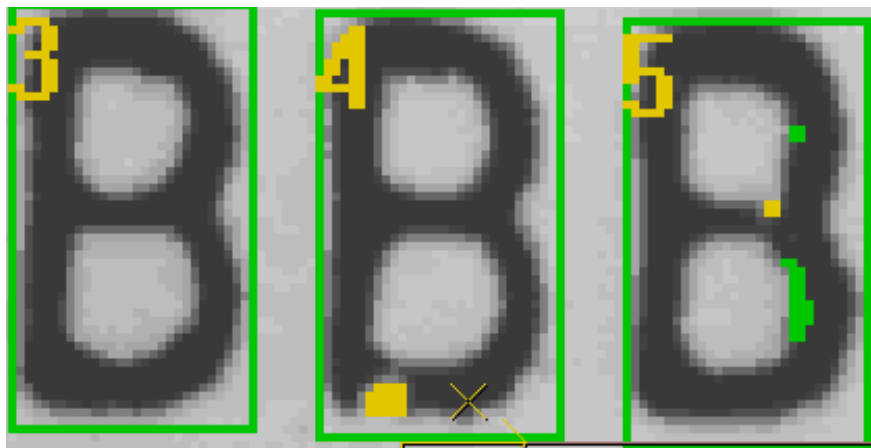
- **Test For Character Breaks** — Enables and disables the checks for character break appearance flaws. The default setting is disabled.
- **Runtime ID Checking** — Enables and disables the tests that determine if the correct symbol is present at runtime. The default setting is enabled.

During training of an OCVFont, the FontSymbols that are added are checked against each other to determine how similar they are. When FontSymbols are found to be very similar, tests for determining the presence of the correct symbol are set up and stored with the FontSymbols. These tests are only performed at runtime when Runtime ID Checking is enabled.

Note: We highly recommend that you do not use Automatic Segmentation, that is, leave its setting in its default position of off, and carefully use symbol boxes of equal size for all special characters like O, 0, B, 8, D, and so on.

- **Enable Mask Output** — Enables and disables (default) the creation and output of a mask at runtime. This property is used in conjunction with the DynamMask Tool to allow the printed characters to be excluded (masked out) from other image processing. Enabling this property increases inspection time.
- **Mask: Number of Dilations** — Sets the number of expansions that are performed on the output mask. The default value is 1. This property is used in conjunction with the DynamMask Tool to allow the printed characters to be excluded (masked out) from other image processing.
- **Graphics Level** — Sets up different levels of debug graphics at runtime. The default Show ROI Only will only show the ROI boxes associated with the OCVFontTool and the characters being inspected (green for passed, red for failed). When set to Show None, no graphics are shown at runtime. When set to Show Basic Graphics, a number indicating the symbol's position in the layout is shown, along with the ROI boxes.

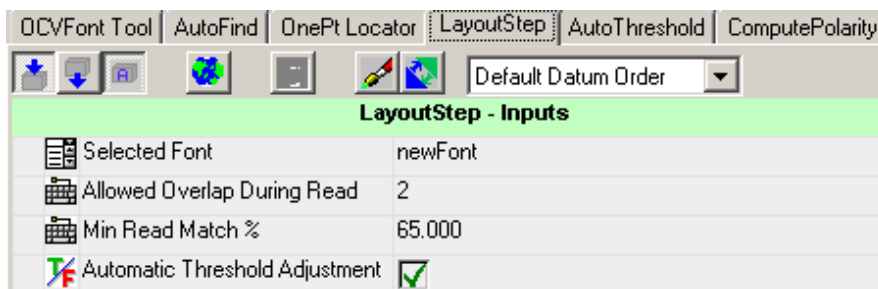
FIGURE 10–32. Graphics Level — Example



When set to Show Details, residue graphics are displayed: green pixels are those that were not there at train time but are in the image at runtime (fills), while yellow pixels are those that were there at train time but are not there at runtime (voids).

The LayoutStep for the OCVFont Tool selects an OCVFont and sets up the learn layout process.

FIGURE 10–33. LayoutStep Properties Page



- **Selected Font** — Allows selection of an OCVFont to use for training and inspections. Clicking on the current selected font will display a drop-down list containing the names of all OCVFonts that are in the Vscope\Jobs\Fonts folder.

- **Allowed Overlap During Read** — Used during the learn layout process. The value of this property specifies the amount of symbol candidate ROI overlap that is allowed. When symbol candidates overlap more than the allowed value, tests are performed to determine the best candidate at the overlap position. The other candidate will not become part of the layout. This overlap measurement is in pixels.

Default: 2 pixels

Range: 0 to 15 pixels

- **Min Read Match %** — Is a correlation percentage used as a minimum requirement for a symbol to be considered a candidate during the learn layout process.

Default: 65%

Range: 0% to 100%

Note: When characters are not being read during Learn Layout, decrease this property to 60%. Avoid settings below 55%.

- **Automatic Threshold Adjustment** — Enables/disables the automatic threshold adjustment feature. When enabled, the best match location during the learn layout process calculates an adjustment to the threshold used to create binary images at runtime. This calculated value is set in the **AutoThreshold Threshold Adjustment** property.

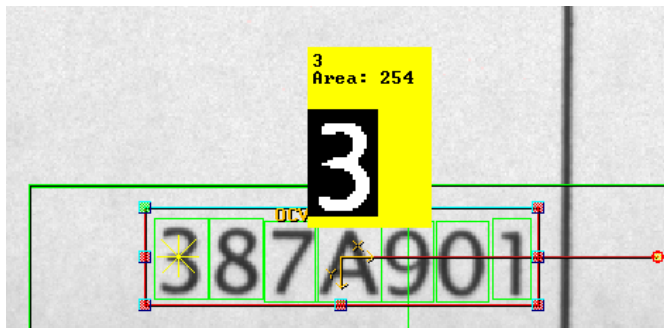
The **AutoThreshold** step of the **OCVFontTool** belongs to the **LayoutStep** and is used only at runtime. The only property used is **Threshold Adjustment**, which serves as a global adjustment for all **FontSymbols** being inspected. **FontSymbols** may still make individual adjustments to the thresholds using their own **Auto Threshold Adjustment** properties.

The **ComputePolarity** step of the **OCVFontTool** belongs to the **LayoutStep**. It is not used by an **OCVFontTool**.

Step Tip

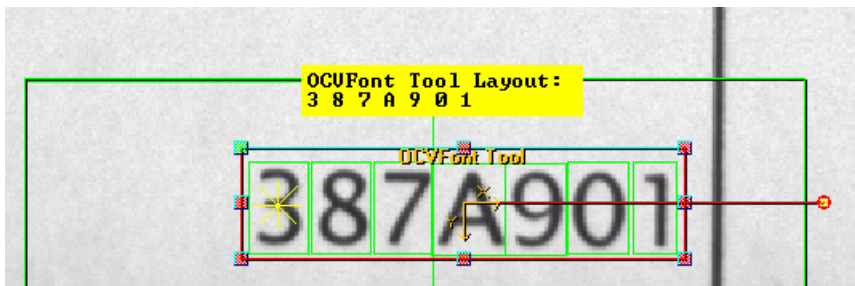
After the **OCVFont Tool** has been trained, positioning the mouse over the ROI displays a **Step Tip**. This **Step Tip** provides information and graphical feedback for individual symbols when the mouse is over a symbol area. Train information includes the Area of the symbol, the number of On pixels in the binary template, and a bitmap representation of the binary template, as shown in Figure 10–34.

FIGURE 10–34. Step Tip — Example 1



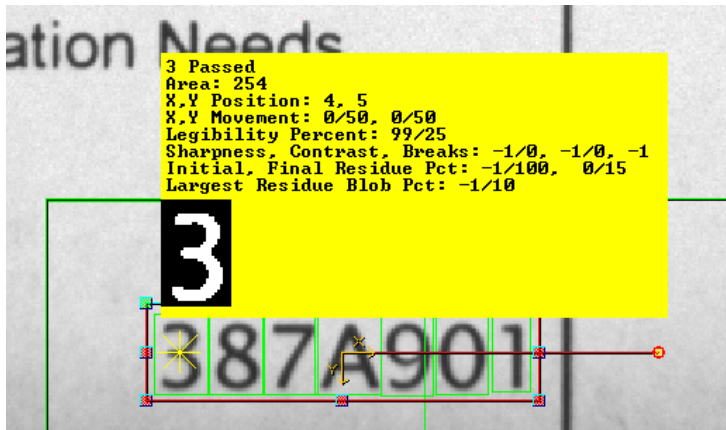
When the mouse is not positioned over a particular symbol area, the Step Tip displays the currently trained Layout Characters, or just the name of the OCVFont Tool when it is not trained, as shown in Figure 10–35.

FIGURE 10–35. Step Tip — Example 2



When the OCVFont Tool has been run doing a Tryout, additional runtime information is available by holding down the Shift key when the mouse is positioned over the symbol area, as shown in Figure 10–36.

FIGURE 10–36. Step Tip — Example 3



Inspection information includes:

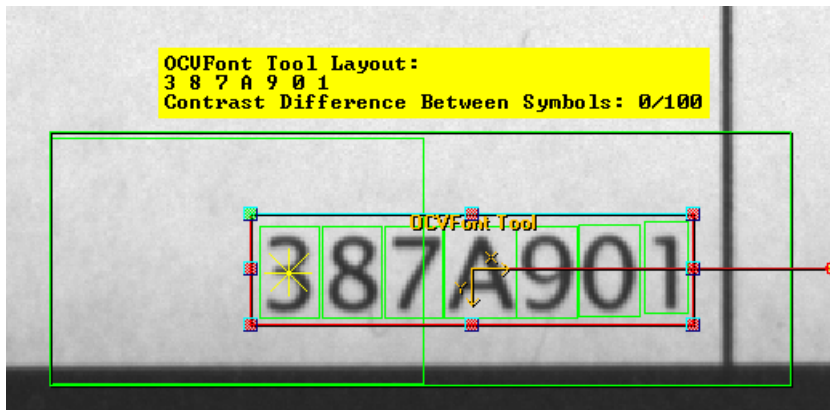
- The Area of the symbol (the number of On pixels in the trained binary template)
- The X and Y position (upper left corner) of the symbol relative to the OCVFontTool shape
- The X and Y allowed movement of the symbol
- The Legibility Percentage and the Legibility Tolerance
- The Sharpness, Contrast and number of Breaks found along with the associated tolerances
- The Initial and Final Residue percentages along with the associated tolerances
- The Largest (Final) Residue Blob Percentage and its associated tolerance

Note: A -1 for any value above, except the X and Y allowed movement, indicates the test is disabled.

- The bitmap representation of the binary runtime symbol area

When the OCVFont Tool has been run doing a Tryout, additional runtime information is available by holding down the Shift key and moving the mouse inside the OCVFont Tool ROI (but not over the symbol area), as shown in Figure 10–37.

FIGURE 10–37. Step Tip — Example 4



OCVRuntimeTool

This tool uses an OCVFont (called the Master Font) to learn the layout, which means that it determines which characters from the OCVFont are in which locations in the ROI. Once the layout is learned, the OCVRuntimeTool creates a new OCVFont (called a Runtime Font) by training a new FontSymbol at each layout position, using the current image data. The OCVRuntimeTool expects to find the symbols at the same locations during inspection. It uses the data from the Runtime Font to verify the quality and correctness of the characters being inspected. Because the train image creates templates, this code should be of good quality.

Training

Training of the OCVRuntimeTool involves placing and sizing the OCVRuntimeTool ROI around the area containing the symbols to be inspected. When **Train** is clicked, the ROI is scanned for symbol candidates. Symbol candidates are determined by searching for each symbol that is in the selected Master OCVFont (chosen through the LayoutStep). When all candidates have been found, a new OCVFont is created and a new symbol is trained and added to this Runtime Font for each candidate position.

Then, the OCVRuntimeTool ROI is reset based on the bounding rectangle of all symbols found and the values of the search extra properties. The AutoFind is trained automatically whenever the OCVRuntimeTool is trained. When the AutoFind Search Area ROI is moved and/or sized, it is automatically re-trained, without requiring re-training of the OCVRuntimeTool.

Inspection

If AutoFind is enabled, the pins are located and the OCVRuntimeTool ROI is repositioned based on the pin locations. Each of the symbols found during training is expected to be at the same location within the OCVRuntimeTool ROI at runtime. For each symbol position, there are several ways that an inspection can fail. Some of the failure modes are listed below:

- The symbol cannot be located.
- The symbol can fail because the sharpness value is out of tolerance.
- The symbol can fail because the contrast value is out of tolerance.
- The symbol can fail because a break larger than the user-specified size appears in the character.

- The symbol failed an ID Test. It could not be determined that the correct symbol was present.
- The symbol can fail the initial residue check.
- The symbol can fail the final residue check, either or both methods. This residue analysis allows for detection of the following:
 - Symbol has become thicker or thinner
 - Symbol has holes or missing features
 - Symbol holes are filled in
 - Symbol contains additional or stray markings

FIGURE 10–38. OCVRuntimeTool Properties Page

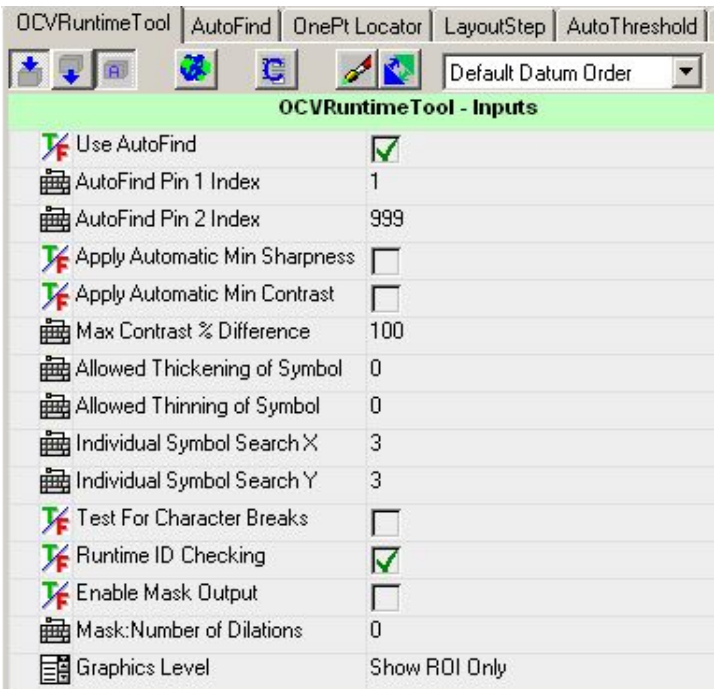


TABLE 10–4. Links to Property Descriptions

For Information About...	Go To...
Allow Thickening of Symbol	page 10–57
Allow Thinning of Symbol	page 10–57
Apply Automatic Min Contrast	page 10–56
Apply Automatic Min Sharpness	page 10–56
AutoFind Pin 1 Index	page 10–56
AutoFind Pin 2 Index	page 10–56
Enable Mask Output	page 10–58
Graphics Level	page 10–58
Individual Symbol Search X	page 10–57
Individual Symbol Search Y	page 10–57
Mask: Number of Dilations	page 10–58

TABLE 10–4. Links to Property Descriptions

For Information About...	Go To...
Max Contrast % Difference	page 10–57
Runtime ID Checking	page 10–58
Test For Character Breaks	page 10–58
Use AutoFind	page 10–56

- **Use AutoFind** — Enables (default) and disables the locator. Switching between enabled and disabled requires re-training the OCV Tool so that the appropriate templates can be set up.
- **AutoFind Pin 1 Index** — Allows selection of the symbol position that trains the templates for AutoFind Pin 1. When set to a value less than or equal to 1, the first symbol position is used. When set to a value greater than or equal to the number of trained symbols, the last symbol position is used.

Default: 1 (use the first symbol)

Range: 1 to n, where n is greater than or equal to the number of trained symbols

- **AutoFind Pin 2 Index** — Allows selection of the symbol position that trains the templates for AutoFind Pin 2 (when the AutoFind is set up as a 2PinFind). When set to a value less than or equal to 1, the first symbol position is used. When set to a value greater than or equal to the number of trained symbols, the last symbol position is used.

Default: 999 (use the last symbol)

Range: 1 to n, where n is greater than or equal to the number of trained symbols

- **Apply Automatic Min Sharpness** — Allows the FontSymbols trained and added to the Runtime Font to have a sharpness tolerance automatically calculated for them. This automatically calculated tolerance is equal to 65% of the sharpness value calculated using the trained template. The default setting is disabled.
- **Apply Automatic Min Contrast** — Allows the FontSymbols trained and added to the Runtime Font to have a contrast tolerance automatically calculated for them. This automatically calculated tolerance is equal to 50% of the contrast value calculated using the trained template. The default setting is disabled.

- **Max Contrast % Difference** — Sets the maximum percentage difference between the calculated contrast values for symbols being inspected by the tool. When set to 100%, any contrast difference is acceptable. In order to use this functionality, the symbol contrast check must be performed. If no contrast calculations are performed for the inspected symbols, the calculated percent difference is 0. Otherwise, the smallest contrast from the inspected symbols is divided by the largest contrast from the inspected symbols. This value is then subtracted from 1 to get the percentage difference. If the calculated the difference is larger than the value of “Max Contrast % Difference”, the inspection fails.

Default: 100%

Range: 0 to 100%

- **Allowed Thickening of Symbol** — Determines the number of pixels that a symbol is allowed to grow along its perimeter. Residue will be ignored if it is found in the region between the edge of the symbol and the set number of pixels away from the edge, in the direction away from the center of the symbol.

Default: 0 pixels

Range: 0 to 10 pixels

- **Allowed Thinning of Symbol** — Determines the number of pixels that a symbol is allowed to shrink along its perimeter. Residue will be ignored if it is found in the region between the edge of the symbol and the set number of pixels away from the edge, in the direction toward the center of the symbol.

Default: 0 pixels

Range: 0 to 10 pixels

- **Individual Symbol Search X** — Determines the width of the search area for individual symbols. This number is doubled and added to the symbol width to get the search width.

Default: 3 pixels

Range: 0 to 50 pixels

- **Individual Symbol Search Y** — Determines the height of the search area for individual symbols. This number is doubled and added to the symbol height to get the search height.

Default: 3 pixels

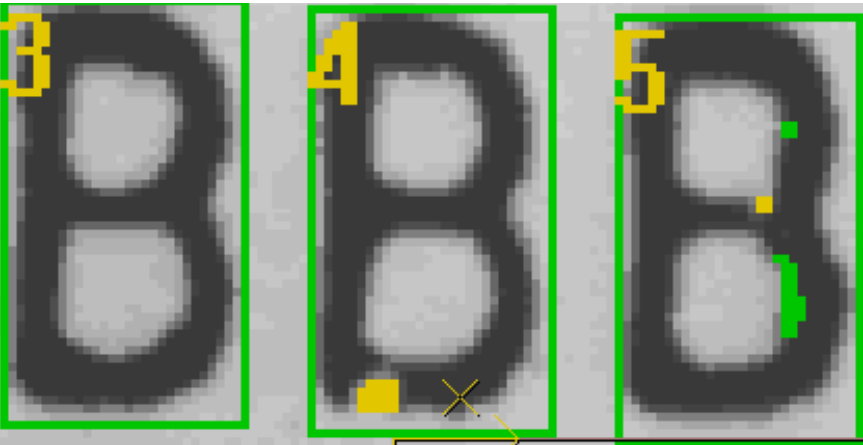
Range: 0 to 50 pixels

- **Test For Character Breaks** — Enables and disables the checks for character break appearance flaws. The default setting is disabled.
- **Runtime ID Checking** — Enables (default) and disables the tests that determine if the correct symbol is present at runtime. During training of an OCVFont, the FontSymbols that are added are checked against each other to determine how similar they are. When FontSymbols are found to be very similar, tests for determining the presence of the correct symbol are set up and stored with the FontSymbols. These tests are only performed at runtime when Runtime ID Checking is enabled.

Note: We highly recommend that you do not use Automatic Segmentation, that is, leave its setting in its default position of off, and carefully use symbol boxes of equal size for all special characters like O, 0, B, 8, D, and so on.

- **Enable Mask Output** — Enables and disables the creation and output of a mask at runtime. The default is disabled. This property is used in conjunction with the DynamMask Tool to allow the printed characters to be excluded (masked out) from other image processing. Enabling this property increases inspection time.
- **Mask:Number of Dilations** — Sets the number of expansions that are performed on the output mask. The default value is 1. This property is used in conjunction with the DynamMask Tool to allow the printed characters to be excluded (masked out) from other image processing.
- **Graphics Level** — Sets up different levels of debug graphics at runtime. The default Show ROI Only will only show the ROI boxes associated with the OCVRuntimeTool and the characters being inspected (green for passed, red for failed). When set to Show None, no graphics are shown at runtime. When set to Show Basic Graphics, a number indicating the symbol's position in the layout is shown, along with the ROI boxes, as shown in Figure 10–39.

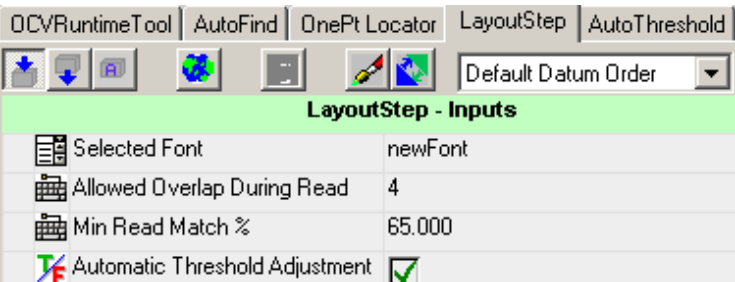
FIGURE 10–39. Graphics Level — Example



When set to Show Details, residue graphics are displayed: green pixels are those that were not there at train time but are in the image at runtime (fills), while yellow pixels are those that were there at train time but are not there at runtime (voids).

The **LayoutStep** for the OCVRuntimeTool selects a Master OCVFont and set up the learn layout process.

FIGURE 10–40. LayoutStep Properties Page



- **Selected Font** — Allows selection of an OCVFont to use for training and inspections. Clicking on the current selected font will display a drop-down list containing the names of all OCVFonts that are in the Vscape\Jobs\Fonts folder.

- **Allowed Overlap During Read** — Used during the learn layout process. The value of this property specifies the amount of symbol candidate ROI overlap that is allowed. When symbol candidates overlap more than the allowed value, tests are performed to determine the best candidate at the overlap position. The other candidate will not become part of the layout. This overlap measurement is in pixels.

Default: 2 pixels

Range: 0 to 15 pixels

- **Min Read Match %** — Is a correlation percentage used as a minimum requirement for a symbol to be considered a candidate during the learn layout process.

Default: 65%

Range: 0% to 100%

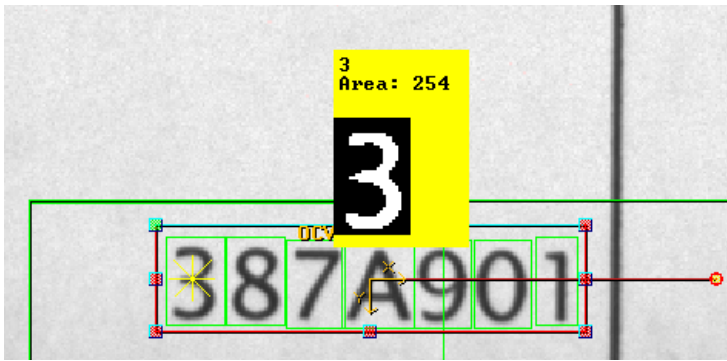
- **Automatic Threshold Adjustment** — Enables/disables the automatic threshold adjustment feature. When enabled, the best match location during the learn layout process calculates an adjustment to the threshold used to create binary images at runtime. This calculated value is set in the AutoThreshold step's Threshold Adjustment property.

The AutoThreshold step of the OCVRuntimeTool belongs to the LayoutStep and is used only at runtime. The only property used is the Threshold Adjustment property, which serves as a global adjustment for all FontSymbols being inspected. FontSymbols may still make individual adjustments to the thresholds using their own **Auto Threshold Adjustment** properties.

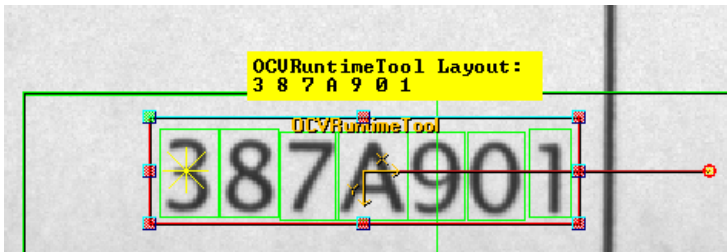
The ComputePolarity step of the OCVRuntimeTool belongs to the LayoutStep. It is not used by an OCVFontTool.

Step Tips

After the OCVRuntimeTool has been trained, positioning the mouse over the ROI displays a Step Tip. This Step Tip provides information and graphical feedback for individual symbols when the mouse is over a symbol area. Train information includes the Area of the symbol, the number of On pixels in the binary template, and a bitmap representation of the binary template, as shown in Figure 10-41.

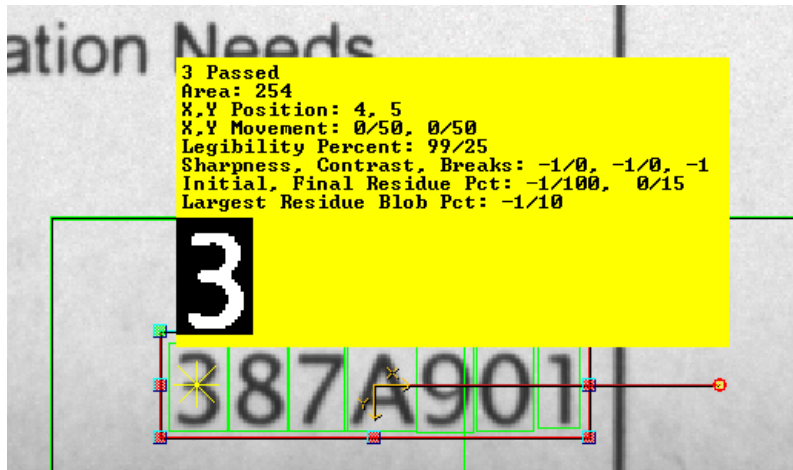
FIGURE 10–41. Step Tip — Example 1

When the mouse is not positioned over a particular symbol area, the Step Tip displays the currently trained Layout Characters, or just the name of the OCVRuntime Tool when it is not trained, as shown in Figure 10–42.

FIGURE 10–42. Step Tip — Example 2

When the OCVRuntimeTool has been run using Tryout, additional runtime information is available by holding down the Shift key when the mouse is positioned over the symbol area, as shown in Figure 10–43.

FIGURE 10–43. Step Tip — Example 3



Inspection information includes:

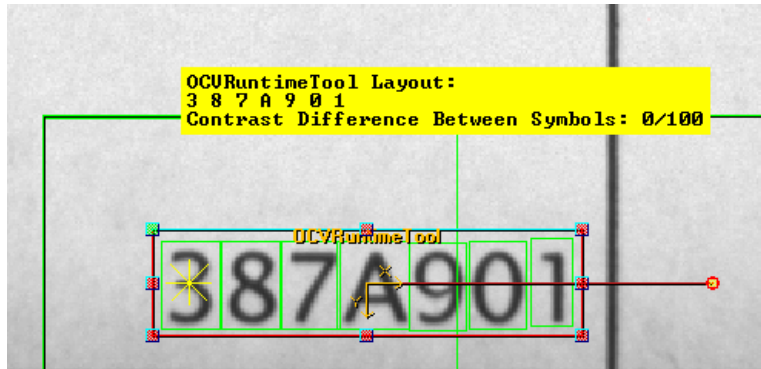
- The Area of the symbol, the number of On pixels in the trained binary template
- The X and Y position, upper left corner, of the symbol relative to the OCVRuntimeTool shape
- The X and Y allowed movement of the symbol
- The Legibility Percentage and the Legibility Tolerance
- The Sharpness, Contrast and number of Breaks found along with the associated tolerances
- The Initial and Final Residue percentages along with the associated tolerances
- The Largest (Final) Residue Blob Percentage and its associated tolerance

Note: A -1 for any value above, except the X and Y allowed movement, indicates the test is disabled.

- The bitmap representation of the binary runtime symbol area

When the OCVRuntime Tool has been run doing a Tryout, additional runtime information is available by holding down the Shift key and moving the mouse inside the OCVRuntime Tool ROI (but not over the symbols), as shown in Figure 10–44.

FIGURE 10–44. Step Tip — Example 4



OCVFontless Tool

This tool does not require an OCVFont. Instead, it determines the location of characters in the FOV using a blob-analysis technique. Then, it stores training data for each character location as an OCVSymbolStep. The OCVFontless Tool expects to find the symbols at the same locations during inspection. It uses the trained data to verify the quality of the characters being inspected.

Training

Training the OCVFontless Tool involves placing and sizing the OCVFontless Tool ROI around the area containing the symbols to be inspected. When **Train** is clicked, the ROI is scanned for symbol candidates. A symbol candidate is a group of connected pixels that have foreground polarity. Each symbol candidate that contains enough pixels, as defined by the **Min Symbol Size in pixels** parameter, is trained and stored as an OCVSymbolStep.

Then, the OCVFontless Tool ROI is reset based on the bounding rectangle of all symbols found and the values of the search extra properties. The **AutoFind** is trained automatically whenever the OCVFontless Tool is trained. When the **AutoFind Search Area ROI** is moved and/or sized, it is automatically re-trained, without requiring re-training of the OCVFontless Tool.

Inspection

When **AutoFind** is enabled, the pins are located and the OCVFontless Tool ROI is re-positioned based on the pin locations. Each symbol found during training is expected to be at the same location within the OCVFontless Tool ROI at runtime. For each symbol position, there are several ways that an inspection can fail. Some of the failure modes are listed below:

- The symbol cannot be located.
- The symbol can fail because the sharpness value is out of tolerance.
- The symbol can fail because the contrast value is out of tolerance.
- The symbol can fail because a break larger than the user-specified size appears in the character.
- The symbol can fail the initial residue check.
- The symbol can fail the final residue check, either or both methods. This residue analysis allows for detection of the following:
 - Symbol has become thicker or thinner
 - Symbol has holes or missing features
 - Symbol holes are filled in
 - Symbol contains additional or stray markings

FIGURE 10–45. OCVFontless Tool Properties Page

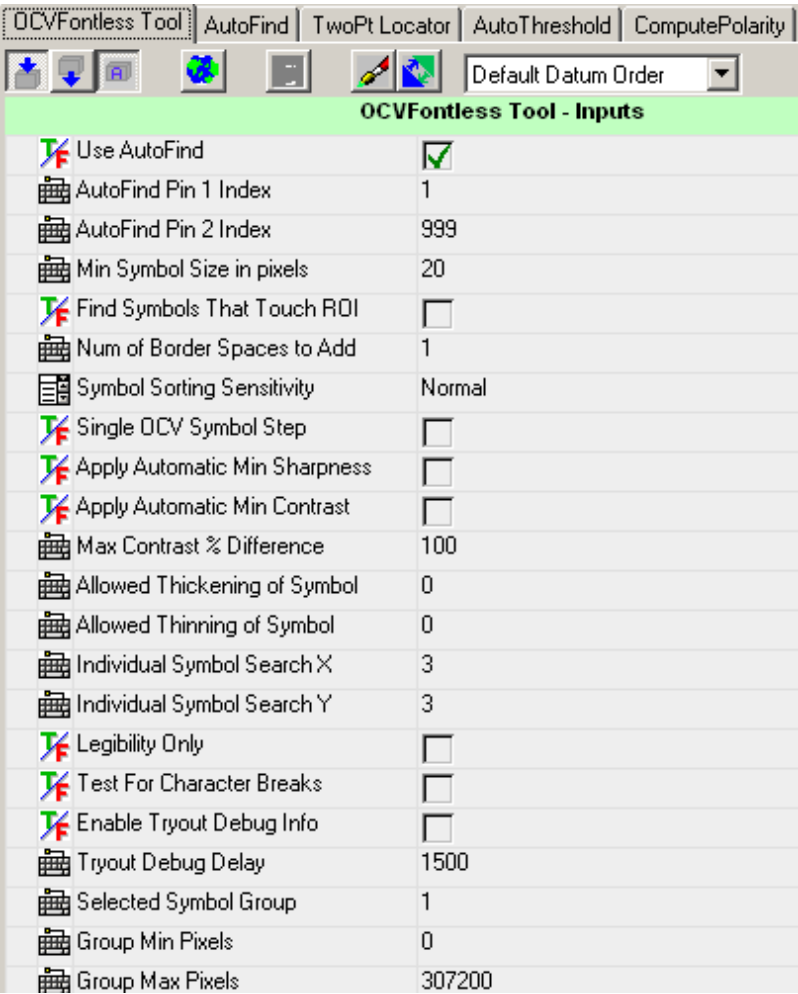


TABLE 10–5. Links to Property Descriptions

For Information About...	Go To...
Allowed Thickening of Symbol	page 10–68
Allowed Thinning of Symbol	page 10–68
Apply Automatic Min Contrast	page 10–67
Apply Automatic Min Sharpness	page 10–67
AutoFind Pin 1	page 10–66

TABLE 10–5. Links to Property Descriptions

For Information About...	Go To...
AutoFind Pin 2	page 10–66
Enable Tryout Debug Info	page 10–69
Find Symbols That Touch ROI	page 10–67
Individual Symbol Search X	page 10–68
Individual Symbol Search Y	page 10–68
Legibility Only	page 10–69
Max Contrast % Difference	page 10–68
Min Symbol Size in pixels	page 10–67
Num of Border Spaces to Add	page 10–67
Single OCV Symbol Step	page 10–67
Symbol Sorting Sensitivity	page 10–67
Test For Character Breaks	page 10–69
Tryout Debug Delay	page 10–69
Use AutoFind	page 10–66

- **Use AutoFind** — Enables and disables the locator. By default, the AutoFind is enabled. Switching between enabled and disabled requires re-training the OCV Tool so that the appropriate templates can be set up.
- **AutoFind Pin 1 Index** — Allows selection of the symbol position that trains the templates for AutoFind Pin 1. When set to a value less than or equal to 1, the first symbol position is used. When set to a value greater than or equal to the number of trained symbols, the last symbol position is used.

Default: 1, meaning use the first symbol

Range: 1 to n, where n is greater than or equal to the number of trained symbols

- **AutoFind Pin 2 Index** — Allows selection of the symbol position that trains the templates for AutoFind Pin 2, when the AutoFind is set up as a 2PinFind. When set to a value less than or equal to 1, the first symbol position is used. When set to a value greater than or equal to the number of trained symbols, the last symbol position is used.

Default: 999, meaning use the last symbol

Range: 1 to n, where n is greater than or equal to the number of trained symbols

- **Min Symbol Size in pixels** — Adjusts the minimum size that a blob must be in order to be considered a symbol.

Default: 20 pixels

Minimum Value: 5 pixels

- **Find Symbols That Touch ROI** — When enabled, symbols are trained for blobs that are not fully within the ROI. When disabled, blobs that are not fully within the ROI are ignored. The default setting is disabled.
- **Num of Border Spaces to Add** — Determines how many pixels to allow between actual character pixels and the edge of the box that defines the OCVSymbolStep.

Default: 1 pixels

Range: 0 to 19 pixels

- **Symbol Sorting Sensitivity** — Adjusts the sensitivity of the sorting of symbols into rows during training. The sorting is based on the positions of the symbols. When set to Normal, any two symbols whose top Y positions are separated by more than 50 percent of the average height of all trained symbols will be considered to be on separate rows. When set to Low, the allowed separation is increased to 75 percent of the average height. When set to Lowest, the allowed separation is increased to 90 percent of the average height. When set to High, the allowed separation is decreased to 25 percent of the average height. When set to Highest, the allowed separation is decreased to 10 percent of the average height.

Default: Normal

- **Single OCV Symbol Step** — When set to Enabled, the OCVFontlessTool trains a single OCVSymbolStep that includes all the symbols of the mark that are within the mark area box. When set to Disabled, the OCVFontlessTool trains an OCVSymbolStep for each symbol of the mark that is located within the mark area box. The default setting is disabled.
- **Apply Automatic Min Sharpness** — Allows the OCVSymbolSteps trained to have a sharpness tolerance automatically calculated for them. This automatically calculated tolerance is equal to 65% of the sharpness value calculated using the trained template. The default setting is disabled.
- **Apply Automatic Min Contrast** — Allows the OCVSymbolSteps trained to have a contrast tolerance automatically calculated for them. This automatically calculated tolerance is equal to 50% of the contrast value calculated using the trained template. The default setting is disabled.

- **Max Contrast % Difference** — Sets the maximum percentage difference between the calculated contrast values for symbols being inspected by the tool. When set to 100%, any contrast difference is acceptable. In order to use this functionality, the symbol contrast check must be performed. If no contrast calculations are performed for the inspected symbols, the calculated percent difference is 0. Otherwise, the smallest contrast from the inspected symbols is divided by the largest contrast from the inspected symbols. This value is then subtracted from 1 to get the percentage difference. If the calculated the difference is larger than the value of “Max Contrast % Difference”, the inspection fails.

Default: 100%

Range: 0 to 100%

- **Allowed Thickening of Symbol** — Determines the number of pixels that a symbol is allowed to grow along its perimeter. Residue will be ignored if it is found in the region between the edge of the symbol and the set number of pixels away from the edge, in the direction away from the center of the symbol.

Default: 0 pixels

Range: 0 to 10 pixels

- **Allowed Thinning of Symbol** — Determines the number of pixels that a symbol is allowed to shrink along its perimeter. Residue will be ignored if it is found in the region between the edge of the symbol and the set number of pixels away from the edge, in the direction toward the center of the symbol.

Default: 0 pixels

Range: 0 to 10 pixels

- **Individual Symbol Search X** — Determines the width of the search area for individual symbols. This number is doubled and added to the symbol width to get the search width.

Default: 3 pixels

Range: 0 to 50 pixels

- **Individual Symbol Search Y** — Determines the height of the search area for individual symbols. This number is doubled and added to the symbol height to get the search height.









Default: 3 pixels

Range: 0 to 50 pixels



























- **Legibility Only** — Enables and disables (default) Legibility-Only inspections. When enabled, the symbols will pass/fail based only on the symbol's **Legibility (%)** parameter.
- **Test For Character Breaks** — Enables and disables the checks for character break appearance flaws. The default setting is disabled.
- **Enable Tryout Debug Info** — Enables and disables the display of debug information during tryout. The default setting is disabled. When enabled, inspection data is displayed on the Output line for each OCVSymbol that is part of the OCVFontlessTool.
- **Tryout Debug Delay** — Sets a delay that is used during the display of debug information. The default delay is 1500ms. This delay is the minimum amount of time that the information is displayed.

FIGURE 10–46. Symbol Group Settings

OCVFontless Tool | AutoFind | TwoPt Locator | AutoThreshold | ComputePolarity








Default Datum Order 

OCVFontless Tool - Inputs

	Selected Symbol Group	1
	Group Min Pixels	0
	Group Max Pixels	307200
	Group Legibility (%)	25.000
	Group Allowed Movement in X (+/-)	50
	Group Allowed Movement in Y (+/-)	50
	Group Residue Limit Units	Percentage
	Group Initial Res Limit	100.000
	Group Final Res Method	Total Residue Area
	Group Final Res Limit	15.000
	Group Final Res Max Blob Size	10
	Group Max Flaw Size	1
	Group Appear Flaw Break Test	<input checked="" type="checkbox"/>
	Group Appear Flaw Break Size	2
	Group Sharpness Limit Units	Gray Level
	Group Min Allowed Sharpness	0
	Group Contrast Limit Units	Gray Level
	Group Min Allowed Contrast	0
	Group Auto Threshold Enabled	<input checked="" type="checkbox"/>
	Group Auto Threshold Adjust	0
	Group Manual Threshold	135
	Group Edge Energy Threshold	20
	Group Character Expansions	0
	Group Filter Bright Defects	<input type="checkbox"/>
	Group Bright Defect % Range	0
	Apply Symbol Group Settings	<click to execute>

- **Selected Symbol Group** — Selects which symbol group has its symbol properties displayed on the properties page. Changing this value updates the property page to display the correct symbol group properties.

Symbol groups are defined based on the number of On pixels in the templates of the symbols. The grouping of symbols is accomplished by setting a range of values, Group Min Pixels and Group Max Pixels, for a group. By default, only one symbol group is defined. This group contains all OCVSymbolSteps because the range is automatically set to a minimum of 0 pixels and a maximum of 307200, maximum possible in a 640x480 symbol.

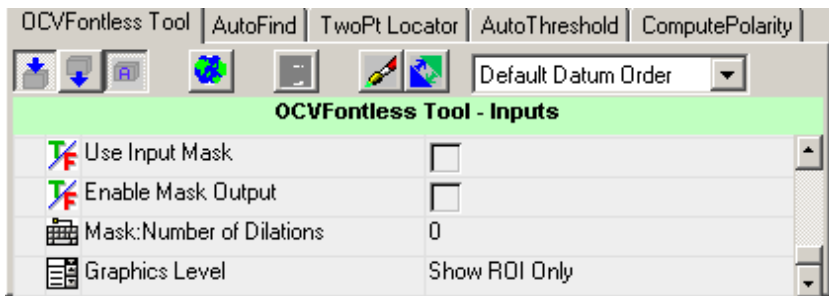
The groups are defined in order of maximum character pixels, so that group 2's Group Min Pixels property is always equal to group 1's Group Max Pixels property plus one, group 3's Group Min Pixels property is always equal to group 2's Group Max Pixels property plus one, etc. The final group is always defined to have a Group Max Pixels equal to 307200.

To add a group, change Group Max Pixels of the current group from 307200 to a lower value.

The following properties on the OCVFontlessTool property page can be set on a group basis. Refer to “OCVSymbolStep” on page 10-79 for more information.

- Group Legibility (%)
- Group Allowed Movement in X
- Group Allowed Movement in Y
- Group Residue Limit Units
- Group Initial Res Limit
- Group Final Res Method
- Group Final Res Limit
- Group Final Res Max Blob Size
- Group Max Flaw Size
- Group Appear. Flaw Break Test
- Group Appear. Flaw Break Size
- Group Sharpness Limit Units

- Group Min Allowed Sharpness
- Group Contrast Limit Units
- Group Min Allowed Contrast
- Group Auto Threshold Enabled
- Group Auto Threshold Adjust
- Group Manual Threshold
- Group Edge Energy Threshold
- Group Character Expansions
- Group Filter Bright Defects
- Group Bright Defect % Range
- **Apply Symbol Group Settings** — Sets the group parameters for the currently selected group to the current values in the group properties.

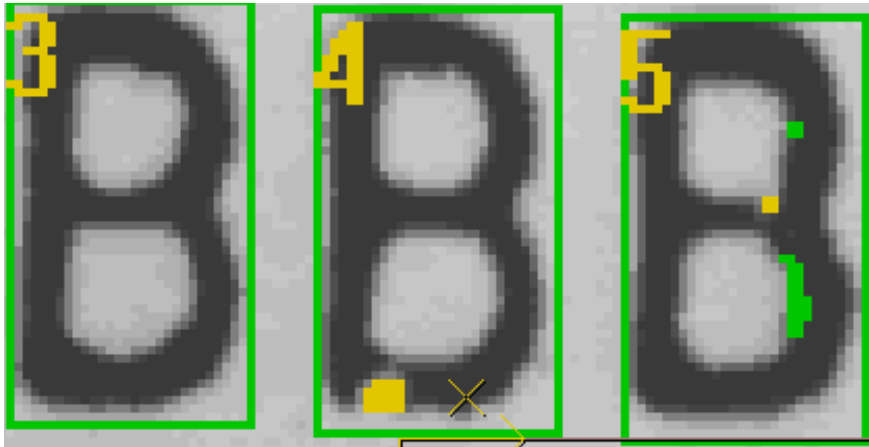
FIGURE 10–47. Mask Settings

- **Use Input Mask** — This property is applicable only when the OCVFontlessTool has a child step that produces a mask buffer as an output. When such a child step is inserted into the OCVFontlessTool, Use Input Mask is automatically enabled. When enabled, the OCVFontlessTool must be trained. After it is trained, the mask pixels are highlighted in red. When disabled, this property will allow the OCVFontlessTool to retain the child mask-generating step but does not apply the mask at runtime.

Default: disabled

- **Enable Mask Output** — Enables and disables the creation and output of a mask at runtime. The default setting is disabled. This property is used in conjunction with the DynamMask Tool to allow the printed characters to be excluded, masked out, from other image processing.
- **Mask: Number of Dilations** — Sets the number of expansions that are performed on the output mask. The default value is 1. This property is used in conjunction with the DynamMask Tool to allow the printed characters to be excluded, masked out, from other image processing.
- **Graphics Level** — Sets up different levels of debug graphics at runtime. The default Show ROI Only only shows the ROI boxes associated with the OCVFontlessTool and the characters being inspected, green for passed, red for failed. When set to Show None, no graphics are shown at runtime. When set to Show Basic Graphics, a number indicating the symbol's position in the layout is shown, along with the ROI boxes, as shown in Figure 10–48.

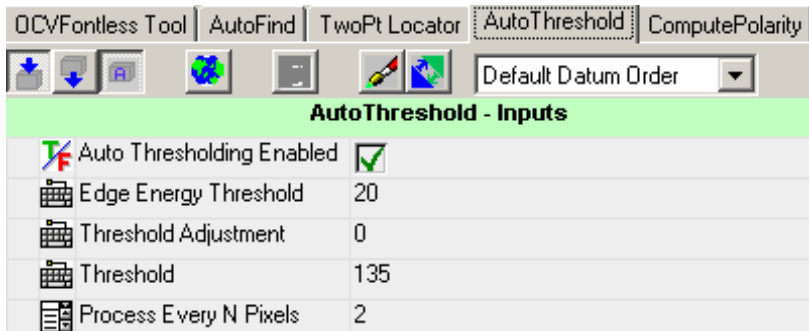
FIGURE 10–48. Graphics Level — Example



When set to Show Details, residue graphics are displayed: green pixels are those that were not there at train time but are in the image at runtime (fills), while yellow pixels are those that were there at train time but are not there at runtime (voids). Show Details also displays the blob outline of the characters at train time.

The AutoThreshold properties page of the OCVFontlessTool is used for segmentation of the image during training, as shown in Figure 10–49.

FIGURE 10–49. AutoThreshold Properties Page



- **Auto Thresholding Enabled** — Enables and disables the automatic thresholding. When enabled, a threshold is calculated using the ROI of the step. This calculation uses edge detection to determine foreground and background information. The calculated threshold is displayed in **Threshold**. Although it is called **AutoThreshold**, the **Auto** portion can be disabled. When disabled, no calculation is done. The threshold used by the parent step is whatever value is in **Threshold**. **Edge Energy Threshold** and **Threshold Adjustment** are not used when **Auto** is disabled. The default setting is enabled.
- **Edge Energy Threshold** — Defines the pixel value at which a pixel in a Sobel Edge Enhancement is considered to be an edge pixel. This property is only used when **Auto Thresholding Enabled** is enabled.

Default: 10

Range: 0 to 255

- **Threshold Adjustment** — Used to offset or bias the dynamically calculated threshold, when **Auto** is enabled.

Default: 0

Range: -255 to 255

- **Threshold** — Displays the dynamically calculated threshold when **auto** is enabled. When **auto** is disabled, the value of this property is the threshold that is used by the parent step.

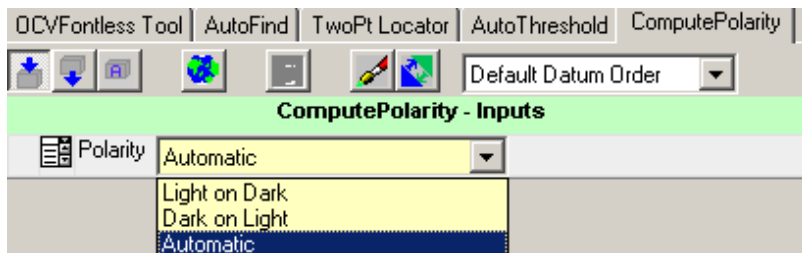
Default: 135

Range: 0 to 255

- **Process Every N Pixels** — As N increases, the accuracy of the threshold calculation may decrease, but the processing time for this step is reduced.

The ComputePolarity step of the OCVFontlessTool is used for automatic segmentation of the image during training.

FIGURE 10–50. ComputePolarity Properties Page

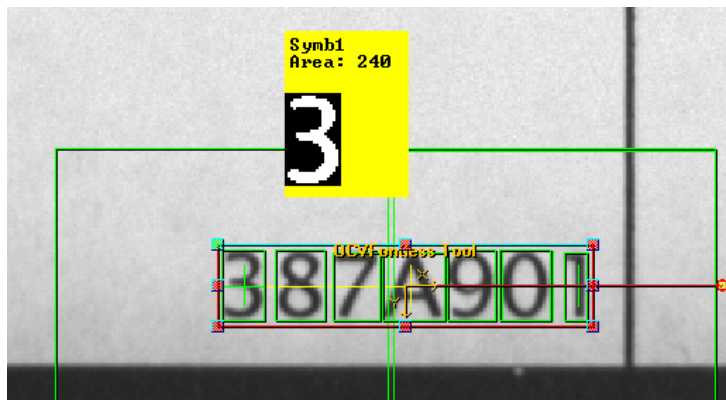


- **Polarity** — Allows the step to be set up to always return `Light_On_Dark`, always return `Dark_On_Light`, or return an automatically determined polarity. The default setting is `Automatic`.

Step Tips

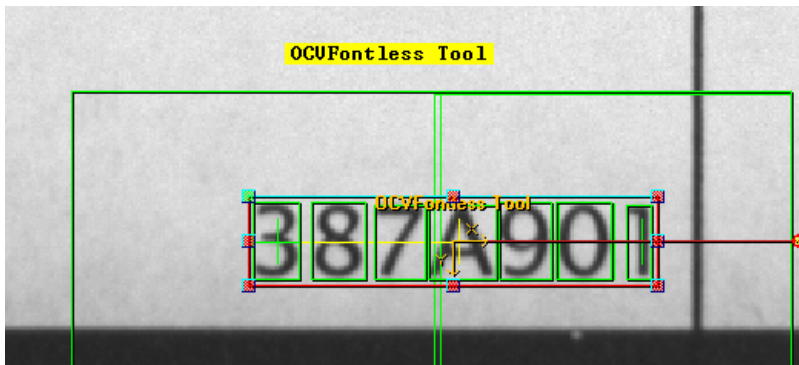
After the OCVFontlessTool has been trained, positioning the mouse over the ROI displays a Step Tip. This Step Tip provides information and graphical feedback for individual symbols when the mouse is over a symbol area. Train information includes the Area of the symbol, the number of On pixels in the binary template, and a bitmap representation of the binary template, as shown in Figure 10–51.

FIGURE 10–51. Step Tip — Example 1



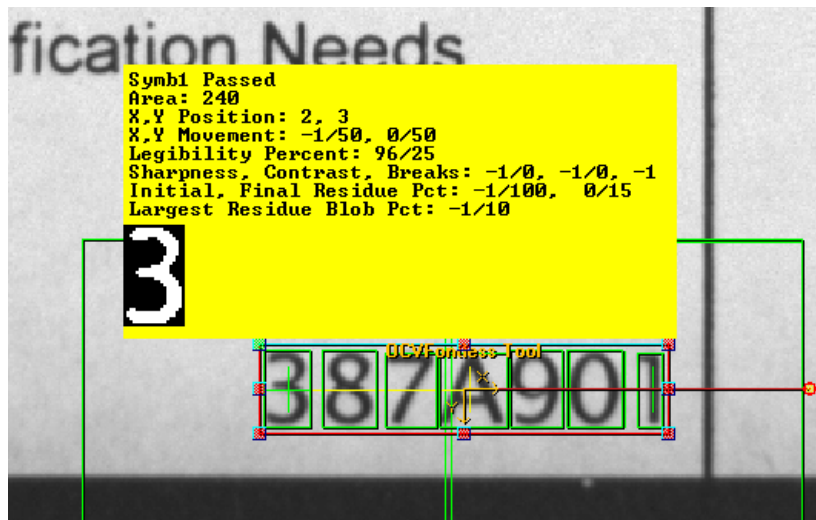
When the mouse is not positioned over a particular symbol area, the Step Tip displays the name of the OCVFontless Tool, as shown in Figure 10–52.

FIGURE 10–52. Step Tip — Example 2



When the OCVFontless Tool has been run, additional runtime information is available by holding down the Shift key when the mouse is positioned over the symbol area, as shown in Figure 10–53.

FIGURE 10–53. Step Tip — Example 3



Inspection information includes:

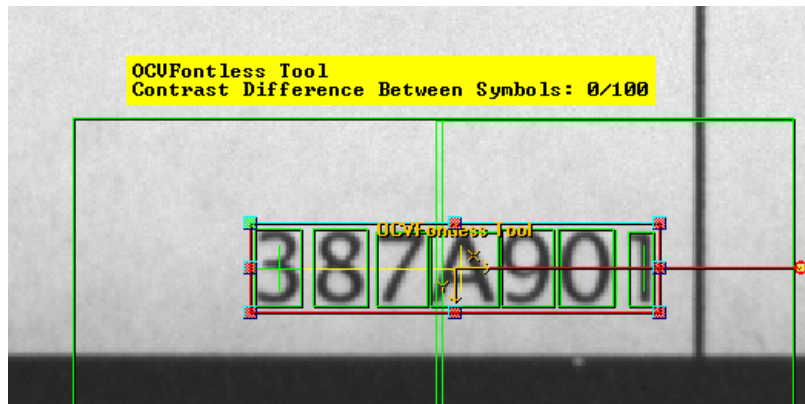
- The Area of the symbol, the number of On pixels in the trained binary template
- The X and Y position, upper left corner, of the symbol relative to the OCVRuntime Tool shape
- The X and Y allowed movement of the symbol
- The Legibility Percentage and the Legibility Tolerance
- The Sharpness, Contrast and number of Breaks found along with the associated tolerances
- The Initial and Final Residue percentages along with the associated tolerances
- The Largest (Final) Residue Blob Percentage and its associated tolerance

Note: A -1 for any value above, except the X and Y allowed movement, indicates the test is disabled.

- The bitmap representation of the binary runtime symbol area

When the OCFontless Tool has been run doing a Tryout, additional runtime information is available by holding down the Shift key and moving the mouse inside the OCFontless Tool ROI (but not over the symbols), as shown in Figure 10–54.

FIGURE 10–54. Step Tip — Example 4



OCVSymbolStep

This step is a collection of template images, settings and tolerances that inspect a character or logo at runtime. OCVSymbolSteps are created during the training of an OCVFontlessTool. They are used by the OCVFontlessTool to inspect the print at runtime.

FIGURE 10–55. Symb1 Properties Page

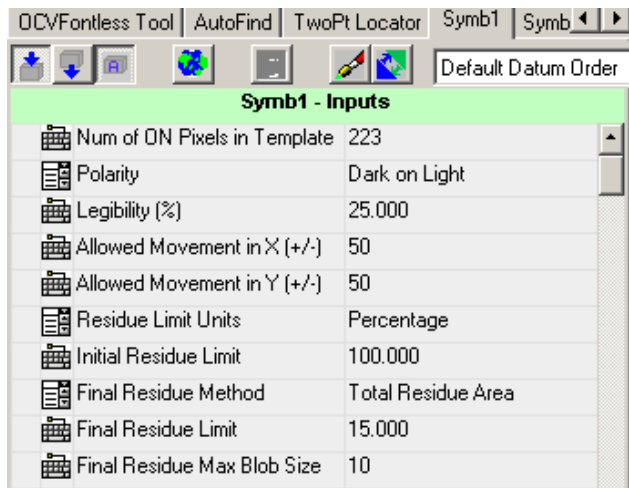


TABLE 10–6. Links to Property Descriptions

For Information About...	Go To...
Allowed Movement in X (+/-)	page 10–80
Allowed Movement in Y (+/-)	page 10–81
Appearance Flaw Break Test	page 10–83
Auto Threshold Adjustment	page 10–85
Auto Threshold Enabled	page 10–85
Bright Defect % Range	page 10–86
Character Expansions	page 10–85
Contrast Limit Units	page 10–84
Edge Energy Threshold	page 10–85
Filter Bright Defects	page 10–86
Final Residue Limit	page 10–82

TABLE 10–6. Links to Property Descriptions

For Information About...	Go To...
Final Residue Max Blob Size	page 10–83
Final Residue Method	page 10–82
Initial Residue Limit	page 10–81
Legibility (%)	page 10–80
Manual Threshold	page 10–85
Maximum Flaw Size	page 10–83
Min Appear. Flaw Break Size	page 10–83
Minimum Allowed Contrast	page 10–84
Minimum Allowed Sharpness	page 10–84
Num of ON Pixels in Template	page 10–80
Output Mask Type	page 10–86
Polarity	page 10–80
Residue Limit Units	page 10–81
Sharpness Limit Units	page 10–83

- **Num of ON Pixels in Template** — Displays the number of foreground pixels in the trained binary template.
- **Polarity** — Allows the step to always train with polarity `Light_On_Dark`, or always train with polarity `Dark_On_Light`. The default setting is set by the `OCVFontlessTool` during training.
- **Legibility (%)** — Passes/fails the symbol based on this minimum correlation percentage. The symbol will fail inspection when the correlation percentage is less than this value.

Default: 25%

Range: 0% to 100%

- **Allowed Movement in X (+/-)** — Sets the maximum number of pixels that a symbol can move in the X-axis (relative to other symbols) from its trained position.

Default: 50 pixels (any movement is allowed)

Range: 0 to 50 pixels. The maximum of 50 pixels comes from the parent `OCVTool` setting “Individual Symbol Search X”, which limits the search range in X to a maximum of 50 pixels in either direction.

- **Allowed Movement in Y (+/-)** — Set the maximum number of pixels that a symbol can move in the Y-axis (relative to other symbols) from its trained position.

Default: 50 pixels (any movement is allowed)

Range: 0 to 50 pixels. The maximum of 50 pixels comes from the parent OCVTool setting “Individual Symbol Search Y”, which limits the search range in Y to a maximum of 50 pixels in either direction.

- **Residue Limit Units** — Inputs the residue limits in either a maximum pixel count value or a percentage value (percentage value is based on the number of On pixels in the trained template). The default value is Percentage. When the value of this property changes, the values of **Initial Residue Limit** and **Final Residue Limit** are changed to match the selected units.
- **Initial Residue Limit** — Provides a quick check of the character quality and correctness. The initial residue calculation is done before any image processing is done on the residue image. When the system looks at the symbol being inspected, it determines the residue of the symbol, which is a count of those pixels that differ between the trained template and the current image. Based on the value of this property, the system determines if the residue is within tolerances. If it is not within tolerances, the symbol fails. Otherwise, the system continues on with the rest of the inspection procedure.

When **Residue Limit Units** is set to Percentage:

Default: 100.0%

Range: 0.0% to 100.0%

The value of this property is the smallest percentage of residue pixels (relative to the trained On pixel count) in the inspected image that makes the symbol fail the inspection.

When **Residue Limit Units** is set to Pixels:

Default: symbol size

Range: 0 to symbol size

The value of this property is the smallest count of residue pixels in the inspected image that makes the symbol fail the inspection.

This property is good for catching smudges that are aesthetically poor, but would pass after all inspection operations are performed on it. A value of 100% or symbol size means initial residue is ignored.

- **Final Residue Method** — Selects between three algorithms for final residue analysis:
 - **Total Residue Area** — This is the default. This choice counts all On pixels in the residue image and use the value in **Final Residue Limit**, pixel or percent, as the tolerance.
 - **Max Residue Blob** — Only counts the pixels in the largest blob of the residue image and use the value in **Final Residue Max Blob Size** as the tolerance.
 - **Both** — Performs both methods.
- **Final Residue Limit** — Sets the amount of objectionable residue that is to be deemed passable when **Final Residue Method** is set to Total Residue Area or Both. Final residue calculation is done after the image processing on the residue image that is associated with **Maximum Flaw Size**.

If there is little information in a symbol, i.e., a 1 as compared to an L, the percentage variation allowed should be reduced. An assignment of 0% (residue pixel count = 0) means that no residue is passable. An assignment of 100% (residue pixel count = symbol size) means that objectionable residue as large as the area of the prototype itself is passable.

When **Residue Limit Units** is set to Percentage:

Default: 15.0%, meaning 15% variation is acceptable
Range: 0.0% to 100.0%

The value of this property is the smallest percentage of residue pixels (relative to the trained On pixel count) in the inspected image that makes the symbol fail the inspection.

When **Residue Limit Units** is set to Pixels:

Default: 15% of the symbol size
Range: 0 to symbol size, and the default is 15% of the symbol size

The value of this property is the smallest count of residue pixels in the inspected image that makes the symbol fail the inspection.

Note: Determining the proper value for **Final Residue Limit** is a subjective decision; the higher the quality of the character/symbol desired, the lower the **Final Residue Limit** should be.

- **Final Residue Max Blob Size** — Used when Final Residue Method is set to Max Residue Blob or Both. A blob analysis is performed on the residue image and the largest blob is found. If this blob has an area that is greater than the value of this property, the symbol will fail the inspection.

Default: 10

Range: 1 to 512

- **Maximum Flaw Size** — Represents the maximum width in pixels that a discrepancy is allowed to be before it is considered objectionable. The larger the number assigned, the larger a discrepancy is allowed before causing the symbol inspection to fail.

Default: 1

Range: 0 to 20

- **Appearance Flaw Break Test** — Determines whether the OCVSymbolStep is to inspect for character breaks in the symbol. When enabled, the inspection fails if a break is found in the symbol. When disabled, the inspection ignores breaks in the symbol. The default setting is enabled.

- **Min Appear. Flaw Break Size** — Is the smallest size break that causes a character break failure.

Default: 2 pixels

Range: 1 to 10 pixels

- **Sharpness Limit Units** — Sets the units for Minimum Allowed Sharpness.
 - When set to “**Gray Level**,” the value of Minimum Allowed Sharpness is an absolute minimum value that the calculated sharpness value must be in order for the inspection to pass.
 - When set to “**Percentage**,” the value of Minimum Allowed Sharpness calculates a percentage of the trained sharpness value, which is then used as an absolute minimum value that the calculated sharpness value must be in order for the inspection to pass.
 - When switched from “**Gray Level**” to “**Percentage**,” Minimum Allowed Sharpness is updated to be the percentage value that corresponds to the gray level value that it previously held.

- When switched from “**Percentage**” to “**Gray Level**,” **Minimum Allowed Sharpness** is updated to be the gray level value that corresponds to the percentage value that it previously held.

Default: “Gray Level”

- **Minimum Allowed Sharpness** — Determines how crisp a symbol must be to pass inspection. It is measured by average edge strength over the entire symbol. Typical edge strengths are from 20 to 80 sharpness units. Whenever **Minimum Allowed Sharpness** has a value of 0, no sharpness tests are performed.

Default: 0

Range: 0 to 256 “Gray Level” or 0 to 100 “Percentage”

- **Contrast Limit Units** — Sets the units for **Minimum Allowed Contrast**.
 - When set to “**Gray Level**,” **Minimum Allowed Contrast** is an absolute minimum value that the calculated contrast value must be in order for the inspection to pass.
 - When set to “**Percentage**,” **Minimum Allowed Contrast** calculates a percentage of the trained contrast value, which is then used as an absolute minimum value that the calculated contrast value must be in order for the inspection to pass.
 - When switched from “**Gray Level**” to “**Percentage**,” **Minimum Allowed Contrast** is updated to be the percentage value that corresponds to the gray level value that it previously held.
 - When switched from “**Percentage**” to “**Gray Level**,” **Minimum Allowed Contrast** is updated to be the gray level value that corresponds to the percentage value that it previously held.

Default: “Gray Level”

- **Minimum Allowed Contrast** — This is the measurement that defines the grayscale foreground to background relationship of the symbol data. To calculate the contrast value, the average gray level value of the background pixels is subtracted from the average gray level of the foreground pixels. Whenever **Minimum Allowed Contrast** has a value of 0, no contrast checks are performed.

Default: 0

Range: 0 to 255 “Gray Level” or 0 to 100 “Percentage”

- **Auto Threshold Enabled** — Enables (default) and disables the automatic calculation of a threshold for binarizing the image at both train and run time. When enabled, the calculated threshold is displayed in **Manual Threshold**.

When disabled, no calculation is done. The threshold used for binarizing is whatever value is in **Manual Threshold**. **Edge Energy Threshold** and **Threshold Adjustment** are not used when this property is disabled.

- **Auto Threshold Adjustment** — Used to offset or bias the dynamically calculated threshold, when **Auto Threshold Enabled** is enabled.

Default: 0

Range: -64 to 64

- **Manual Threshold** — Displays the dynamically calculated threshold when the **Auto Threshold Enabled** property is enabled. When **Auto Threshold Enabled** is disabled, **Manual Threshold** is the threshold that is used for binarizing the image.

Default: 135

Range: 0 to 255

- **Edge Energy Threshold** — Defines the pixel value at which a pixel in a Sobel Edge Enhancement is considered to be an edge pixel. This property is only used when the **Auto Threshold Enabled** property is enabled.

Default: 20

Range: 0 to 255

- **Character Expansions** — Useful when dealing with print from a dot matrix printer or any print that is broken up in segments. The more sparse the print, the higher the value of this property should be.

This property allows for the broken print to become solid by expanding the segments until they come together. Dilations expand each segment. Then, erosions decrease the size of the character in every direction except the direction in which the segments have connected. Dilations and erosions work together to make the segments solid without making the character fatter.

Default: 0

Range: 0 to 9

- **Filter Bright Defects** — When enabled, runtime inspection of the symbol includes a pre-processing step for filtering out any bright defects in the image. The default setting is disabled.
- **Bright Defect % Range** — Percentage that determines the threshold at which the bright defect filter processes. The threshold is calculated by taking this percentage of the range between the binarizing threshold and 255.

Default: 0 (binary threshold is used if **Filter Bright Defects** is enabled)

Range: 0 to 100

- **Output Mask Type** — Used in conjunction with the DynamicMask step. This property allows three selections:
 - **None** — Adds nothing to the mask.
 - **Mask Template** — Only the foreground area of the symbol is added to the mask. This is the default.
 - **Mask ROI** — The entire area within the symbol's ROI is added to the mask.

OCV Tips

The OCV tools found in Visionscape have many settings and adjustments to allow for maximum flexibility. However, most applications require attention to only a few of these settings and adjustments.

OCVFont

- *ID Test Determination Pct* — Lowering the default value of 90% to 85% or 80% causes more characters to be flagged as similar.

Layout Step

- Automatic Segmentation — By default, this option is off, which allows the font library creation box to be manually sized around each character as it is entered into the library. Visionscape is designed such that you perform this individual training of characters to activate runtime ID checking of special characters like O, 0, B, 8, D, etc. ID checking requires that these symbol boxes be the same size.

Note: Whenever possible, use the same size box to train all font library characters.

You can enable the automatic segmentation option, and Visionscape automatically locates and places a box around all characters in the FOV.

DefaultSymbol

- Final Residue Limit — Increasing the default value of 15% to a higher number allows characters to vary more and still be accepted. Increasing this value has a gradual effect.
- Max Flaw Size — Increasing the default value of 1 pixel to a higher value allows characters to vary more and still be accepted. Increasing this value has a rapid effect. This number should not be set greater than 2 for most applications.

OCVRuntime Tool

The OCVRuntime tool is the vision step that actually performs the inspection of a code. The following settings are found under **Tool Settings**.

Tool Settings

- Individual Symbol Search X — Increasing these values allows individual characters to move more in relation to one another.
- Individual Symbol Search Y — Increasing these values allows individual characters to move more in relation to one another.

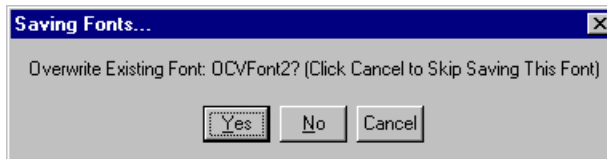
Layout Step

- Allowed Overlap During Read — Increasing this value from the default value of 2 allows characters to be identified during the train step when their character boxes overlap.
- Min Read Match % — Decreasing the 65% default value allows characters to be identified at train time when they vary significantly from what was trained into the font library.

Converting Jobs with Embedded OCVFonts

Jobs created with Version 2.3 or earlier, or 3.0 - 3.1, that contain font based OCV tools are updated to use the new OCVFont method when those Jobs are read into Visionscape. When the old Job is read in, any OCVFonts found in an existing OCVFontFolder are saved to the Vscape\Jobs\Fonts folder using the font name found in the Job. If a font with that name already exists, you are asked whether or not to overwrite the existing font, as shown in Figure 10–56.

FIGURE 10–56. Overwrite Existing Font Dialog Box



Clicking **Cancel** causes the font to not be saved. Clicking **Yes** causes the font being read to overwrite the existing one. Clicking **No** causes a prompt to appear so that you can enter a unique name for the font being saved, as shown in Figure 10–57.

FIGURE 10–57. Enter a Unique Name Dialog Box



When the old Job is read in, any font based OCV tool found is modified so that the LayoutStep's "Select Font" datum has the correct font chosen. This allows old Jobs to be loaded and run without requiring a re-train of the font based OCV tools.

Troubleshooting

Training Font Based Tools – Read Match%

To reproduce the behavior, Visionscape V2.5.1.12 was used. We believe that this behavior is consistent across all versions of Visionscape.

The Behavior

1. First, train the OCVFont. In the example given, “text02.tif” was used to train characters “L”, “2”, “A”, “0”, “6”, “9”, “3”.
2. Set Read Match % = 80 % on Layout step in the OCVRuntime tool.
3. Train OCVRuntime tool.
4. You will find that the “L” character is not found even though the same “text02.tif” file is being used.

The Explanation

Training

When a character is trained into an OCVFont, there are several templates stored for the character. One of these templates is a Sobel Edge Enhancement template.

When the OCVRuntimeTool is trained, the ROI being searched for characters is first passed through a Sobel Edge Enhancement. Then, the Sobel Edge Enhancement templates (for each symbol) are used in a correlation to determine where the symbols are within the ROI. The “Read Match%” value accepts/rejects a found character based on correlation match percentage.

Correct Character Training

When a character is trained, it is important to correctly size the training box around the character. This manual describes character training as follows:

Individual character training requires that the OCVFont ROI be placed close around a single character in the image, leaving a 1-2 pixel border. This box should not include any portion of the adjacent characters. The minimum character width we recommend is 20 pixels.

The 1 - 2 pixel border is important because it allows room for proper training of templates.

Auto-Segmenting Character Training

When the “Auto-Segment” option is used to train characters, the system locates blobs within the ROI. A bounding box for each blob trains the symbols. You can use the OCVFont’s LayoutStep property Num Border Spaces to Add to increase the size of the box used to train the symbols.

This “Auto-Segment” method of training OCVFonts is not necessarily the optimal method for training characters. In particular, the 1 - 2 pixel border requirement may not be met.

Example with Auto-Segment

Using the “text02.tif” image, a sample OCVFont, found in the \\Vscope\Jobs folder, was trained using auto-segmentation with all parameters at the default values. The characters “L”, “2”, “A”, “0”, “6”, “9”, “3” were trained into the OCVFont.

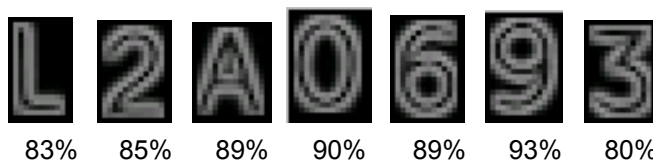
When the OCVRuntimeTool was trained, the Sobel Edge Enhancement search buffer was:

FIGURE 10–58. Contents of Buffer



The Sobel Edge Enhancement templates and the resulting Match% were:

FIGURE 10–59. Resulting Match%



When you look closely at the template images and compare them to the search buffer, you can see that the template images are missing some data. This is because the Auto-Segment boxes were not big enough when the characters were trained. It is this missing data that causes the match % to go down, even when training on the same image.

Example with Manually Trained Characters

Using the “text02.tif” image, a sample OCFont was trained by manually placing the training box over the characters. All parameters were at the default values. The characters “L”, “2”, “A”, “0”, “6”, “9”, “3” were trained into the OCFont.

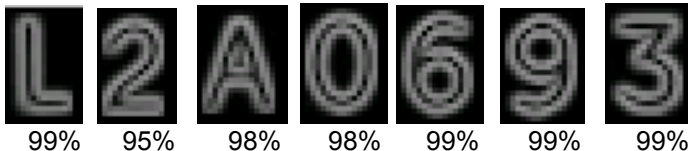
When the OCVRuntimeTool was trained, the Sobel Edge Enhancement search buffer was (the same as the above example):

FIGURE 10–60. Contents of Buffer



The Sobel Edge Enhancement templates and the resulting Match% were:

FIGURE 10–61. Resulting Match%



Comparing these templates to the search buffer, you can see that the template characters more closely resemble those in the search area as reflected by the Match % values. By manually positioning the training boxes, we were able to ensure that the proper amount of spacing was available for template training.

Conclusions

- Currently, we recommend that OCVFont training be performed using manual placement of the training box. This is because the training box should be the same size over symbols that are similar in order for the Runtime ID Checking algorithms to work optimally. The “Auto-Segmentation” cannot assure that the boxes around similar characters are the same size. Using manual placement with the correct 1 - 2 pixel border will also help prevent the behavior described in this document.
- The “Auto-Segmentation” training of OCVFonts is useful for quick demos or to quickly prove out an inspection scenario. While the behavior described may cause confusion at train time when using a frozen image, the Runtime inspections remain intact. The runtime inspections do not use the Sobel Edge Enhancement templates.
- You may need to increase the value for **Num of Border Spaces to Add** to 2 (from the default value of 1). This would give the Auto-Segmentation characters more border space with which to train the characters. While this would probably help with the Read Match %, increasing the border space may cause some symbol overlap in cases where the symbols are close together. Again, we recommend manual training of the OCVFont.

Custom Tools

This chapter discusses custom tool functionality.

Custom Step

This step facilitates the development of new steps using the interpreted script language Perl. Custom Steps can be inserted anywhere in a Job.

The Custom Step consists of optional input datums, optional output datums and a script file written in Perl that manipulates these datums. The functionality of the step is controlled by the Perl script.

Other Steps Used

None.

Theory of Operation

Custom Step essentially processes any data-in and/or generates any data-out through a package script written in Perl. Data-in consists of input datums and resource datums. Data-out consists of output datums. The number and type of inputs and outputs is determined by the package script.

Standard Perl calls and functionality are available with additional Perl packages that allow access to Visionscape functionality.

Note: Please refer to the Visionscape Perl Script Custom Tool Programmers Manual for a complete description of how to create a Visionscape Perl script.

When a function in the Custom Step is called, a corresponding subroutine in the package script is called. Each package script must support the following functions/subroutines:

- sub Apply — Called when the Perl script is changed or when you click **Recreate Step Datums**.
- sub Update — Code that needs to run after a Job is loaded in from memory. This function is called after the Job is loaded from disk.
- sub PreRun — Set up resources needed for Run. These can be buffers and allocated memory to hold intermediate results.
- sub Run — The main processing that this step performs each time it is run.
- sub PostRun — Cleanup of resources, memory, and deallocation of the intermediate results.
- sub Draw — Called to show graphics.

Note: A Custom Step does not have an input buffer and, therefore, has no default place to draw. When drawing is needing, use the CustomVision Tool.

- sub Init — Called when the Custom Step is created and occurs when the Job is downloaded to the vision device. Typically, used for creation/allocation of objects that is done once over the life of the script.
- sub Cleanup — Called when the step is released. This occurs when the Job is deleted prior to loading a new Job or during a download to the vision device.
- sub Start — Called when the inspection is started. A flag indicating whether this is the first time running can be obtained through the `GetStartFirsttimeFlag()` call in the `perlutil` package.
 - When the flag is **true**, the Job is prepared to run (all steps have pre-run) but has never run.
 - When the flag is **false**, the Job has previously run and is starting again, as in start, stop, start.
- sub Stop — Called when the inspection is stopped.

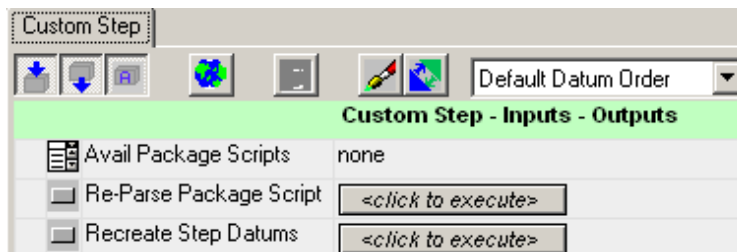
Note: If the step is made trainable by calling `perlutil::MakeMeTrainable()`, then the script must also support:

sub Train — Called when the step is trained.

Description

Custom Step allows editing through the Custom Step properties page, as shown in Figure 11–1.

FIGURE 11–1. Custom Step Properties Page



Settings

- **Avail Package Scripts** — A drop-down list that contains a list of available Perl package script files. Once a package script is written, it should be registered, as follows, before it becomes available from this list box:
 - The name of the Perl script file must be listed in the master script file `perlscr`. This is a text file located in the `\Vscape\Jobs\Perlmod` folder. You add a new Perl script to its list by adding a require statement:
- The Perl script file must be in the `\Vscape\Jobs\perlmod` directory created at installation time.
- The register function is called within the package script and outside of all subroutines in the script. For example:

```
perlutil::register("packagename")
```

Once registered, the tool or step, represented by the Perl script, is available for use through this list box. For example, Visionscape installs several example Perl scripts that are ready to be used with the Custom Step or Custom Vision tool. Those scripts are:

- none — The default package script. This script should always be available, so the line `require none` must remain in the master script file, `perlscr`. The file containing this package, `none.pm`, must remain in the `perlmod` directory. **YOU SHOULD NEVER REMOVE THIS SCRIPT!**
- FailCode — This script provides a way to encode the failure statuses of up to 31 different Steps in your Job into a single 32 bit integer value (the 32nd bit is not supported).

Please refer to the end of this chapter for complete descriptions of each of the above scripts.

- **Re-Parse Package Script** — This button causes the package script to be parsed when clicked. Whenever any changes to the script are made, this button needs to be clicked to make these changes take effect. If there are syntax errors in your script, a message will be dumped to the Visionscape Debug Window to alert you.
- **Recreate Step Datums** — This button causes the input and output datum lists in the step to be recreated. This button only needs to be clicked when a datum is either added, removed or changed in the script. If the script is changed, but no input or output datums are changed, then this button does not need to be clicked. Clicking this button also causes all input datums to be set to their default values and lose their connections to other step results or parameters.

Any datums created by the Perl script will be added to the property page of the Custom Step. The input datums can be linked to other similar type datums in the Job, just as they can with any other Visionscape Step. Resource datums are shown as user-editable boxes that can be set to a value directly.

Versions of Script Files

Custom Steps have no mechanism for determining whether a change has been made to a script. During development, you will need to re-parse the script whenever you make changes to it, and you will need to re-create the datums in the step whenever you modify the list of input or output datums that are created by your Perl script.

Note: When loading an existing Job that contains a script that has been changed, the datums in the step will not update when the Job is imported/opened. This means the datums created will match those created by the script when the Job was saved, and will not match the datums in the revised script. You must explicitly re-create the datums in the step, by clicking **Recreate Step Datums**, and re-link all input and output datums as well. Failure to do so could lead to your script crashing Visionscape.

For example, consider a saved Job that contains a Custom Step using a script that created three output datums (DoubleDm & DoubleDm & IntDm) when the Job was saved originally. Now, the script has been revised to create a PointDm and an IntDm. When the saved Job is loaded, the original datums (DoubleDm, DoubleDm, IntDm) are created. The script, however, will try to set the output datums as PointDm, IntDm, which will not work. Therefore, when a script's list of input or output datums is modified, any Jobs containing that script should re-parse the script and re-create the datums after they have been opened/read and before they are run.

This is only necessary when the changes to the script affect the number or type of datums (input, output or resource) in the step. If only internal processing has changed, then there is no reason to re-create the datums in the step.

Script Files Not Found or “none” Script

If the script in any Custom Step cannot be found, or is set to the default “none” script, then the step will fail when it is run. A message will also be sent to the debug window indicating the problem. If a Job is read in with Custom Steps, and one or more of the embedded Perl scripts cannot be found, then the script selection is changed to the “none” script, which is always present. This, however, may cause some part of the inspection to not be done, so the inspection must fail. Also, a Custom Step with the “none” script selected does not serve any purpose and has most likely been left uninitialized, so, as a safeguard, the inspection fails.

To ensure that the desired Perl script can be found, there must be an entry in the master script file, `perlscr`, with the filename containing that script. Also, if the Perl interpreter runs into a problem reading any of the files in the `perlscr` file (syntax errors, file not found, etc.), then the interpreter will not continue to try to parse any subsequent files in the list. Consider the following example of the default `perlscr` file:

```
require none; # Always on top for default insertions
require cylinderunwrap;
require dynamicbinarize;
require failcode;
require findrotated;
```

If the Perl interpreter was unable to load the “failcode” Perl script in our example above, then it would stop parsing the `perlscr` file and exit, meaning the “findrotated” Perl script would not be loaded either. Keep this in mind if you are having trouble getting one of your scripts to show up in the list of Available Package Scripts.

Training

The Custom Step can be made trainable by doing a `perlutil::MakeMeTrainable()` call in the `Init` subroutine of the script. The processing that is done during training is contained in the `Train` subroutine of the script.

Results

Results of the step are determined by the package script. The datums in the step can be set through function calls in the script (`perlutil::set_datum_*`) to expose these results outside of this step.

I/O Summary

None.

Custom Vision Tool

This tool facilitates the development of new steps using the interpreted script language Perl. CustomVision Tool requires an input image and can go anywhere a step would normally go (i.e., somewhere that has an input image).

The CustomVision Tool takes an image input and (optional) data input and creates (optional) data output (optionally, the step can create image output, too, depending on the script). The step consists of an input buffer datum, a script file written in Perl, and optional input datums and output datums defined by the Perl Script. The functionality of the step is controlled by the Perl script.

Note: Please refer to the Visionscape Perl Script Custom Tool Programmers Manual for a complete description of how to create a Visionscape Perl script.

Other Steps Used

None.

Theory of Operation

The CustomVision Tool essentially processes any image/data-in (and/or generates any data-out) through a package script written in Perl. Data-in consists of an input buffer datum, input datums and resource datums. Data-out consists of output datums. The number and type of inputs and outputs is determined by the package script.

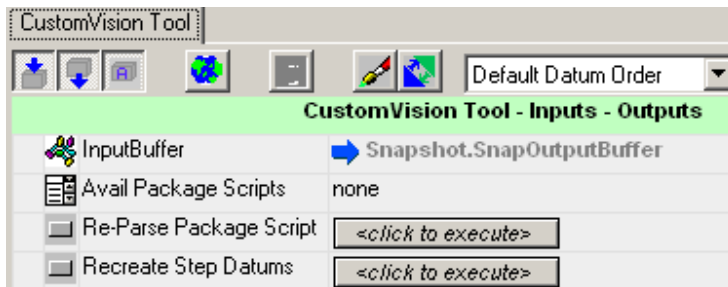
Standard Perl calls and functionality are available with additional Perl packages that allow access to Visionscape functionality.

Refer to the Visionscape Perl Script Custom Tool Programmers Manual for a complete description of how to create a Visionscape Perl script.

Description

CustomVision Tool allows editing through the CustomVision Tool properties page, as shown in Figure 11–2.

FIGURE 11–2. CustomVision Tool Properties Page



Settings

- **Avail Package Scripts** — A drop-down list that contains a list of available Perl package script files. Once a package script is written, it should be registered as follows before it becomes available from this list box:

- The Perl script file must be listed in the master script file perlscr. This is a text file located in the \Vscape\Jobs\Perlmod folder. You add a new Perl script to it's list by adding a require statement:

```
require filename; #note that .pm is not included
```

- The Perl script file must be in the \Vscape\Jobs\perlmod directory created at installation time.
- The register function is called within the package script and outside of all subroutines in the script. For example:

```
perlutil::register("packagename")
```

Once registered, the tool or step, represented by the Perl script, is available for use through this list box. For example, Visionscape installs several example Perl scripts that are ready to be used with the Custom Step or Custom Vision tool:

- none — The default package script. This script should always be available, so the line require none must remain in the master script file, perlscr. The file containing this package, none.pm, must remain in the perlmod directory. **IMPORTANT: You should never remove this script.**

- **Cylinder_UnWarp** — Used to unwrap the surface of a cylindrical object, such as a bottle or tube. This script produces an output buffer with the unwrapped image that you can then run other steps within.
- **Dynamic_Binarize** — This script will produce a binarized output image (an image where every pixel is either 0 or 255) of the pixels contained within the ROI. The threshold used to binarize the image is calculated dynamically each time the Step runs. An output buffer is created that contains the binarized image, and additional steps may be inserted into this buffer to perform analysis.
- **FailCode** — The FailCode script does not draw any graphics nor does it need an ROI, so it is best used with the Custom Step rather than the Custom Vision Tool.
- **FindRotated** — This script allows you to run the correlation algorithm over a range of angles. The Template find Step can only find features if they rotate by less than 5° from their trained orientation. This script allows you to specify a larger amount of rotation, and then the tool will warp the image by specified increments looking for the trained feature.

Please refer to the end of this chapter for complete descriptions of each of the above scripts.

- **Re-Parse Package Script** — This button causes the package script to be parsed when clicked. Whenever any changes to the script are made, this button needs to be clicked to make these changes take effect. If there are any syntax errors in your script, an appropriate message will be dumped to the Visionscape Debug Window.
- **Recreate Step Datums** — This button causes the input and output datum lists in the step to be recreated. This button only needs to be clicked when a datum is either added, removed or changed in the script. If the script is changed, but no input or output datums are changed, then this button does not need to be clicked. Clicking this button also causes all input datums to be set to their default values.

Any datums created by the Perl script will be added to the property page of the Custom Step. The input datums can be linked to other similar type datums in the Job, just as they can with any other Visionscape Step. Resource datums are shown as user-editable boxes that can be set to a value directly.

Versions of Script Files

Custom Steps have no mechanism for determining whether a change has been made to a script. During development, you will need to re-parse the script whenever you make changes to it, and you will need to re-create the datums in the step whenever you modify the list of input or output datums that you are creating in your Perl script.

Note: When loading an existing Job that contains a script that has been changed, the datums in the step will not update when the Job is imported/opened. This means the datums created will match those created by the script when the Job was saved, and will not match the datums in the revised script. You must explicitly re-create the datums in the step, by clicking Recreate Step Datums, and re-link all input and output datums as well. Failure to do so could lead to your script crashing Visionscape.

For example, consider a saved Job that contains a Custom Step using a script that created three output datums (DoubleDm & DoubleDm & IntDm) when the Job was saved originally. Now, the script has been revised to create a PointDm and an IntDm. When the saved Job is loaded, the original datums (DoubleDm, DoubleDm, IntDm) are created; the script, however, will try to set the output datums as PointDm, IntDm, which will not work. Therefore, when a script modifies the list of input or output datums, any Jobs containing that script should re-parse the script and recreate the datums after they have been opened/read and before they are run. If only internal processing has changed, then there is no reason to recreate the datums in the step.

Script Files Not Found or “none” Script

If the script in any Custom Step cannot be found or is set to the default “none” script, then the step will fail when it is run. A message will also be sent to the debug window indicating the problem. If a Job is read in with Custom Steps and if any of the embedded Perl scripts cannot be found, then the script selection is changed to the “none” script which is always present. This, however, may cause some part of the inspection to not be done, so the inspection must fail. Also, a Custom Step with the “none” script selected does not serve any purpose and has most likely been left uninitialized, so, as a safeguard, the inspection fails.

To ensure that the desired Perl script can be found, there must be an entry in the master script file, `perlscr`, with the filename containing that script. Also, if the Perl interpreter runs into a problem reading any of the files in the `perlscr` file (syntax errors, file not found, etc.), then the interpreter will not continue to parse any subsequent files in the list. Consider the following example of the default `perlscr` file:

```
require none; # Always on top for default insertions
require cylinderunwrap;
require dynamicbinarize;
require failcode;
require findrotated;
```

If the Perl interpreter was unable to load the “failcode” Perl script in our example above, then it would stop parsing the `perlscr` file and exit, meaning the “findrotated” Perl script would not be loaded either. Keep this in mind if you are having trouble getting one of your scripts to show up in the list of Available Package Scripts.

Training

The Custom Vision Tool can be made trainable by doing a `perlutil::MakeMeTrainable()` call in the `Init` subroutine of the script. The processing that is done during training is contained in the `Train` subroutine of the script.

Results

Results of the step are determined by the package script. The datums in the step can be set through function calls in the script (`perlutil::set_datum_*`) to expose these results outside of this step.

I/O Summary

None.

Perl Scripts Included with Visionscape

Visionscape includes several standard and color Perl scripts that are ready to be used immediately, and that provide useful functionality. In this section, we will describe each of those scripts, and explain how to use them.

Cylinder_UnWarp

This step is documented fully in Chapter 12, “Image Transform Tools”.

Dynamic_Binarize

Theory of Operation

The Dynamic Binarize script is used when you want to binarize your image, which means to convert all of the pixels below a threshold to 0 and all those above the threshold to 255. This script will dynamically calculate its binary threshold each time it runs. This Step can only be used with the Custom Vision Tool, and it can not be used with the Custom Step.

Using Dynamic_Binarize

This script provides you with an ROI like any other Vision tool in Visionscape would. You can position and size the ROI over any area of your image, and an output buffer will be created of the same width and height, and containing the binary representation of all of the pixels within the ROI. In Figure 11–3 and Figure 11–4, we demonstrate how the text on a chip can be binarized:

FIGURE 11-3. Custom Vision Tool Running Dynamic_Binarize Perl Script

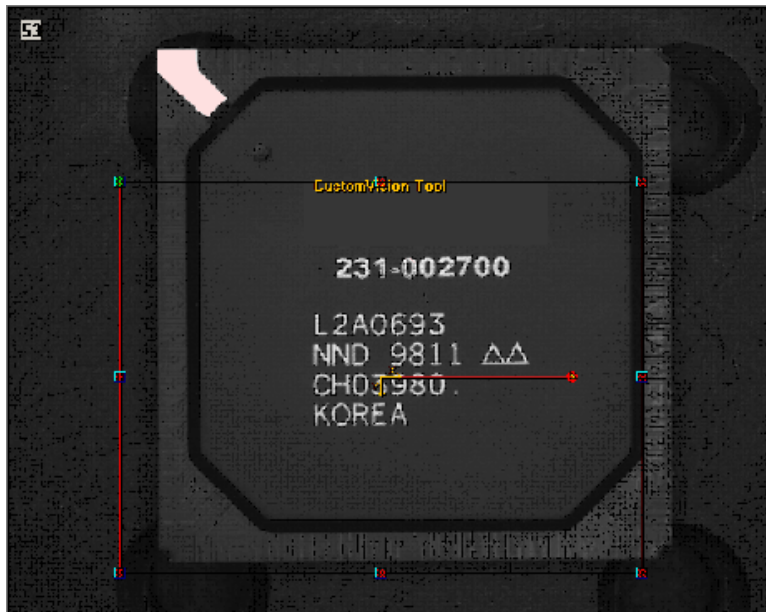


FIGURE 11-4. Output Buffer produced by Dynamic_Binarize Perl Script

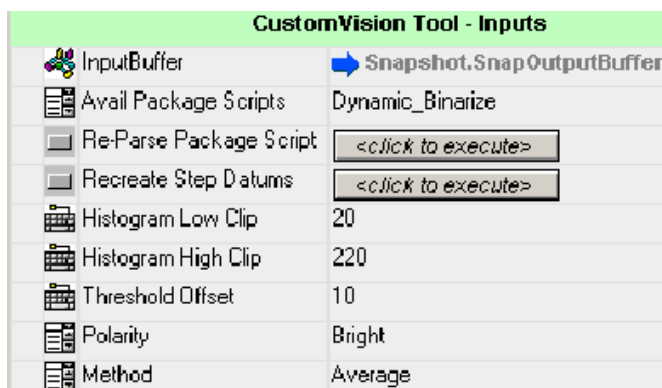


Description

The Dynamic Binarize script will calculate either the average or median gray value of all the pixels within its ROI, and this value will be used as the binarize threshold. When calculating the average, you can choose to ignore the very lowest and highest gray values. You may also apply an offset to the calculated threshold.

Settings

FIGURE 11–5. Dynamic Binarize Script



- **Histogram Low Clip** — Pixels below this gray value will be left out of the calculation of the average or median gray value.
- **Histogram High Clip** — Pixels above this gray value will be left out of the calculation of the average or median gray value.
- **Threshold Offset** — An offset that will be applied to the calculated binary threshold.
- **Polarity** — Determines the polarity of the pixels in the output buffer.
- **Method** — Selects whether you want the threshold to be based on the average or the median gray value.

Results

- Average — Calculated average gray value.
- RunLoThr — When polarity is set to “Bright”, this will hold the actual threshold that was used to binarize the image; in other words, it will be the average or median gray value + the Offset value. When polarity is set to “Dark”, this will always be 0.
- RunHiThr — When polarity is set to “Dark”, this will hold the actual threshold that was used to binarize the image; in other words, it will be the average or median gray value - the Offset value. When polarity is set to “Bright”, this will always be 255.
- Median — The calculated median gray value.
- ComputedGray — This is the computed gray value that was combined with the offset value to produce the threshold. In other words, if the selected “Method” was “Average”, this will be equivalent to the Average output datum, and if the selected “Method” was “Median”, this will be equivalent to the Median output datum.



































FailCode

The FailCode script allows you to encode the failed statuses of up to 31 Steps in your Job into a single 32 bit integer value. Typically, this is used when the user wants to cut down on the amount of data that is uploaded from the inspection, but also wants to know the status of a large number of Steps.

Using FailCode

The FailCode script does not draw any graphics nor does it need an ROI, so it is best used with the Custom Step rather than the Custom Vision Tool. When added to your Job, the FailCode script will present you with a list of 31 inputs on the properties page, as shown below:

FIGURE 11-6. FailCode Script

Custom Step - Inputs - Outputs	
 Bit0	➔ Snapshot.Status : { True }
 Bit1	➔ Blob Tool.Status : { True }
 Bit2	➔ Flaw Tool.Status : { True }
 Bit3	➔ OCvFontless Tool.Status : {
 Bit4	➔ <Unassigned>
 Bit5	➔ <Unassigned>
 Bit6	➔ <Unassigned>
 Bit7	➔ <Unassigned>
 Bit8	➔ <Unassigned>
 Bit9	➔ <Unassigned>
 Bit10	➔ <Unassigned>
 Bit11	➔ <Unassigned>
 Bit12	➔ <Unassigned>
 Bit13	➔ <Unassigned>
 Bit14	➔ <Unassigned>
 Bit15	➔ <Unassigned>
 Bit16	➔ <Unassigned>
 Bit17	➔ <Unassigned>
 Bit18	➔ <Unassigned>
 Bit19	➔ <Unassigned>
 Bit20	➔ <Unassigned>
 Bit21	➔ <Unassigned>
 Bit22	➔ <Unassigned>
 Bit23	➔ <Unassigned>
 Bit24	➔ <Unassigned>
 Bit25	➔ <Unassigned>
 Bit26	➔ <Unassigned>
 Bit27	➔ <Unassigned>
 Bit28	➔ <Unassigned>
 Bit29	➔ <Unassigned>
 Bit30	➔ <Unassigned>
 Avail Package Scripts	FailCode
 Re-Parse Package Script	<click to execute>
 Recreate Step Datums	<click to execute>

Each of these inputs can be connected to any Status datum in your job. Typically, you would connect the Statuses of all of the Steps whose pass/fail state you care about. In the example here, we have connected the statuses of the Snapshot step to Bit 0, a Blob Step to Bit 1, a Flaw Tool to Bit 2 and an OCV Fontless Tool to Bit 3. If all of these Steps should pass, the FailCode script will produce an output value of 0. If any of these Steps should fail, the corresponding Bit in the output integer value will be set to a 1. So, for example, if the Blob tool and the OCV Fontless tool should fail, this would mean that bits 1 and 3 would be set to 1, producing an output value of 10. Any Bits that are left “Unassigned” are ignored and will not effect the output value.

Settings

- **Bit0 - Bit30** — Each of these input datums can be connected to any Status Datum in the Job. If the Status is False, then the corresponding bit of the output word is set to 1.

Results

- **Output Value** — This integer value holds the failure code. The bits of this word correspond to the 31 input datum values (bit 32 not supported).

FindRotated

The FindRotated script allows you to run the correlation algorithm over a range of angles, allowing you to find features that will rotate by more than 5° from the trained orientation. The Template Find step in Visionscape runs the correlation algorithm, but can typically only find features that will rotate by no more than $\pm 5^\circ$.

Using FindRotated

FindRotated is used in the same way that the Template Find step is used. You will be provided with two ROIs:

- The first represents the template you wish to train on.
- The second represents the search area (the area within which you will search for the template).

You must train this step before you can use it.

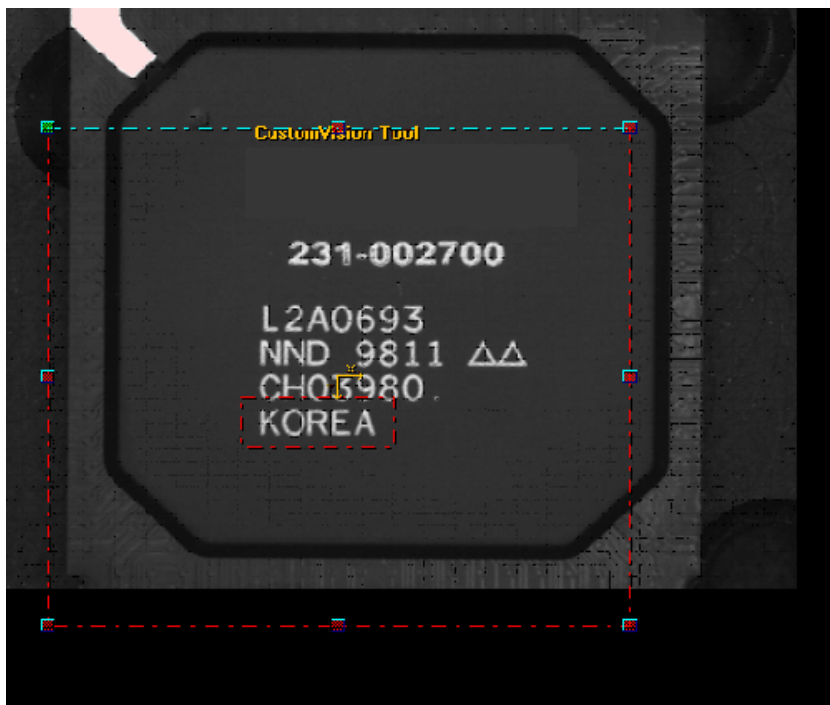
Insert a Custom Vision tool into your Job, and then select the Find Rotated Script. You should see two ROIs in your image that look something like this:

FIGURE 11–7.



Unfortunately, the ROIs are not labeled, so it is confusing to understand which of the ROIs is used for the Template, and which is used for the search area. We have labeled which is which in the image above. If we wanted to train the tool to find the “KOREA” text in our sample image, we would position the ROIs like this:

FIGURE 11–8.















Press the **Train** button in FrontRunner to train the template.

Description

The FindRotated script will search for the trained template by searching over a specified range of angles specified by the **Maximum Search Angle** and **Minimum Search Angle** datums. It accomplishes this by warping the image contents inside of its ROI. It will start by warping the image contents by the angle specified by the **Minimum Search Angle** datum, and then it will run correlation on the result. All qualifying match locations are recorded, and then the angle is incremented by an amount equal to the value specified in the **Angle Step Size** datum, and the image is warped and searched again. It will continue in this fashion until it reaches the angle value specified in the **Maximum Search Angle** datum, and then it will stop. Then, the script will scan through all of the qualifying template locations, and choose the best one.

Settings

FIGURE 11–9. FindRotated Script

CustomVision Tool - Inputs - Outputs	
 InputBuffer	 Snapshot.SnapOutputBuffer
 Avail Package Scripts	FindRotated
 Re-Parse Package Script	
 Recreate Step Datums	
 Minimum Search Angle	-15.000
 Maximum Search Angle	15.000
 Angle Step Size	5.000
 Angle Step Size	5.000
 Accept Threshold	0.700

- **Minimum Search Angle** — The minimum warp angle that the step should start searching at.
- **Maximum Search Angle** — The maximum warp angle that the step should search at.
- **Angle Step Size** — This is the amount in degrees that the angle should be incremented by for each search iteration.
- **Angle Step Size (2)** — This datum is mistakenly specified twice in the current Perl script. This datum should be ignored as it is not used.
- **Accept Threshold** — This is the minimum correlation match percentage.

Results

- Point of Best Match — This is the X,Y location and angle of the best match found.

Additional Example Scripts

The following scripts are also installed with Visionscape, but are not included in the perlscr file, and therefore are not made available to the user by default. If you wish to use any of these scripts, simply add them to the perlscr file.

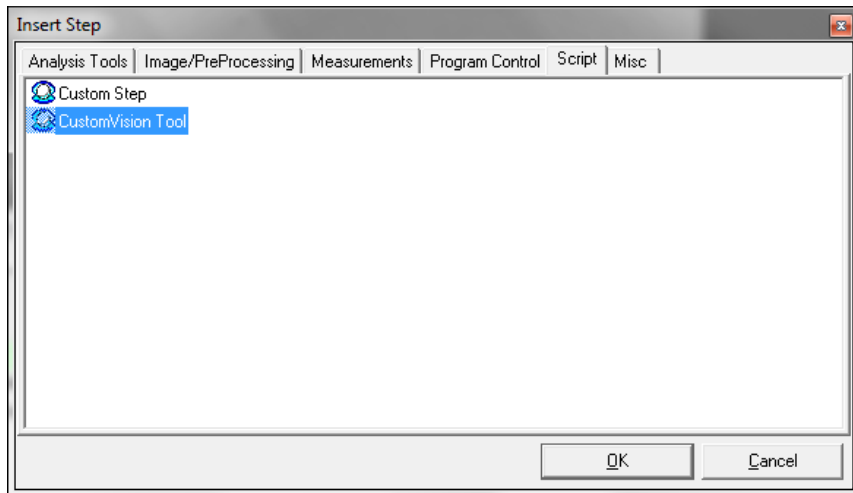
- AngleFind.pm — This script file contains the Angle_Find package. This script finds the dominant angle of a scene using the Sobel agent to compute an angle image and then uses the Histogram agent to find the dominant angle bucket. The script outputs the dominant angle value and a score.
- AngleTest.pm — This script file contains the JustStraighten package. This script uses the SceneAngleAgent to find the dominant angle in the image. Then, it will rotate the contents of the input buffer such that its X-axis is aligned with the dominant angle. This step is somewhat unique in that it actually modifies the input buffer, rather than following Visionscape convention and creating its own output buffer with the modified pixel data.
- Measure.pm — This script file contains the PtLineNormal_Meas package. This script performs a Point to Line Distance measurement. It is intended as an example of how to perform measurements in Perl, and also of how to handle transforming points and lines into a common coordinate system.

Color Perl Scripts

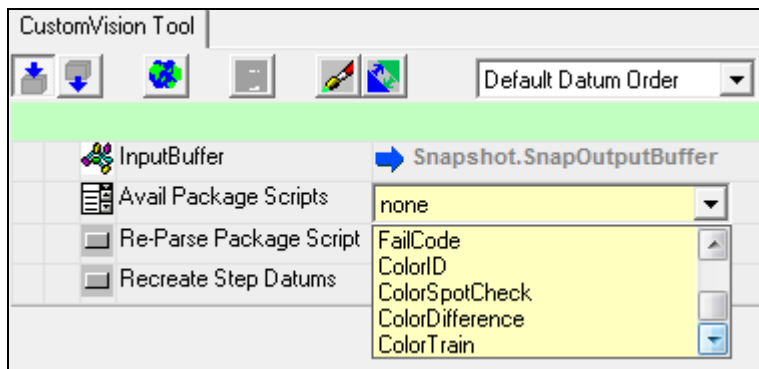
Color versions of the MicroHAWK MV, HAWK MV-4000, and GigE Camera are supported by Visionscape FrontRunner. All FrontRunner tools can be used in different **color planes**.

Specific color-related functionality is available in the form of custom scripts. The white balance of the color cameras can also be set from FrontRunner.

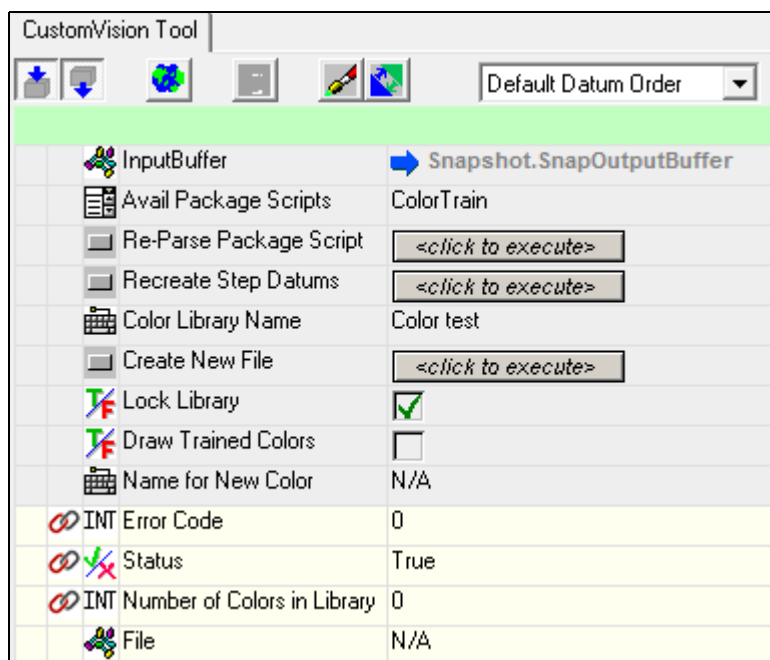
A set of Perl scripts is provided with the Visionscape installation that can perform analysis of color images. The scripts are automatically available in Visionscape and can be inserted into a job by navigating to the **Script** tab in the **Insert Step** dialog and selecting **Custom Vision Tool**.



Then, from the Custom Vision Tool Input datums, select the desired script by clicking the **Available Package Script**:



The tool will then be available:



ColorTrain and ColorID

These scripts are used together to train a library of colors and then check for those specific colors at runtime.

ColorTrain

Purpose: Allows you to create libraries of trained colors.

Use:

- Drag and size the ROI over a solid color area that you wish to train.
- Select the “Color Library Name” datum and enter the name of an existing color library, or the name you wish to use for a new color library. If you want to create a new color library, check the “Create New File” datum.

Note: Color Libraries are always saved in the **\Microscan\Visionscape\Jobs\Fonts** folder with the extension OCL.

- To train a new color, enter the name of the color in the “Name for New Color” datum. Now train the Step, and your new color will be added.
- Check the “Draw Trained Colors” datum, and all trained colors will be drawn in the upper left corner of the image.
- Check the “Lock Library” datum if wish to prevent any more colors from being trained.

The **ColorTrain** tool has the following input and output parameters:

The screenshot shows the CustomVision Tool interface with the ColorTrain tool selected. The tool's parameters are listed in a table, and several are annotated with callouts:

Parameter	Value
InputBuffer	Snapshot.SnapOutputBuffer
Avail Package Scripts	ColorTrain
Re-Parse Package Script	<click to execute>
Recreate Step Datums	<click to execute>
Color Library Name	Color test
Create New File	<click to execute>
Lock Library	<input type="checkbox"/>
Draw Trained Colors	<input type="checkbox"/>
Name for New Color	Blue
INT Error Code	0
Status	True
INT Number of Colors in Library	2
File	C:\Microscan\W\scape\Jobs\fonts\Color test.ocl

Annotations and their corresponding parameters:

- Filename of the color library in c:\microscan\jobs\fonts (points to Color Library Name)
- Create a new file instead of appending to existing (points to Create New File)
- Allow no further colors to be trained (points to Lock Library)
- Show trained color names in the image (points to Draw Trained Colors)
- Train the new color under this name (points to Name for New Color)
- Number of colors trained in this file (points to INT Number of Colors in Library)

Training is done by positioning the ROI (region of interest) around the color area, entering a name for the new color, and clicking the Train button. This is an example of the

ColorTrain Tool Setup and Inspection:

The screenshot shows the ColorTrain tool interface. The main window displays a color image of a box of pins. A red square ROI is positioned over a green pin. The ROI is labeled "Training ROI". The "Train button" is located in the top right corner. The "Trained colors" list is in the top left corner, showing: Red (1), Green (2), Yellow (3), White (4), and Blue (5). The "ColorTrain" tool name is visible in the top left corner of the main window. The status bar at the bottom shows "blorBayer8(B,G)Color(H,S)" and "31:31:36 (170:26:26)".

ColorID

Purpose: Attempts to identify a color by comparing the pixels within its ROI to the trained colors in a specified color library.

Use:

- Select the color library that will be used by the Step by specifying its name in the “Color Library Name” datum. If you have not created any color libraries, refer to the next section on the ColorTrain script for a description of how to create one.
- The Step inspects the average color value within its ROI, and compare it to various colors trained in the library. The name and index of the closest color match will be output by the tool.
- The “Color Distance Threshold” can be used to specify how close a color must be to the trained value in order to be considered a match. Decrease this value to require a closer match, increase the value to allow a looser match.
- The Step can be made to check for a specific color, such that it will pass if the color is detected, and fail if any other color is detected. Check the “Color Match” datum to activate this mode, and then enter the name of the color to check for in the “Match to Color” datum.
- Check the “Show Graphics” datum if you want the name of the detected color to be drawn in the image at runtime.

ColorSpotCheck

Purpose: Allows you to train on a single color in RGB or HSI. The Step will then pass only if the same color in RGB or HSI is detected in future images.

Use:

- Position the inner, smaller ROI of the Step over the color you wish to train on, and hit the Train button. The average Red, Green, and Blue values and Hue Saturation and Intensity values, within the ROI will be trained and stored in the Tool.
- Position the outer, larger ROI over the pixels to be inspected at runtime. During each cycle, the Step will pass if the average color value of the pixels within the larger ROI roughly matches the trained color. Otherwise it will fail.
- Use the “Threshold” datum to adjust how closely the trained color must match the inspected color. Increase the value to allow a looser color match, and decrease the value to produce a closer color match.

The **ColorSpotCheck** tool has the following parameters:

The screenshot shows the ColorSpotCheck tool interface with various parameters and their values. Callouts provide additional context for several parameters:

- Process in Color Space :** RGB. Callout: "Use RGB or HSI color space".
- Difference Calculation :** Sum of Differences. Callout: "Calculation of difference between trained and actual".
- Lock Template :** ☐. Callout: "Do not allow retraining".
- Red or Hue Scale Factor :** 1,000. Callout: "Scale factors can be used to adapt the match to a different illumination condition. This should not be required if white balance is the same as at train time."
- Green or Saturation Scale Factor :** 1,000.
- Blue or Intensity Scale Factor :** 1,000.
- Output Scale Factor :** 1,000.
- Threshold :** 50. Callout: "Maximum difference for pass/fail".
- Trained Red or Hue Mean :** 128.
- Trained Green or Saturation Mean :** 111. Callout: "Trained values".
- Trained Blue or Intensity Mean :** 45.
- INT Error Code :** 0.
- Status :** True.
- INT Red or Hue Mean :** 120. Callout: "Measured average color for the ROI".
- INT Green or Saturation Mean :** 104.
- INT Blue or Intensity Mean :** 45.
- INT ROI Average Difference :** 15. Callout: "Color difference to the trained color".

Color space selection options:

This screenshot shows the options for color space selection:

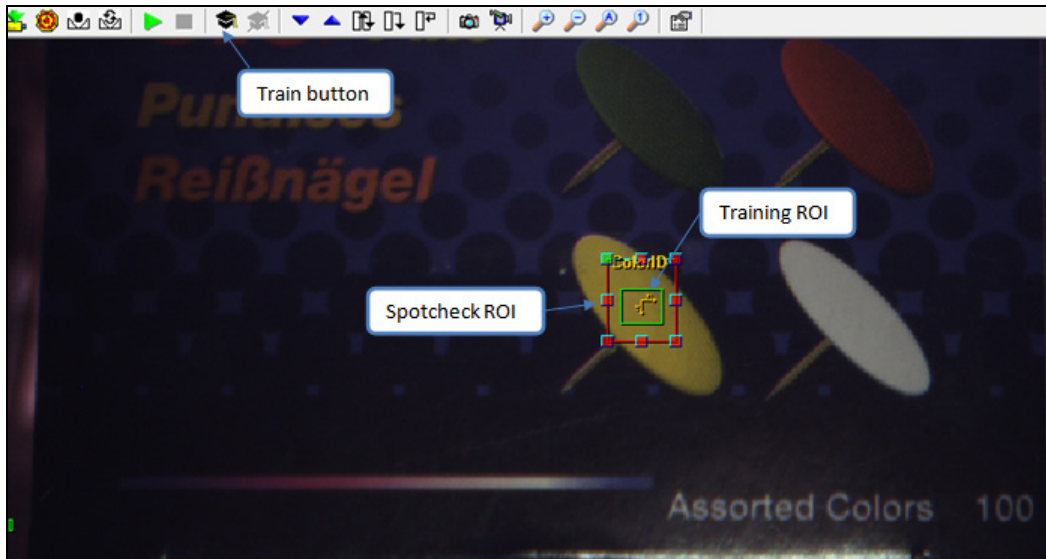
- Process in Color Space :** HSI (Red=0)
- Difference Calculation :** RGB
- Lock Template :** HSI (Red=0)
- Red or Hue Scale Factor :** HSI (Green=0)
- Green or Saturation Scale Factor :** HSI (Blue=0)

Difference calculation method options:

This screenshot shows the options for difference calculation method:

- Difference Calculation :** Sum of Differences
- Lock Template :** Sum of Differences
- Root of squares**

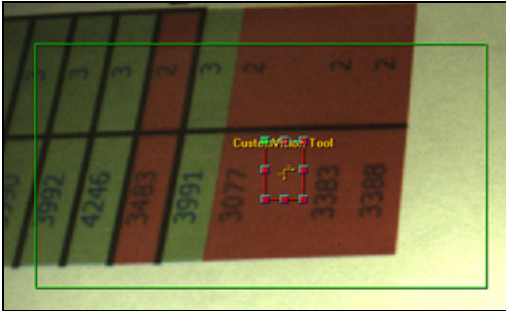
ColorSpotCheck Tool Setup and Inspection:



ColorDifference

Purpose: Outputs an image buffer that is the difference between the trained color and the current image. The resulting image should be near black when the pixels are close to the color that was trained, and will be brighter when they are a different color:

Color image, tool is trained on red:



Resulting output buffer:

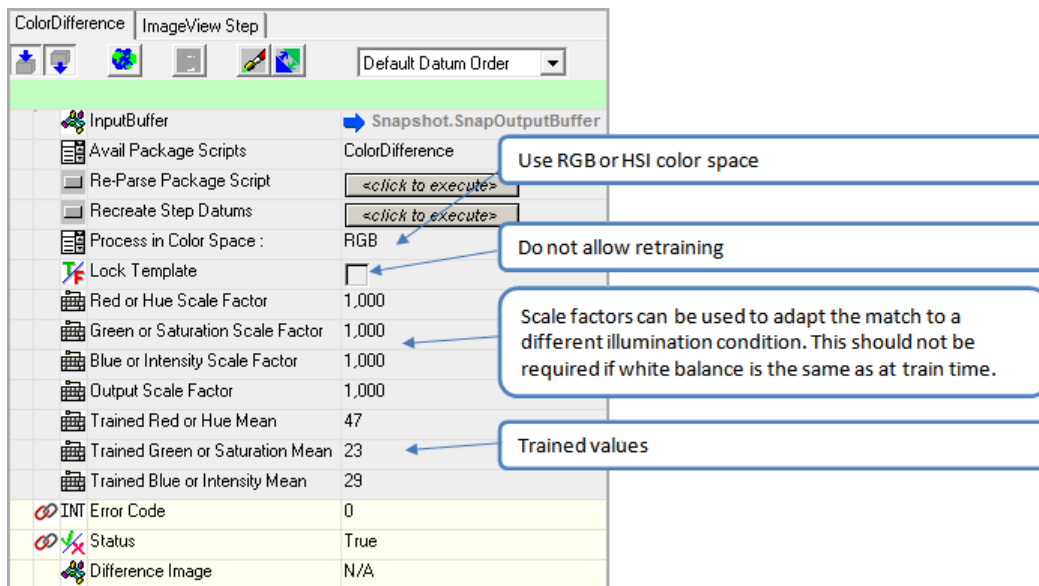


Note that the red pixels from the original image are black, whereas all other colors are brighter. This tool can be used to detect color changes within an area of an image.

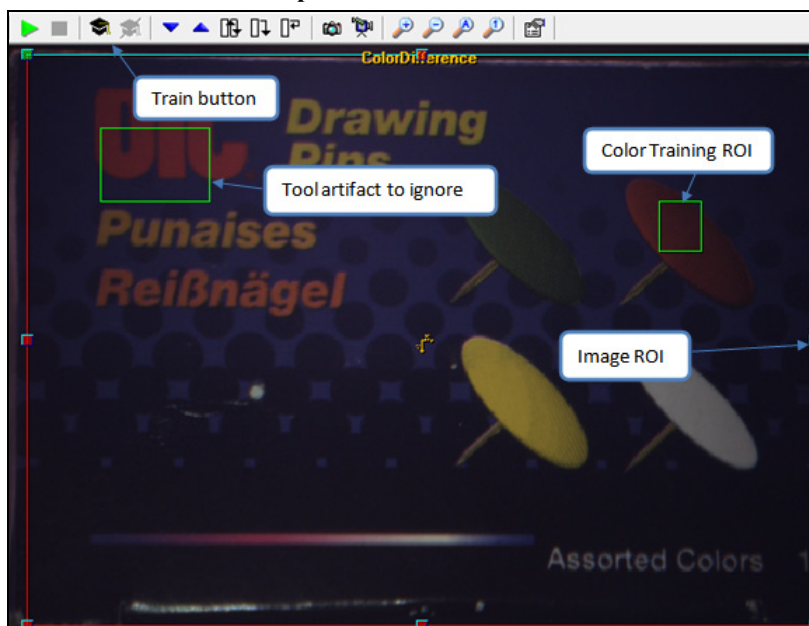
Use:

- This Tool provides two ROIs. The smaller, inner ROI is used to train on a color. The outer, larger ROI is used to specify the area of the image that you want to inspect. An output buffer will be produced that is the same size as this outer ROI.
- Position the inner ROI over a color and press the Train button in FrontRunner.
- Double-click within the outer ROI, and you will be shown the output buffer produced by this Tool. You can insert other steps within this output buffer, such as a Blob or Flaw Tool.

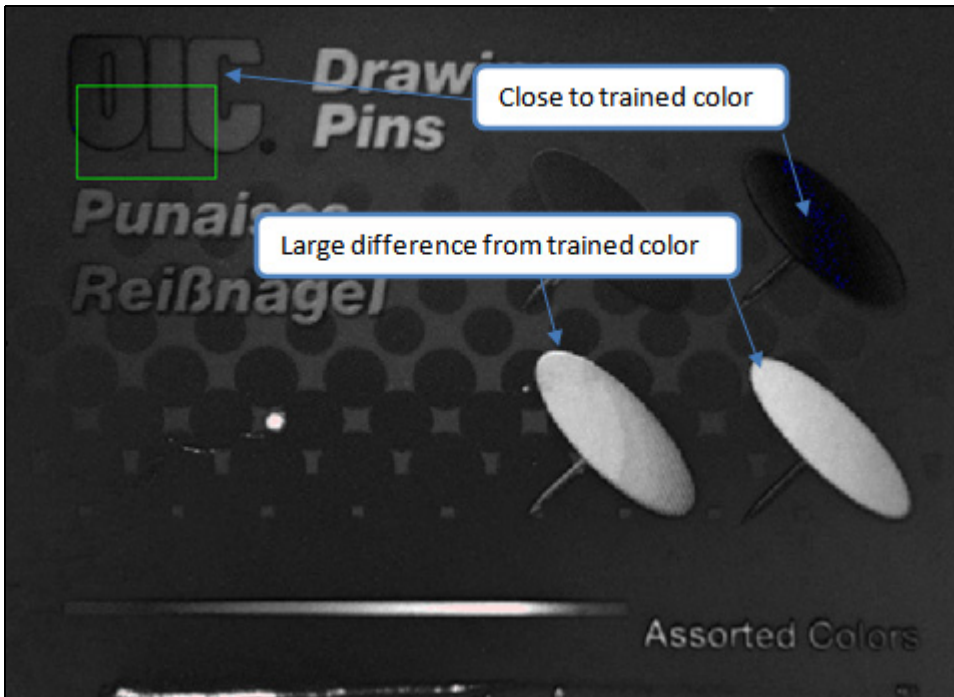
The **ColorDifference** tool has the following input parameters:



ColorDifference Tool Setup



ColorDifference Tool Output Image

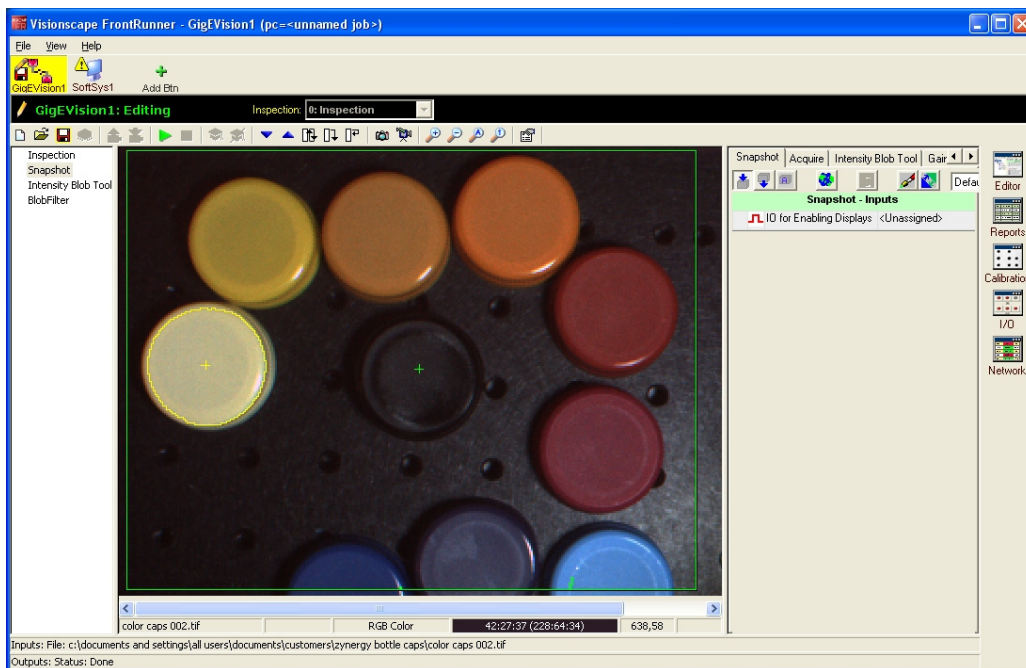


Color Image Display Options

Color Image Display in FrontRunner

The snapshot step shows the color image as acquired from the camera. The buffer manager control in FrontRunner shows both the RGB and HSI (red = 0) values under the cursor if the image acquired is a color image. Zooming fully into the image will show the individual RGB values for each pixel of a color image.

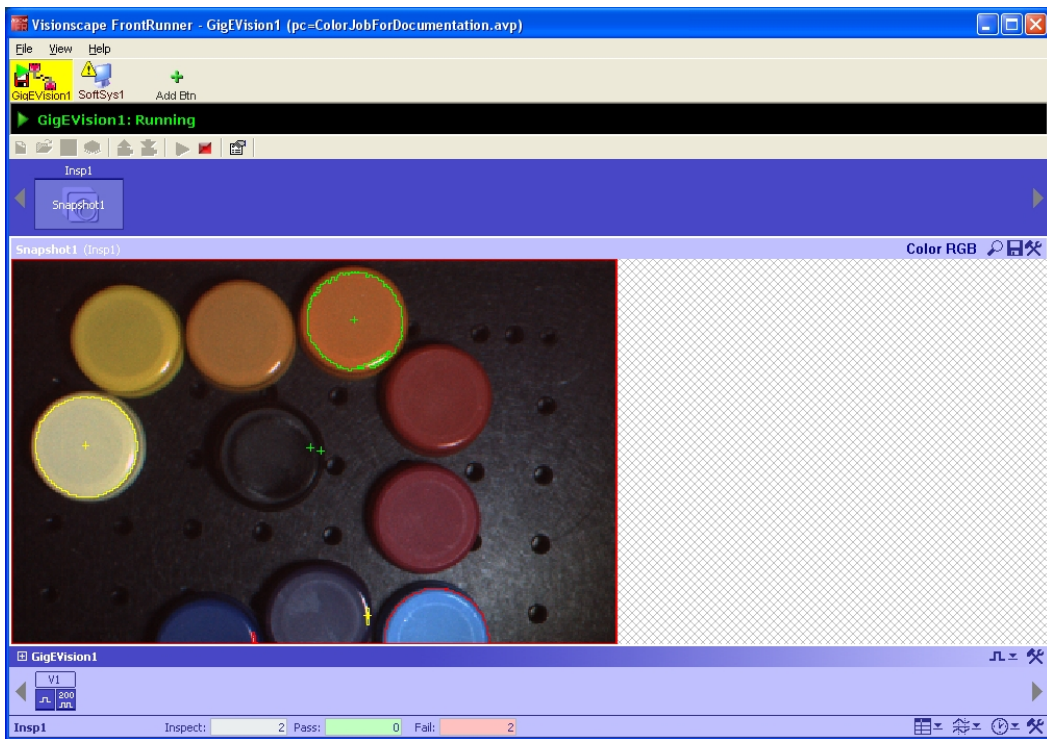
FIGURE 11–10. Color Buffer Display of Zoomed Image



Running Color Plane Selection

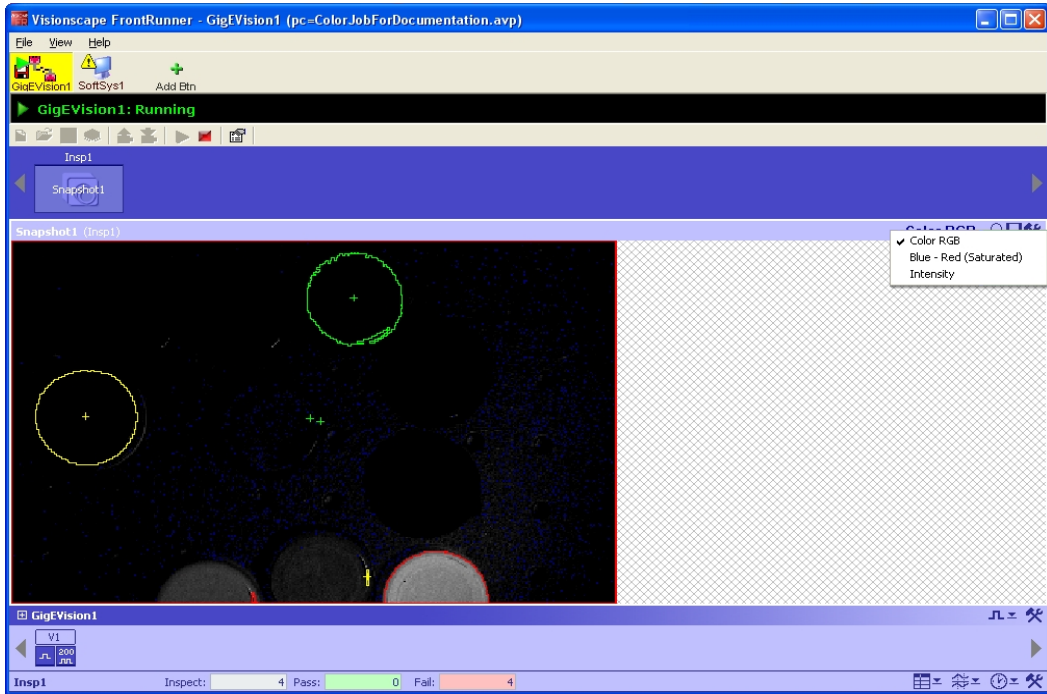
When using a color camera, the upper right of the image display will include the text of the current image plane displayed. The default will be the RGB color image. Channel selection will be limited to those channels used by vision tools that are inserted in the current snapshot.

FIGURE 11–11. Runtime Displaying Color Image



When the user selects the “RGB Color” text a dropdown menu will appear, allowing the user to select from the image channels used in the job.

FIGURE 11–12. Runtime Image Channel Selection Dialog



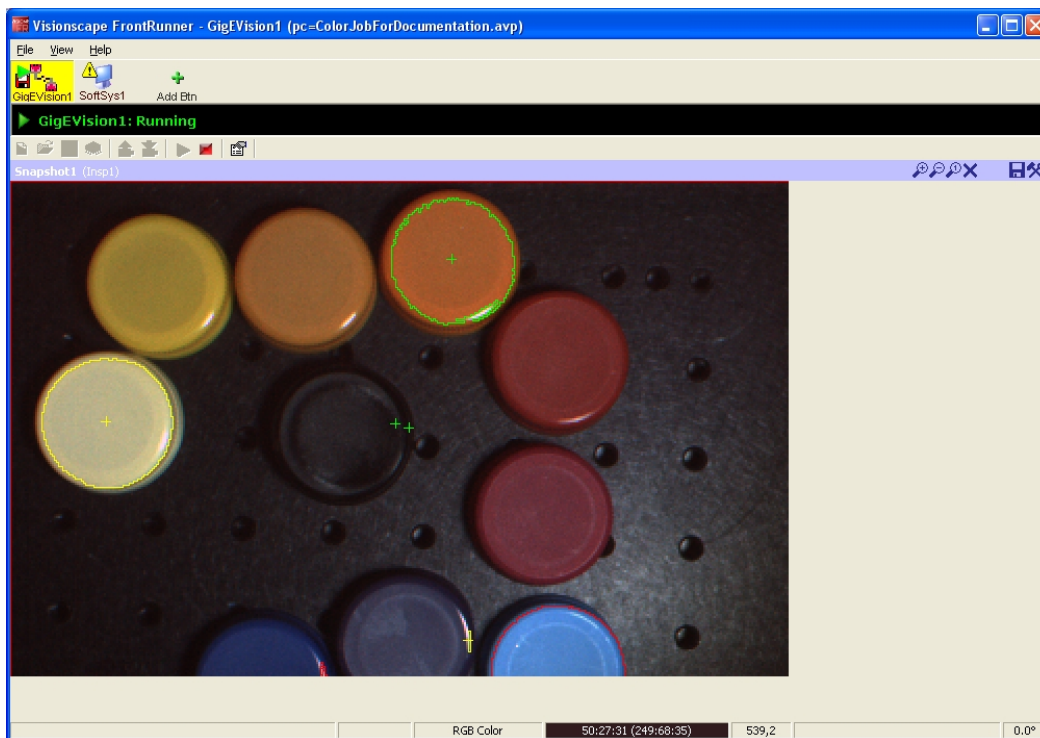
In the following example, the user has selected the “Intensity” image channel. At this point, the software will now display that channel.

FIGURE 11–13. Runtime Display of Intensity Channel



Finally, if the user clicks on the magnifying glass icon to zoom the image the name of the image channel zoomed will appear in the image status bar next to the pixel value and location.

FIGURE 11–14. Zoom Options with Color Runtime Display

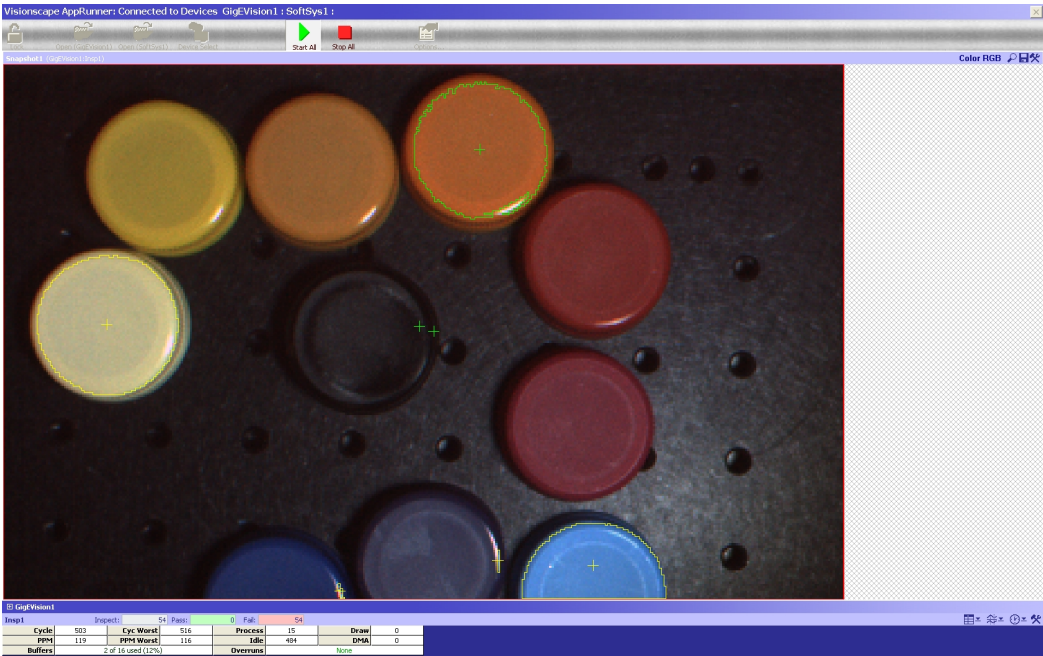


Note: The user will be able to select what image channel to display but ALL the enabled graphics for the job will be displayed regardless of which plane they are running in.

Color Image Display in AppRunner

The snapshot step shows the color image as acquired from the camera. The buffer manager control in AppRunner shows both the RGB and HSI (red = 0) values under the cursor if the image acquired is a color image. Zooming fully into the image will show the individual RGB values for each pixel of a color image.

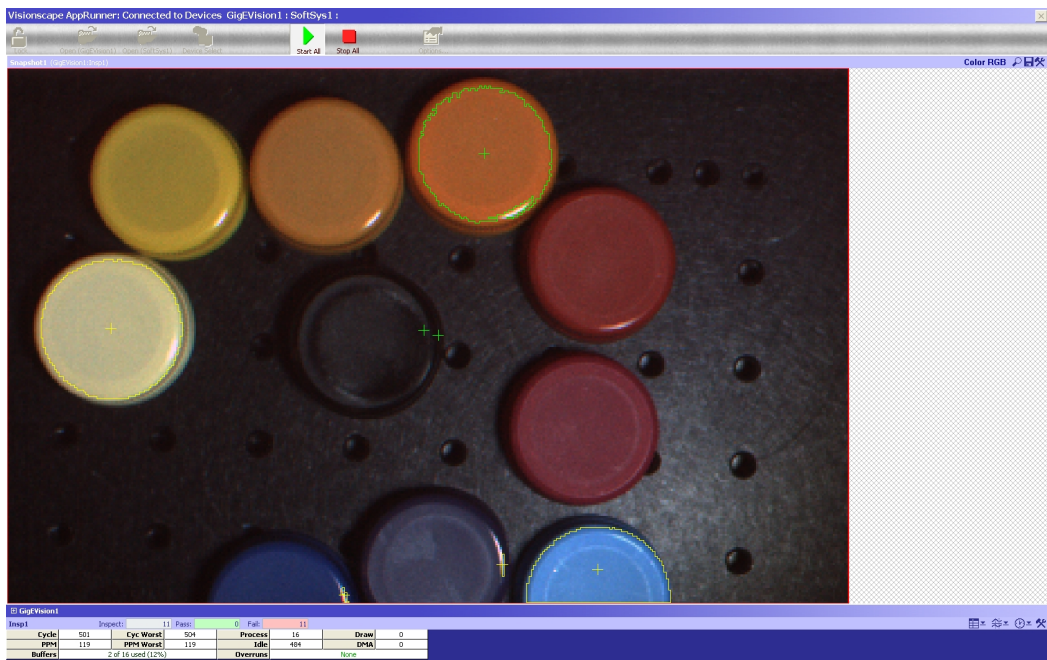
FIGURE 11–15. Color Buffer Display of Zoomed Image



Running Color Plane Selection

When using a color camera, the upper right of the image display will include the text of the current image plane displayed. The default will be the RGB color image. Channel selection will be limited to those channels used by vision tools that are inserted in the current snapshot.

FIGURE 11–16. Runtime Displaying Color Image



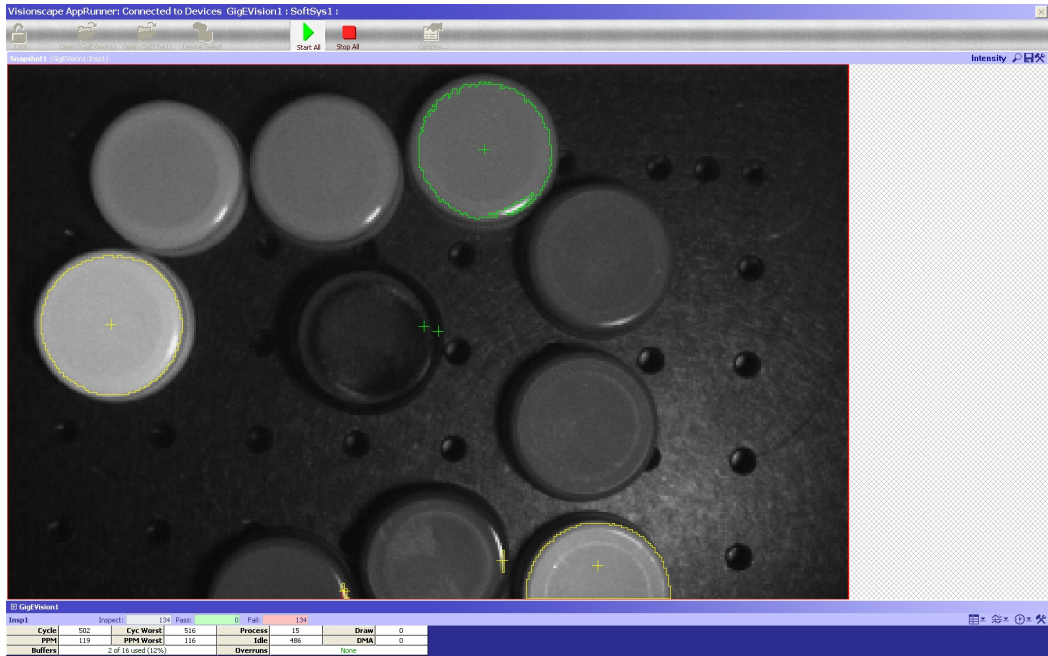
When the user selects the “RGB Color” text a dropdown menu will appear, allowing the user to select from the image channels used in the job.

FIGURE 11–17. Runtime Image Channel Selection Dialog



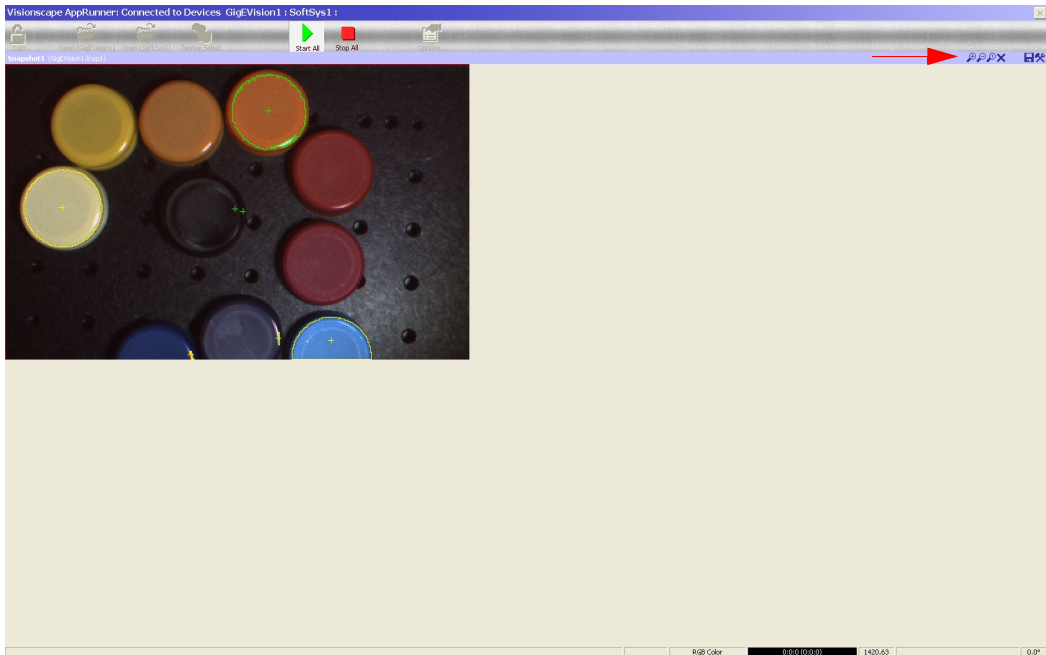
In the following example, the user has selected the “Intensity” image channel. At this point, the software will now display that channel.

FIGURE 11–18. Runtime Display of Intensity Channel



Finally, if the user clicks on the magnifying glass icon to zoom the image the name of the image channel zoomed will appear in the image status bar next to the pixel value and location.

FIGURE 11–19. Zoom Options with Color Runtime Display



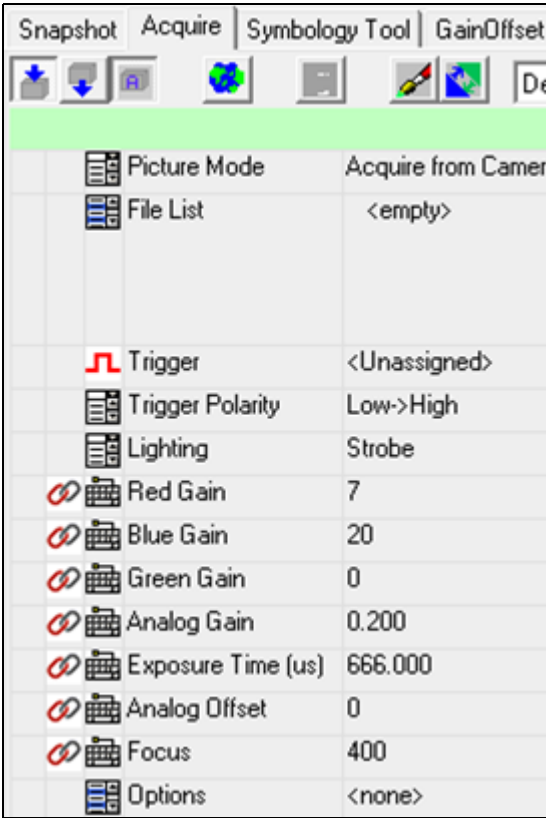
Note: The user will be able to select what image channel to display but ALL the enabled graphics for the job will be displayed regardless of which plane they are running in.

White Balance

This section outlines the white balance functionality for color smart cameras in FrontRunner.

White Balance Gain Values

The color channel gain values can be viewed by clicking the **Snapshot** step, selecting the **Acquire** tab and then activating the **Advanced** parameters as shown below:



The screenshot shows the 'Acquire' tab in the software interface. The 'Advanced' parameters section is expanded, showing a list of camera settings. The 'Red Gain', 'Blue Gain', and 'Green Gain' parameters are highlighted in red, blue, and green respectively. The 'Trigger' parameter is set to '<Unassigned>' and the 'Trigger Polarity' is set to 'Low->High'. The 'Lighting' parameter is set to 'Strobe'. The 'Red Gain' is 7, 'Blue Gain' is 20, and 'Green Gain' is 0. The 'Analog Gain' is 0.200, 'Exposure Time (us)' is 666.000, 'Analog Offset' is 0, and 'Focus' is 400. The 'Options' parameter is set to '<none>'.

Snapshot	Acquire	Symbology Tool	GainOffset
	Picture Mode	Acquire from Camera	
	File List	<empty>	
	Trigger	<Unassigned>	
	Trigger Polarity	Low->High	
	Lighting	Strobe	
	Red Gain	7	
	Blue Gain	20	
	Green Gain	0	
	Analog Gain	0.200	
	Exposure Time (us)	666.000	
	Analog Offset	0	
	Focus	400	
	Options	<none>	

The parameters are **Red Gain**, **Blue Gain**, and **Green Gain**. These values can be manually adjusted for optimal color fidelity or by using the white balance calibration operation outlined in the next section.

White Balance Calibration

Before running white balance calibration, place a white object such as a piece of paper in front of the camera at the current focus plane. Then initiate the white balance operation by selecting the white balance icon shown below. The color channel gain is then equalized such that the white object appears white.



After white balance calibration, the white balance gain values are updated and the results are saved as customer parameters.

Restore Preset White Balance Configuration

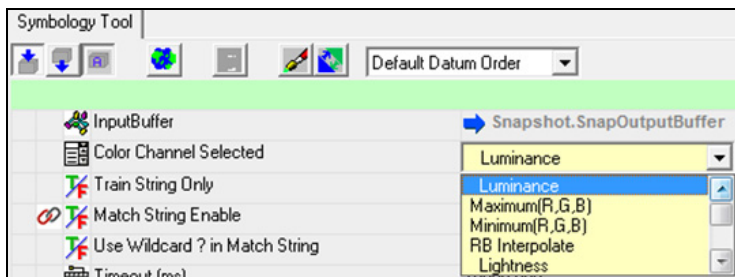
The color smart camera is pre-configured with factory calibrated white balance settings. To restore the color channel gain to these preset values, select the preset white balance icon as illustrated below:



After this operation, the white balance gain values are restored to the factory preset values and saved as customer parameters.

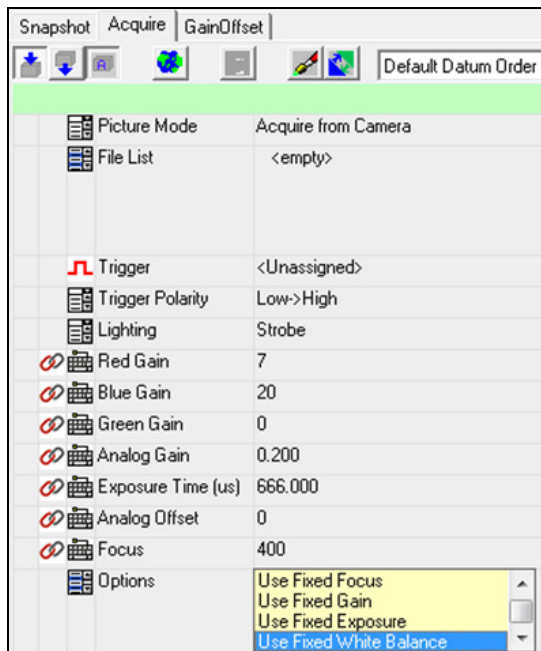
Color Channel Options

When a step is inserted in FrontRunner, the **Color Channel** or **Interpolation** operation can be selected or applied to the step as shown below:



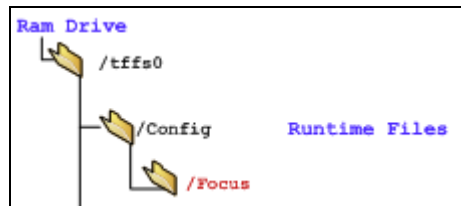
Use Fixed White Balance

The color channel gain values can be fixed by clicking the **Snapshot** step, selecting the **Acquire** tab and then activating the **Advanced** parameters. To fix white balance parameters, select the **Options** dropdown and click the **Use Fixed White Balance** option as show below:



Device parameters are referred to as camera parameters that are saved outside of a job, such that they can be applied globally or independent of a job as well as updated outside a job. These parameters include photometry settings (gain and exposure), focus, and white balance gain.

- A new job will always inherit the current “device data” so you do not need to re-calibrate the device.
- The device data for focus, photometry, white balance, and dimensional calibration exists in two places at all times:
 - In the job (copied from the device data at job creation);
 - On the device’s global /Config flash folder.



- Whenever the device parameters are calibrated, two things happen:
 - The device global data is updated in the /Config flash folder;
 - The job loaded in RAM is updated with the new data.
- The **Use Fixed White Balance** option controls whether the device parameters (white balance gains) are updated when the job is loaded from flash or downloaded to RAM with the device-wide values, or if the values last saved in the job are used instead.
 - **Normal:** Device parameters are updated when the job is loaded from flash or downloaded to RAM with the device-wide parameter values (from the **acqcfg** file).
 - If a job is opened on the PC or from a flash job slot on the device, and if a device parameter is unlocked, the value saved in the global device parameter file (acqcf) is used.
 - Job device parameter value (RAM) = Global device parameter value (/Config flash folder).

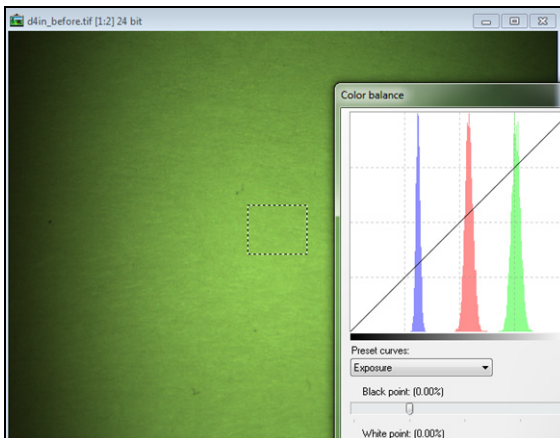
- Whenever an “unlocked” device parameter is updated, two things happen:
 - The device global data in RAM is updated;
 - The job loaded in RAM is updated.
- When a job is saved to a job slot, the device global data file in the /Config flash folder is updated.
- **Fixed:** Values last saved in the job are used.
- If a job is opened on the PC or from a flash job slot on the device, and if a device parameter is locked, the value saved in the job is used.
 - Job device parameter loaded in RAM = Job device parameter opened (Flash Slot/PC).
 - The global device parameter file is untouched.
- Whenever a “locked” device parameter is updated, the job loaded in RAM is updated.

White Balance Implementation

White balance is a processing operation performed to ensure proper color fidelity in a captured digital image. The image sensor does not detect light exactly as the human eye does, and so some processing or correction of the detected image is necessary to ensure that the final image realistically represents the colors of the original image. Proper white balance is required to take into account the “color temperature” of the light source, which refers to the relative “coolness” of white light. The main purpose of white balance as it relates to the camera is to render neutral colors correctly (gray/white) and to provide consistent color results.

Factory pre-set white balance calibration should be satisfactory for most applications, but the color smart camera allows for user adjustment or calibration of the white balance to account for exposure to different lighting conditions.

Before white balance:



After white balance:

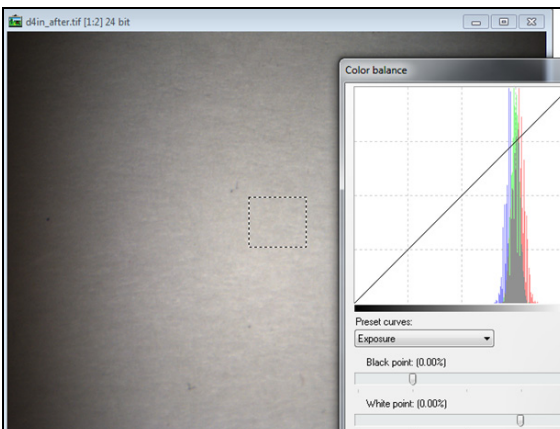


Image Transform Tools

The Image Transform tools generate a result image by applying a geometric transformation on the pixels within the input image ROI. The result image is usually of the same size as the source ROI. These transformations are:

- **Annular Unwrap Warp** — The result image is generated by unwrapping an annular ROI in the source image. You use this tool when the features to detect are arranged around an annulus region of the image. For example, text on CDs or ball-bearing inspections where the spheres must be inspected or counted.
- **Cylinder Unwrap Warp** — The result image is generated by unwrapping an image on a cylindrical surface, reducing the distortion caused by the surface.
- **Projection Tool** — The result image is of one pixel high. Data in this image can be specified as a projection of all pixels along the direction of the rotation handle of the ROI. This tool is most useful when looking for boundaries between objects along a specified direction. For example, I.C. lead spacing.
- **Rect Warp** — The result image is generated by rotating, translating and/or scaling the source ROI. Separate scaling for x and y is available. This tool corrects rotation and translation of the part and is, therefore, placed inside a dynamic location tool. Another use for Rect Warp is to reduce the number of pixels to process by applying a scale factor in x and/or y.

With the exception of the Annular Unwrap Warp, the result image system of coordinate is connected to the source ROI by a geometric transform object. Then, features calculated in the result image can be compared with features calculated in the input image.

Image transform tools do not generate any features that may be used by geometry or measurement tools.

References

Annular Unwrap Warp

This step takes an annular-shaped segment of an image as input. It applies an operation to the pixels, resulting in a new image in which the original pixels have been stretched into a rectangular shape. Depending on the setup, this image warping can also include scaling and/or axis inversion.

Other Steps Used

None.

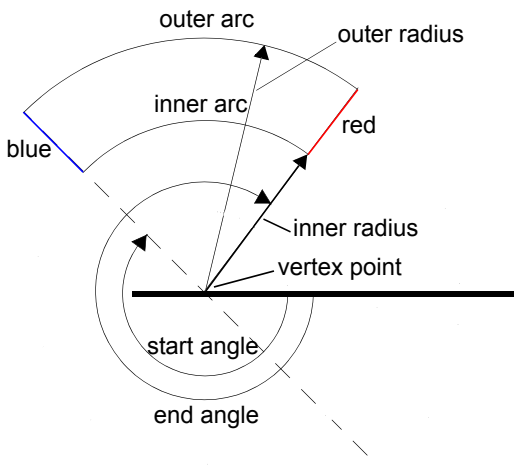
Theory of Operation

AnnularUnwrap Warp modifies an image so that marks that are seen in a circular pattern can be operated on as if they were made along a straight line. It is also useful for correcting rotation and translation of such an image from one part to the next. An annular ROI section of the input image is corrected or warped, and the resulting image placed into a new Buffer.

Annular ROI

An annular shape has four sides, two radial sides and two concentric arcs, as shown in Figure 12–1.

FIGURE 12-1. Annular Shape

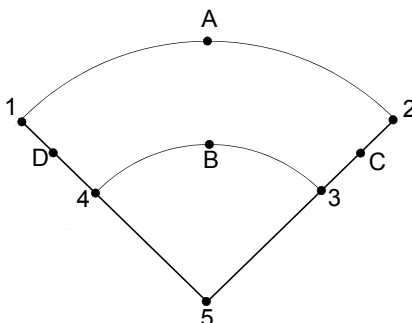


The two straight sides, after being extended, should converge at the vertex point of the annular shape. The blue-colored side marks the start angle of the annular area. The red-colored side marks the end angle of the annular. Both angles are measured clockwise from the x-axis (horizontal) of the buffer coordinates.

The vertex is the circular center of the two concentric arcs. The graphics display of the vertex can be hidden or shown by opening the Context Menu and selecting **Show Vertex**. Right-click anywhere inside the Image Display to display the Context Menu.

The annular ROI can be sized by dragging on one of its nine handles shown in Figure 12-2.

FIGURE 12-2. Annular ROI Handles



The handles are:

- Handles A and B — Allow adjustment of the radii while maintaining the start and end angles unchanged.
- Handles C and D — Allow adjustment of the end and start angles individually while maintaining the arc radii unchanged.
- Handle 1 — Allows simultaneous move of the outer radius and the start angle.
- Handle 2 — Allows simultaneous move of the outer radius and the end angle.
- Handles 3 and 4 — Allow adjustment of the curvature of the annular shape. These are useful when the vertex is placed out of the Image Display.
- Handle 5 — Allows the adjustment of the curvature. This handle can be shown, hidden, or placed outside the buffer screen.

Note: All handles, except the vertex, must remain within the buffer screen.

Unwrap Process

The unwrap action is such that the pixels along the outer arc, ordered from the start angle to the end angle clockwise, are placed on the first row of the output image, ordered from left to right. The provisions for Flip Vertical and/or Invert Vertical can place the output image in the desired orientation.

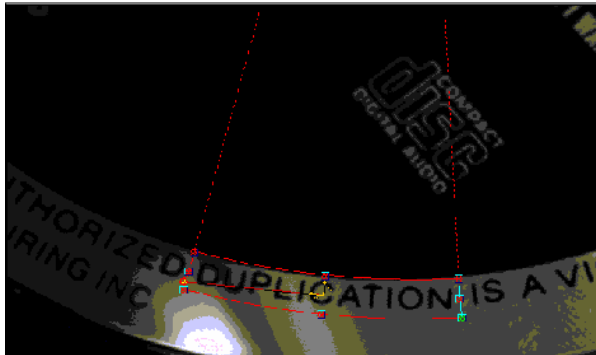
Use of the Unwrap Warp

Typically, other tools and steps are placed in its output image where the image pixels have been unwrapped into a rectangle, and where part rotation and translation have been corrected. In the new image, the tool ROI positions remain fixed from part to part.

In addition, if the position of the input ROI is adjusted by registering its position with known features in the input image, then part rotation and translation can be corrected dynamically. This is performed by combining an NPt Locator with the AnnularUnwrap Warp. In that sense, the component tools and steps contained in the output image of AnnularUnwrap Warp are dynamically located based on features detected by the NPt Locator. Refer to “NPt Locator Tool” on page 4-33 for more information.

The ROI that defines the pixels to warp can be adjusted by moving, sizing and rotating the search area shape associated with an AnnularUnwrap Warp. The input image to the warp operation is shown Figure 12–3.

FIGURE 12–3. Input Image to a Warp Operation — Example



In the image, the position of the AnnularUnwrap Warp ROI is determined by the setting of the vertex point, the start angle and range, and the inner and outer radii. The output image of the warp operation corresponding to the input ROI WarpInput is shown in Figure 12–4.

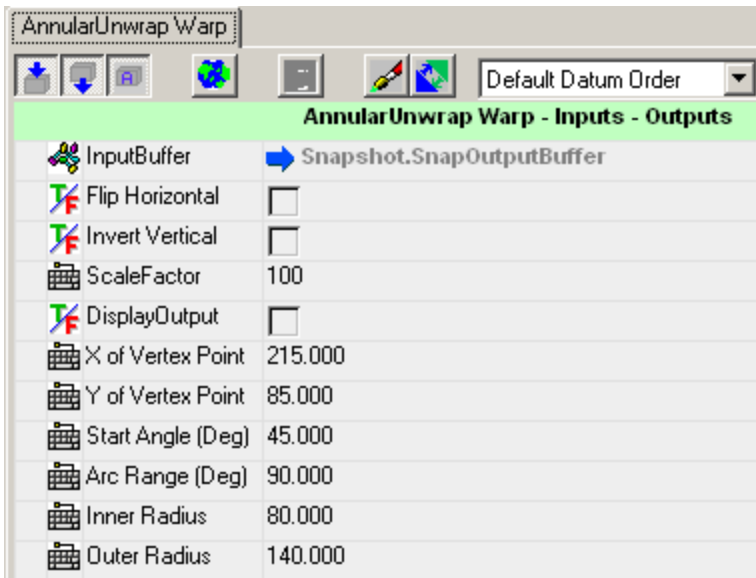
FIGURE 12–4. Output Image of the Warp Operation — Example



Description

AnnularUnwrap Warp allows editing through the AnnularUnwrap Warp properties page, as shown in Figure 12–5.

FIGURE 12-5. AnnularUnwrap Warp Properties Page



Settings

- **Flip Horizontal** — Activates horizontal flip of the output image. By default, this is disabled.
- **Invert Vertical** — Activates vertical inversion of the output image. By default, this is disabled.
- **ScaleFactor** — Reduces or enlarges the output image. Selecting a scale factor less than 100% can improve the performance of the tools placed within the step since there are less pixels to process.

Default: 100% (Normal size, no zooming)

Range: 50% (half as big) to 2000% (20 times as big as the original image within the input ROI)

- **DisplayOutput** — By default, this is disabled.
- **X of Vertex Point / Y of Vertex Point** — Allows the movement of the ROI on the input image without dragging it in the setup view. When the X and Y values of the vertex point change, the entire annular ROI moves along with

those changes. The default value of the vertex point is (216,85), and it can be moved anywhere within the boundary of the input image.

Range: -10000 to 10000

- **Start Angle (Deg)** — Allows the sizing of the arc of the annular shape. The start angle defines the angle from the vertex point to the top left corner of the ROI. To unwrap an entire circle, the start angle can be set to 0° and the arc range to 360°.

Start Angle — Default: 45°
Range: 0° to 360°

- **Arc Range (Deg)** — Allows the sizing of the arc of the annular shape. The Arc Range defines the range of angle from that start angle to the end angle, where the end angle is the angle from the vertex point to the bottom right corner of the ROI. To unwrap an entire circle, the start angle can be set to 0° and the arc range to 360°.

Arc Range — Default: 90°
Range: 0° to 360°

- **Inner Radius/Outer Radius** — Allows the height of the annular shape to be set up. The outer radius is the distance from the vertex point to the top left corner of the ROI. The inner radius is the distance from the vertex point to the bottom right corner of the ROI. The values are limited such that the outer radius is at least 4 greater than the inner radius. For example, if the inner radius is 100, the outer radius will be 104. Conversely, if the outer radius is 100, and you set the inner radius to 99, the software will set the inner radius to 96.

Inner Radius — Default: 80
Range: 0 to 10000

Outer Radius — Default: 140
Range: 80 to 10000

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.
- *Unwrap Out Buffer* — Contains the corrected image.

I/O Summary

AnnularUnwrap Warp provides a I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: HorizFlip: aaa VertFlip: bbb Scale: ccc%

Where: aaa = True or False
bbb = True or False
ccc = The % of scale being used

Cylinder Unwrap Warp

This step is an image-in, image-out operation, and unwraps an image on a cylindrical surface, reducing the distortion caused by the surface.

Other Steps Used

Custom Vision Tool — The Cylinder Unwrap step is based on a Custom Vision Tool. You must insert a Custom Vision Tool and select the Cylinder_UnWarp script.

Theory of Operation

Given a description of the geometry of a cylinder, the CylinderUnwrap step will warp the image on the cylinder in such a way as to unwrap the image onto a flat surface. This reduces the distortion caused by the cylindrical surface.

The geometry of the cylinder is specified using the:

- Radius of the cylinder
- Distance the cylinder is from the camera
- Vertical axis of the cylinder
- Point within the ROI where the image is correct (not distorted)

Cylinder Unwrap ROI

The Cylinder Unwrap ROI is a rotatable rectangle. Typically, the rectangle is rotated to match the angle of the cylinder axis.

Using the Cylinder Unwrap Warp

Typically, other tools and steps are placed in its output image where the image pixels have been unwrapped into a rectangle. This is useful for studying features or text on a cylinder that will be distorted closer to the edges of the cylinder.

The ROI that defines the pixels to warp can be adjusted by moving, sizing and rotating the search area shape associated with a CylinderUnwrap Warp.

Figure 12–6 and Figure 12–7 show an input image and the corresponding output image for a Cylinder Unwrap Warp. The cylinder axis input is set to the output of a BisectLines Meas step, which is the line which bisects the left and right edges of the cylinder.

FIGURE 12–6. Input Image to Cylinder Unwrap Operation Example

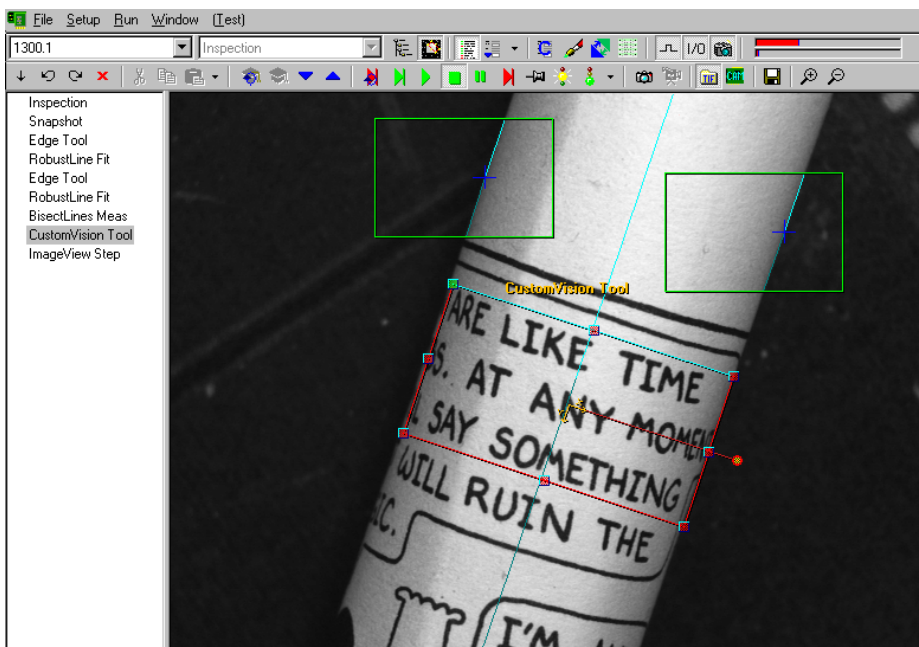
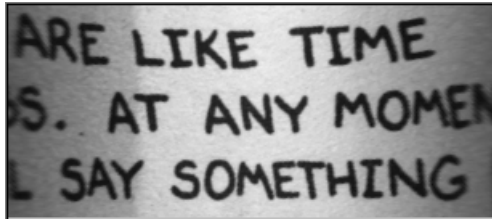
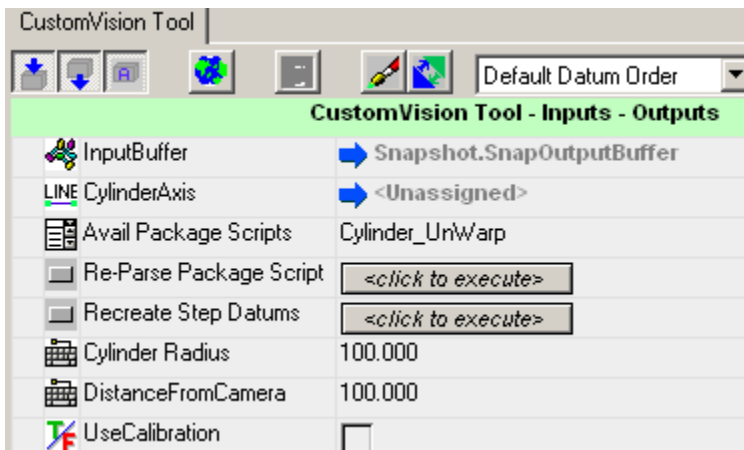


FIGURE 12-7. Output Image to Cylinder Unwrap Operation Example

Description

CylinderUnwrap Warp allows editing through the CylinderUnwrap Warp properties page, as shown in Figure 12-8.

FIGURE 12-8. Cylinder Unwrap Warp Properties Page

Settings

- **Cylinder Axis** — An input line datum that is selectable. This input line is usually the bisecting line of the right and left edges of the cylinder. The point within the ROI where the image is correct (i.e., the point of no distortion) should lie along the cylinder axis.
- **Re-Parse Package Scripts** — This button causes the package script to be parsed when clicked. Whenever any changes to the script are made, this button needs to be clicked to make these changes take effect.

- **Recreate Step Datums** — This button causes the input and output datum lists in the step to be recreated. This button only needs to be clicked when a datum is either added, removed or changed in the script. If the script is changed, but no input or output datums are changed, then this button does not need to be clicked. Clicking this button also causes all input datums to be set to their default values and lose their connections to other step results or parameters.

Datums created by the package script are added to the user interface. Input datums are shown as a box with a drop-down list button. The input datums can be linked to other similar type datums in the Job. Clicking the drop-down list button causes the Job Tree to be displayed, allowing you to select the datum to link to the input datum. Resource datums are shown as user-editable boxes that can be set to a value directly. Output datums are not shown in the user interface for this step, but can be seen in the Job Tree that comes up when linking an input datum.

- **Cylinder Radius** — The radius of the cylinder.
- **DistanceFromCamera** — The distance from the camera to the cylinder.
- **UseCalibration** — If the inputs (CylinderRadius, DistanceFromCamera) are specified in calibrated/world coordinates (for example, millimeters), then the UseCalibration checkbox should not be checked. If the inputs are specified in pixel coordinates, then UseCalibration should be checked.

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.
- *CylUnwrapped Image* — The modified image.

I/O Summary

None.

Projection Tool

This tool compacts a specified region of input image into one-dimensional data. The one-dimensional data can be chosen to be one of three types of averages of all pixels along the projection direction. The results are stored in both a one-row image buffer and an integer list.

Other Steps Used

Rect Warp — Used internally to the Projection Tool. Rect Warp will not visibly rotate the image in the FrontRunner window.

Theory of Operation

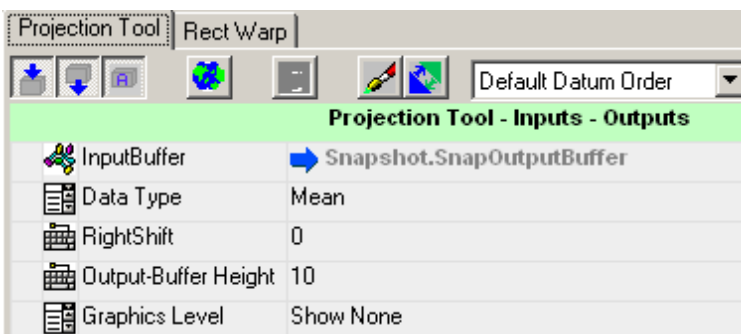
The Projection Tool transforms a region of an input image (ROI) into a one-dimensional image by calculating a user-specified average among all pixels along the projection direction. You can adjust the orientation of the ROI to obtain desired projection direction. The projection is always along the direction of the ROI's rotation handle. The output data, stored in the output image and an integer list, can be selected to be one of three types of averages over a row of the ROI. For display purposes, the output buffer can be set to any height. All rows in the output buffer are identical.

The projected data can also be graphically displayed on the input image of this tool.

Description

Projection Tool allows editing through the Projection Tool properties page, as shown in Figure 12–9.

FIGURE 12–9. Projection Tool Properties Page



Settings

- **Data Type** — Allows selection of three types of averages:
 - **Mean (Default)** — This is an approximate arithmetic average. That is, the sum of all pixels in a row divided by the number of pixels in the row.
 - **Automatic Average** — This is the result of the sum of all pixels in a row divided by 2 multiple times. This multiple is automatically calculated by the step. This method of averaging generally yields smaller values than the Mean.
 - **Choice Average** — Specifies the multiple to right-shift (to divide by 2 multiple times) the sum. The datum RightShift is provided for this purpose.
- **RightShift** — Allow the specification of the multiple used when **Data Type** is set to Choice Average.

Default: 0

- **Output-Buffer Height** — Specifies the output buffer height. All rows in the output buffer will be identical. This parameter and the width of the Projection ROI are kept in sync; changing the ROI width with the mouse or changing this parameter in the Tool property page changes the ROI as well. If the Projection width is fixed but the projected image needs to be more visible for placing tools on it, use the zoom tool.

Default: 10 pixels

- **Graphics Level** — Enables or disables the graphics for this tool. Although five settings are available for this property, there are in effect only three levels:
 - **Show Basic Graphics** — The projected data graphics are shown in the tool's input buffer.
 - **Show Details** — The projected data graphics are shown in the tool's input buffer.
 - **Show Details and Mask** — The projected data graphics and mask are shown in the tool's input buffer.
 - **Show None (Default)** — No graphics are shown for this tool.
 - **Show ROI Only** — The ROI is shown in green when the step passes. The ROI is shown in red when the step fails.

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.
- *Projection Data* — This result contains a list of integer values corresponding to the projection along each scan line of the ROI. They will be equal to the gray value of each column of pixels in the output image.
- *Output Buffer* — Contains an output image whose width is the same as the height of the ROI, and whose height is specified by **Output-Buffer Height**. The result is also stored in an integer list that may be used for data processing.

I/O Summary

Projection Tool provides a I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: Output Height: aaa

Where: aaa = Height in pixels of the output buffer.

Outputs: ProjectionCount: bbb

Where: bbb = Height in pixels of the ROI.

Rect Warp

This step applies an operation to the pixels in an image, which has the effect of rotating and possibly scaling the original image pixels. This is called image warping or image re-sampling.

Other Steps Used

None.

Theory of Operation

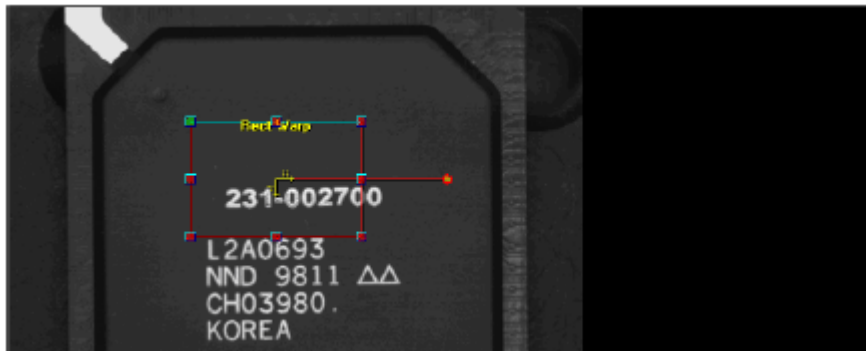
Rect Warp corrects an image for part rotation and translation. A quadrilateral section of the input image is corrected or warped and the resulting image placed into a new buffer. The correction applied is such that the top side of the input ROI corresponds to the output image first row and the top-left corner pixel of the input ROI corresponds to the first pixel in the output image.

Typically, other tools and steps are placed in its output image where the part rotation and translation have been corrected. In that new image the tool ROI positions remain fixed from part to part.

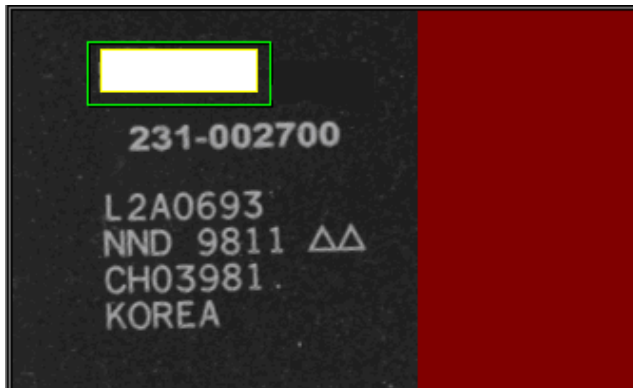
In addition, if the position of the input ROI is adjusted by registering its position with known features in the RectWarp input image, then part rotation and translation can be corrected dynamically. This is done by combining an NPt Locator with the Rect Warp. In that sense, the tools and steps contained in the output image of Rect Warp are dynamically located based on features detected by the NPt Locator. Refer to “NPt Locator Tool” on page 4-33 for more information.

Rect Warp allows selection of the buffer to work on from the list of currently available buffers. The default buffer will be the output buffer of its parent. This is usually the output buffer of the closest enclosing Snapshot but can be the buffer of any step that generates an output image including another Rect Warp.

The ROI that defines the pixels to warp can be adjusted by moving, sizing and rotating the search area shape associated with the Rect Warp, as shown in Figure 12–10. The default size of the ROI is 150x100 with no rotation.

FIGURE 12–10. Rect Warp

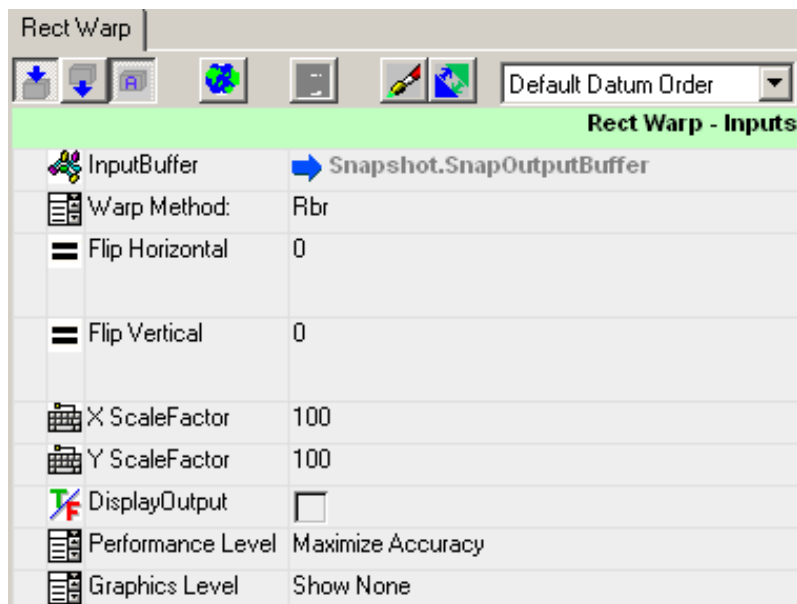
The output image of the warp operation corresponding to the input ROI Rect Warp is shown in Figure 12–11.

FIGURE 12–11. Image Warped

Description

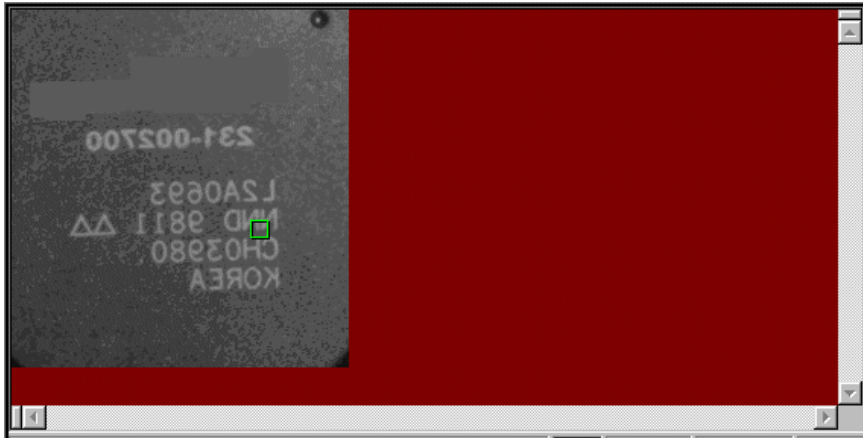
Rect Warp allows editing through the Rect Warp properties page, as shown in Figure 12–12.

FIGURE 12–12. Rect Warp Properties Page



Settings

- **Warp Method** — Selects the type of warp to perform:
 - **Rbr (default)** — Rectangular warp. The ROI can be sized and rotated but remains rectangular.
 - **Rhombus** — Quadrilateral warp. The corner points of the ROI can be moved independently.
- **Flip Horizontal** — Activates horizontal flip of the output image. By default, this property is disabled by setting the expression to 0. Any valid expression that evaluates to true (1) will cause the effect shown in Figure 12–13.

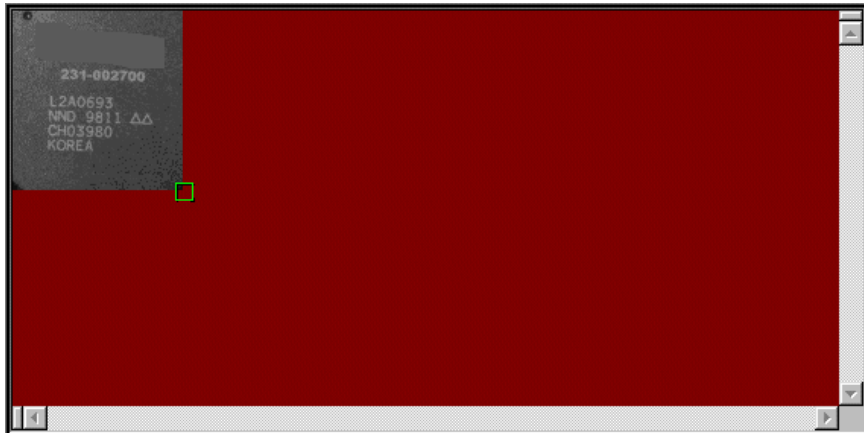
FIGURE 12–13. Flip Horizontal

- **Flip Vertical** — Activates vertical flip of the output image. By default, this property is disabled by setting the expression to 0. Any expression that evaluates to true (1) will cause the effect shown in Figure 12–14.

FIGURE 12–14. Flip Vertical

- **X ScaleFactor/Y Scale Factor** — Reduces or enlarges the output image. The maximum value 16000% is 160 times as large as the original image within the input ROI. Selecting a scale factor less than 100% can improve performance of the tools placed within it since there are fewer pixels to process. The effect when set to 50% is shown in Figure 12–15.
Default: 100%
Range: 50% to 16000%

FIGURE 12–15. Scale Factor



- **DisplayOutput** — Displays the warped image on the target display. Once this image is displayed, it is overwritten by the snapshot image when there is one in the inspection. This feature is only useful when the Direct Display setting is turned Off, which can be performed by deselecting **Activate the target display** in the Runtime window.
- **Performance Level** — Can be set to maximize accuracy or maximize speed. When you select maximize speed, the Warp tool will special case 0°, 90°, 180° and 270° rotation cases to use a simpler and less accurate warp method. This should not be used if accurate sub-pixel gauging is to be performed in the Warp output image.

Training

No special training is required for this tool other than setting the size and position of the input ROI.

Results

- *Status* — Set to true after a successful execution of the step.
- *Warp Output Buffer* — Contains the corrected image.

I/O Summary

Rect Warp provides a I/O summary in the Status Bar located at the bottom of the FrontRunner window.

Inputs: HorizFlip: aaa VertFlip: bbb ScaleX: ccc% ScaleY: ccc%

Where: aaa = True or False
 bbb = True or False
 ccc = The % of scale being used

Image Processing Tools

“Image Processing” refers to tools that modify the gray levels of images. All image processing tools have a common type of output: an image that can be further processed or analyzed.

Visionscape provides the following image processing tools:

- BinaryMorph Filter
- FrameAverage Filter
- Gain and Offset
- GrayMorphFilter
- ImageArith
- MeanLP Filter
- Sobel Filter
- Binarized Color Thresholding Filter
- Zooming and Binning

Among the image processing tools, BinaryMorph Filter, GrayMorph Filter, MeanFilter, and Sobel Filter are further classified as Image Enhancement tools. Image Enhancement uses neighborhood operators to modify the gray levels of the image. As enhancement is performed, each result pixel’s gray level is a mathematical function of the gray levels of the pixels in the immediate neighborhood (i.e., surrounding pixels).

The ImageArith tool can manipulate images arithmetically or logically. It can add, subtract, multiply, or perform logical operations, on a pixel-by-pixel basis, between images or between different regions of a single image. It can also binarize, copy, mask and shift, etc., on each pixel of a single input image.

The GainOffset Filter tool, a special case of the ImageArith tool, applies a gain and an offset to each pixel of the input image.

The image processing tools do not generate any features that may be used by geometry or measurement tools.

BinaryMorph Filter

This step is an image-in, image-out operation that applies a binary morphology operator on an input region and produces an output buffer. All pixels in the output buffer have the value 0 or 255. Binary morphology operations consist of erosions and dilations. A polarity switch allows operations on either light or dark objects.

When **Polarity** is set to:

- Light, erosion means erosion of white, and light objects become smaller when the operator is applied. When the dilation operator is applied, white objects will become larger.
- Dark, erosion means erosion of dark. In this case, dark objects become smaller when erode operator is applied, and larger when the dilation operator is applied.

For the purpose of this document, a pixel is considered:

- On
when **Polarity** = LIGHT and its value is 255, or
when **Polarity** = DARK and its value is 0
- Off
when **Polarity** = LIGHT and its value is 0, or
when **Polarity** = DARK and its value is 255

Other Steps Used

None.

Theory of Operation

BinaryMorph Filter processes an input region by replacing each pixel with a function of its neighbors. The standard binary morph operations supported are:

- Erosion
- Weak erosion
- Shape erosion
- Noise erosion
- Dilation

- Weak dilation
- Shape dilation
- NOP (no operation)

In addition, you may specify a binary morph lookup table to map a source 3x3 pixel neighborhood to a result 3x3 neighborhood to achieve any binary morphology function desired.

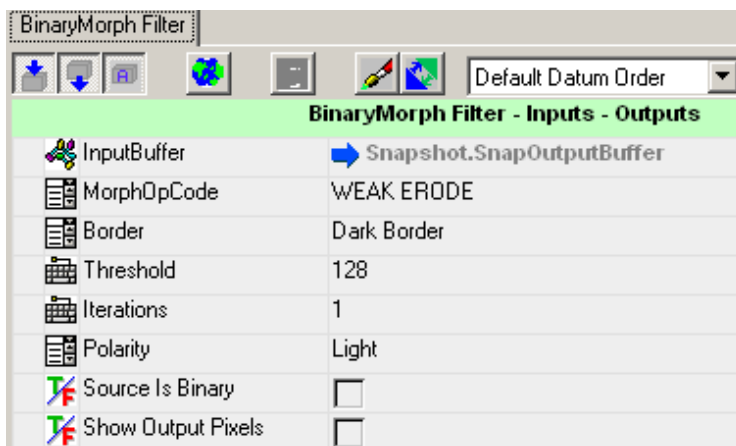
The basic setup for BinaryMorph Filter is shown in Figure 13–1, and consists of parameter setup for threshold, number of iterations, morphology function, value of the border pixels, and polarity of the input image.

Note: When Training, modified pixels are not displayed within the image until a tool is placed within the BinaryMorph Filter step.

Description

BinaryMorph Filter allows editing through the BinaryMorph Filter properties page, as shown in Figure 13–1.

FIGURE 13–1. BinaryMorph Filter Properties Page



Settings

- **MorphOpCode** — The morphology operation (function) that this step performs when run. It is a list box containing allowable values for this setting:
 - **Erode (Default)** — Apply the AND operator to the neighborhood such that the pixel at the center of the neighborhood is eroded (set to Off) if any of the pixels in the neighborhood are Off.
 - **Weak Erode** — Apply the AND operator to the neighborhood such that the pixel at the center of the neighborhood is eroded (set to Off) only if 2 or more pixels in the neighborhood are already Off. This operator is like a diluted erosion.
 - **Shape Erode** — Apply the AND operator to the neighborhood such that the pixel at the center of the neighborhood is eroded (set to Off) if it is located along the contour of the object. This allows an object to erode while maintaining its original shape.
 - **Noise Erode** — Erode single and sparse pixels groups.
 - **Dilate** — Apply the OR operator to the neighborhood such that the pixel at the center of the neighborhood is dilated (set to On) if any of the pixels in the neighborhood are On.
 - **Weak Dilate** — Apply the OR operator to the neighborhood such that the pixel at the center of the neighborhood is deleted (set to On) only if 2 or more pixels in the neighborhood are already On. This operator is like a diluted full dilation.
 - **Shape Dilate** — Apply the OR operator to the neighborhood such that the pixel at the center of the neighborhood is dilated (set to On) if it is located along the contour of the object. This allows an object to dilate while maintaining its original shape.
 - **NOP** — The output matches the input, no changes are done.
- **Border** — The value of virtual pixels surrounding the region being processed. Border pixels are used in 3x3 neighborhood operations of real pixels that are on the edge of the input region. Border values are either **Dark** or **Light**.

Default: Dark

- **Threshold** — Is a value from 0 to 255. This value determines whether a pixel is On or Off, depending on the **Polarity** (Normal/Reverse) selected. With Normal polarity, pixels greater than or equal to this value are treated as On. With Reverse polarity, pixels greater than or equal to this value are treated as Off, and pixels less than this value are treated as On.
- **Iterations** — The number of times the morphology operation is performed on the input image. If you enter a value less than one, one iteration is performed.

Default: 1

- **Polarity** — The polarity of the objects being processed:
 - **Light** — A pixel above the threshold is considered On and a pixel below the threshold is considered Off. In this case, erosion means erosion of white – shrink light objects and dilation means dilation of white – grow light objects.
 - **Dark** — A pixel below the threshold is considered On and a pixel above the threshold is considered Off. In this case, erosion means erosion of dark – shrink dark objects and dilation means dilation of dark – grow dark objects.
- **Source Is Binary** — Flag indicating that the input image is a binary image (all pixels are either 0 or 1). When this flag is On, the threshold value does not have any effect.

Default: Disabled

- **Show Output Pixels** — When enabled, a flag indicating that the results of the step will be displayed on the output target monitor.

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.
- *Binary Morph Output Buffer* — Contains the morphed image data.

I/O Summary

BinaryMorph Filter provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Thr: xxx Iter: xxx OP: sss

Where: xxx = numeric value indicating actual value
 sss = string value indicated selected value

Error Message:

- Step Not Ready to Run

FrameAverage Filter

This step is an image-in, image-out operation that stores each image in a queue and calculates an average image from all saved images for output. Each pixel in the output buffer contains the average value of the corresponding saved images' pixels.

Other Steps Used

None.

Theory of Operation

FrameAverage Filter takes in each image passed to its input buffer and adds it to the end of its internal queue. For each pixel of all saved images, an average value is calculated and written to the output buffer image. This is intended to yield an image that can be compared to each image taken from a camera to highlight variations, i.e., due to variations in the hardware of the camera.

There are two phases of importing images and producing an average:

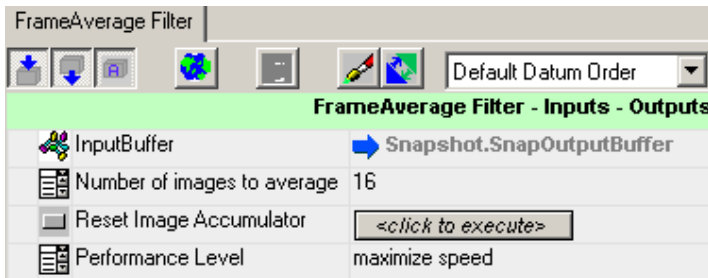
- The queue is not yet full — When the Job begins the queue will be empty. As it fills, new images will be added to the queue. All images read from the camera will be involved in calculating the average.
- The queue is full — Once the queue is full, the oldest image in the queue will be removed when another image is read.

Note: Images are saved internally in order that their contribution to the average frame can be subtracted when they are removed from the queue.

Description

FrameAverage Filter allows editing through the FrameAverage Filter properties page, as shown in Figure 13–2.

FIGURE 13–2. FrameAverage Filter Properties Page



Settings

- **Number of images to average** — This value dictates the size of the image queue. Initially, the values are limited to 2, 4, 8, 16, or 32. If this value is changed, the queue is empty before proceeding with the new queue size. Default is 16.
- **Reset Image Accumulator** — Clears the image queue and starts accumulating again from scratch. This is useful when environmental lighting changes significantly.
- **Performance Level** — Is either of the following:
 - *Maximum Speed* — Snapshot images are kept in the buffer pool. This mode is faster for snapshots since the image data is not copied at runtime. However, **Number of images to average** is limited by the size of the Buffer Counts specified in the property page of the system step. The step will automatically use 'Minimum Size' mode when used for non-snapshot images or during tryout. (Default)
 - *Minimum Size* — In this mode, a copy of the image data is made to perform the averaging operation. This does not require frames to be consumed from the buffer pool, but will slow runtime.

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.

- *Output Buffer* — A queue containing a user-defined number of the last images received. Its output is an image composed of the average gray values of all images in the queue.

I/O Summary

FrameAverage Filter provides information in the Status Bar located at the bottom of the Train window.

Inputs: Queue size limit: xx

Where: xx = is the user-defined queue size limit in the property page.

Outputs: Queue size: xx

Where: xx = is the number of images currently in the queue.

GainOffset Filter

The GainOffset Filter applies the user-specified gain and offset to its input buffer. The resultant image is put back to the same buffer.

Other Steps Used

None.

Theory of Operation

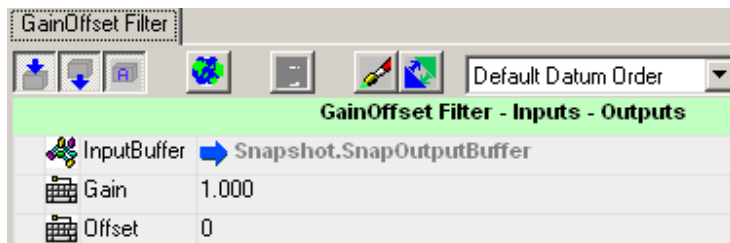
A simple arithmetic copy operation is applied to the input image. This is performed as a post-processing step, altering the pixels after the image has been digitized. The gain is a multiplier for every pixel in the buffer. The offset is added to every pixel in the buffer.

GainOffset Filter is independent of the camera and digitizer being used. However, there are camera and digitizer specific gains and offsets that can also be used.

Description

GainOffset Filter allows editing through the GainOffset Filter properties page, as shown in Figure 13–3.

FIGURE 13–3. GainOffset Filter Properties Page



Settings

- **Gain** — Multiplies every pixel in the buffer by the gain value.
Default: 1
Range: -10 to 10
- **Offset** — Adds the offset value to every pixel in the buffer.
Default: 0
Range: -3000 to 3000

Training

None.

Results

The image that has been modified by applying the gain and offset.

I/O Summary

GainOffset Filter provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Gain: aaa Offset: bbb

Where: aaa = displays value entered in properties page
bbb = displays value entered in properties page

Outputs: None.

GrayMorphFilter

This step is an image-in, image-out operation that applies a grayscale morphological operator on an input region and produces an output buffer.

Other Steps Used

None.

Theory of Operation

GrayMorph Filter processes an input region by replacing each pixel with a function of its neighbors. The simplest grayscale operations are erosion and dilation:

- Erosion — Setting the output pixel to the MIN of the pixels in the neighborhood.
- Dilation — Setting the output pixel to the MAX of the pixels in the neighborhood.

All other grayscale morphological operations are derived from MIN and MAX. The interpretation of MIN and MAX is based on the **Polarity** setting.

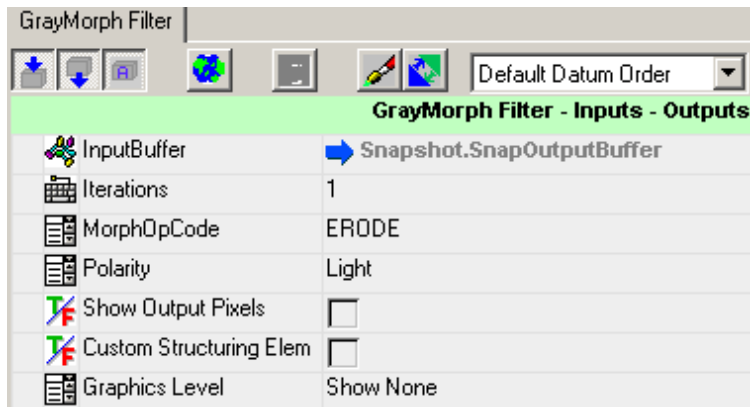
- When **Polarity** is set to *Light*, this means a correspondence of MIN to dark pixels and MAX to light pixels. Therefore, erosion, in this case, would shrink the light pixels, whereas dilation would grow the light pixels.
- When **Polarity** is set to *Dark*, this means a correspondence of MIN to light pixels and MAX to dark pixels. Therefore, erosion, in this case, would shrink the dark pixels, whereas dilation would grow the dark pixels.

Erosion and dilation are complementary. Eroding the light is the same as dilating the dark.

Description

GrayMorph Filter allows editing through the GrayMorph Filter properties page, with default settings, as shown in Figure 13–4.

FIGURE 13–4. GrayMorph Filter Properties Page — Default Setting



Settings

The GrayMorph Filter properties page, in default setting, allows the modification of all properties:

- **Iterations** — The number of times the morphological operator is performed on the input image. If you enter a value less than one, one iteration is performed.

Default: 1

- **MorphOpCode** — The morphological operator (function) that this step performs when run:
 - **ERODE (Default)** — Grayscale minimum function. The output pixel is set to the value of the minimum pixel in the neighborhood.
 - **DILATE** — Grayscale maximum function. The output pixel is set to the value of the maximum pixel in the neighborhood.
 - **OPEN** — Erosion followed by dilation.
 - **CLOSE** — Dilation followed by erosion.
 - **GRADIENT** — The output pixel is set to the difference between the maximum and the minimum pixels in the neighborhood. (MAX-MIN).
 - **TOPHAT** — The difference between the original image and the image after it has been OPENed (original minus OPENed).

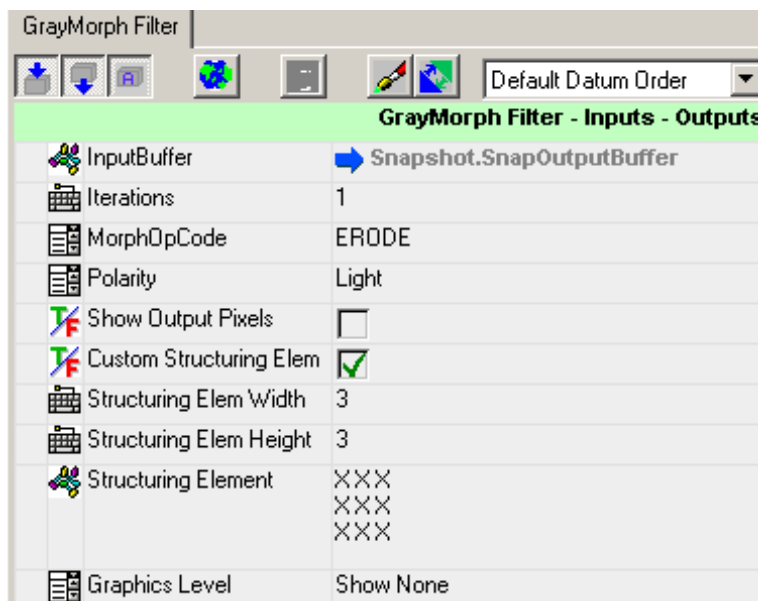
- **WELL** — The difference between the image after it has been **CLOSEd** and the original image. (**CLOSEd** minus original)
- **Polarity** — The polarity of the pixels to be processed. For example, when **Polarity** is set to **Light**, dilations will dilate light colored pixels:
 - **Light** — Pixels of interest are light colored.
 - **Dark** — Pixels of interest are dark colored.
- **Show Output Pixels** — Flag indicating that the results of the **GrayMorph** step will be displayed graphically on the output target monitor.
- **Custom Structuring Elem** — Allows you to specify a custom Morphology kernel that is different from the default 3x3 kernel.

Description

GrayMorph Filter may be further customized by enabling **Custom Structuring Elem**. This setting specifies the neighborhood of pixels that are included in the morphology operation. The default neighborhood consists of the current pixel, as the central pixel in the neighborhood and the eight surrounding pixels. The size of the structuring element neighborhood may be changed by selecting **Custom Structuring Elem** in the user interface, and individual pixels in the neighborhood may be included/excluded from the morphology computation simply by turning on/off the corresponding entry in the structuring element.

GrayMorph Filter allows editing of **Custom Structuring Elem** through the **GrayMorph Filter** properties page, as shown in Figure 13–5.

FIGURE 13–5. GrayMorph Filter Properties Page — Custom Setting



Settings

The GrayMorph Filter properties page, in custom setting, allows the modification of all properties as described in the default setting plus the following:

- **Custom Structure Elem** — Enables the use of a custom structuring element. Enable this datum to show the datums Structuring Element Height, Structuring Element Width, and Structuring Element; otherwise, these datums are hidden.
- **Structuring Element Width** — Indicates the width of the Structuring Element.
Default: 3
- **Structuring Element Height** — Indicates the height of the Structuring Element. Values less than 3 are invalid.
Default: 3
- **Structuring Element** — A two dimensional representation of the structuring element. A pixel beneath an entry that is On is processed accordingly. A pixel beneath an entry that is Off is left unchanged.
Default: 3x3 with all entries on

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.
- *Gray Morph Output Buffer* — Contains the morphed image data.

I/O Summary

GrayMorph Filter provides an I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs: Iter: xxx OP: sss Polarity: sss

Where: xxx = numeric value indicating actual value
 sss = string value indicated selected value

Output:

Error Message:

- Step Not Ready to Run

ImageArith

This step allows you to perform arithmetic operations on images.

Other Steps Used

None.

Theory of Operation

The arithmetic operations fall into two categories:

- One-operand — Operates on one input ROI. The one-operand operations are:
 - ABS
 - BINARIZE
 - COPY
 - MASKSHIFT
 - NEGATE
 - NOT
 - PASSRANGE
- Two-operand — Operates on two input ROIs on single or separate images. Two-operand operations are:
 - ADD
 - AND
 - DIFF
 - MAX
 - MIN
 - MULTIPLY
 - OR
 - SUB
 - XOR

Four types of processing may be applied to the input and/or the output image of the arithmetic operation:

- **Gain** — A floating point value, ranging from -255.0 to 255.0, is a multiplier to the gray value of each pixel in the image.
- **Offset** — A number, ranging from -255 to 255, is being added to the gray value of each pixel in the image.
- **Sign** — An image can be treated as unsigned or signed. For an unsigned image, the pixel values range from 0 to 255. For signed image, the pixel values range from -128 and 127. Converting an unsigned image to a signed image means treating the pixel values higher than 127 in the unsigned image as negative values in the signed image. The negative pixel value can be calculated in the following formula:

$$128 - [\text{Pixel Value in Unsigned Image}] + 1$$

- **Clip** — For an unsigned image, some pixel values may become larger than 255 after arithmetic operations or gain and offset processing. The term **Clip Overflow** describes setting the pixel value to 255 for any pixel where overflow occurs. For a signed image, the term **Clip Negative** describes setting the pixel value to 0 for any negative pixel value in the image.

ImageArith Tool produces an output image through three processing stages:

- **Pre-processing** — Applying any applicable Gain, Offset, Sign, and Clip operations on the input image before the selected arithmetic operation. For all operators of the ImageArith Tool, a Gain and Offset can be applied before the selected arithmetic operation is executed. Gain and Offset processing may cause overflow. When the input image is treated as unsigned, **Clip Input Overflow** is selectable. When the input image is treated as signed, **Clip Input Negative** is selectable. Selecting these decide the ImageArith Tool handling of overflow situations.
- The selected arithmetic operation on the pre-processed input image.
- **Post-processing** — Applying any applicable Gain, Offset, Sign, and Clip operations on the arithmetic-operated image. For all operators of the ImageArith Tool, an arithmetic-operated image can be applied to Gain and Offset processing. Gain and Offset processing may cause overflow. When the image is treated as unsigned, **Clip Output Overflow** is selectable. When the image is treated as signed, **Clip Output Negative** is selectable. Selecting these decide the ImageArith Tool handling of overflow situations.

Arithmetic operators COPY, ADD, SUB, and DIFF allow the input image and the arithmetic-operated image to be treated as signed or unsigned, and provide parameters for Sign and Clip processing.

For all other operators, the input images have to be treated as unsigned. Clip Input Overflow is provided. For these operators, the arithmetic-operated image is also treated as unsigned, but Clip Output Overflow depends on individual operators.

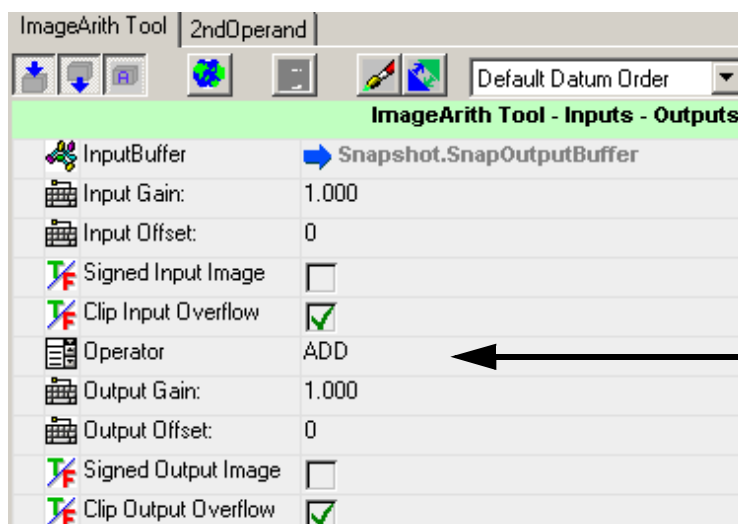
The remainder of the ImageArith section describes the following:

- “Operator Set to ADD, COPY, DIFF or SUB” starting on page 13-20
- “Operator Not Set to ADD, COPY, DIFF, or SUB” starting on page 13-24
- “Operator Set to BINARIZE or PASSRANGE” starting on page 13-25
- “Operator Set to MASKSHIFT” starting on page 13-26
- “Operator Set to Any Two-Operand Operation” starting on page 13-27

Operator Set to ADD, COPY, DIFF or SUB

ImageArith Tool allows editing through the ImageArith Tool properties page, with default settings, as shown in Figure 13–6.

FIGURE 13–6. ImageArith Tool Properties Page



- **Input Gain / Input Offset** — Specify the gain and/or offset to be applied to the input image before the arithmetic operation.

Input Gain range: -255 to 255 (allows for fractional numbers)

Input Offset range: -255 to 255 (using whole numbers only)

- **Signed Input Image and Clip Input Overflow** — When the input image is unsigned, you can clip overflows when they occur. When the input image is signed, this parameter name changes to **Clip Input Overflow**. **Clip Input Overflow** default is enabled.
- **Operator** — 15 operator settings allow arithmetic operations to be applied to the pre-processed input image.

Where: a and b represent the input pixel values in the first and the second pre-processed input buffers, respectively.
 C represents the arithmetic-operated output value for the corresponding pixel.

One-Operand Operators

- **ABS** — This operator treats its input pixels as signed to perform the Absolute operation on the pixel.

$C = (255 - a) + 1$ if the input pixel value is less than 0;

$C = a$ if the input pixel value is greater than 0.

Both input images are unsigned. **Clip Input Overflow** is provided. Arithmetic-operated image is as unsigned.

- **BINARIZE** — When selected, BinarizationSetup is inserted into the Job by the ImageArith Tool. BinarizationSetup provides editing for the Low Threshold and High Threshold, as shown in Figure 13–8, “BinarizationSetup Properties Page,” on page 13-25.

Values range from 0 to 255.

The binarization operation is:

$C = 1$, if a falls on the range from Low Threshold and High Threshold.

$C = 0$, if a falls outside the range from Low Threshold and High Threshold.

Both input images are unsigned. **Clip Input Overflow** is provided.
Arithmetic-operated image is as unsigned.

- **COPY** — The default setting.

$C = a$.

Input image is unsigned or signed.

Clip Input Overflow is provided when the input image is unsigned

Clip Input Negative is provided when the input image is signed.

Arithmetic-operated image is unsigned or signed.

Clip Output Overflow is provided when the arithmetic-operated image is unsigned.

Clip Output Negative is provided when the arithmetic-operated image is signed.

- **MASKSHIFT** — When selected, MaskShiftSetup is inserted into the Job by the ImageArith Tool. The MaskShiftSetup step allows you to specify a mask value to bitwise-operate on each pixel. The bitwise operation can be either AND or OR. You can also specify a left shift by a negative entry in the Shift field, or a right shift by a positive entry in the Shift field. The shift should be a number from -7 to 7. The shift operation, multiplies (left shift) or divides (right shift) the masked result by 2 the number of shift times.

Mask denotes the mask value, shift denotes the number of shift, then,

$C = (a \& \text{mask}) \gg \text{shift}$ for mask AND and right shift;

$C = (a | \text{mask}) \gg \text{shift}$ for mask OR and right shift;

$C = (a \& \text{mask}) \ll \text{shift}$ for mask AND and left shift;

$C = (a | \text{mask}) \ll \text{shift}$ for mask OR and left shift;

Both input images are unsigned. **Clip Input Overflow** is provided.
Arithmetic-operated image is as unsigned.

- **NEGATE** — This operator treats its input pixels as signed and negates their values.

$C = (255 - a) + 1$.

Input image is unsigned.

Arithmetic-operated image is unsigned.

- **NOT** — $C = 255 - a$.

Input image is unsigned.

Arithmetic-operated image is unsigned

- **PASSRANGE** — When selected, BinarizationSetup is inserted into the Job by the ImageArith Tool. BinarizationSetup provides editing for the Low Threshold and High Threshold. The binarization operation is:

$C = \text{Low Threshold}$ if a falls on or below the Low Threshold.

$C = a$ if a falls inside the range from Low Threshold and High Threshold.

$C = \text{High Threshold}$ if a falls on or above the High Threshold.

Both input images are unsigned. Clip Input Overflow is provided. Arithmetic-operated image is as unsigned.

Two-Operand Operators

- **ADD, SUB, DIFF, MULTIPLY** — These operators perform arithmetic add, subtract, and difference on the input pixels of the two input ROIs, respectively.

$C = a+b$ for ADD

$C = a-b$ for SUB

$C = \text{abs}(a-b)$ for DIFF

$C = a*b$ for MULTIPLY

Both input images are unsigned or signed.

Clip Input Overflow is provided when the input images are unsigned.

Clip Input Negative is provided when the input images are signed.

Arithmetic-operated image is unsigned or signed.

Clip Output Overflow is provided when the arithmetic-operated image is unsigned.

Clip Output Negative is provided when the arithmetic-operated image is signed.

- **AND, OR, XOR** — Perform bitwise AND, OR, or XOR operation on the pixels from the two input ROIs.

$C = a\&b$ for AND

$C = a|b$ for OR

$C = a^b$ for XOR

Both input images are unsigned. Clip Input Overflow is provided.

Arithmetic-operated image is as unsigned.

- **MAX** — Compares the pixels of the two input ROIs making the output value the greater one of the two.

$$C = a \quad \text{if } a > b$$

$$C = b \quad \text{if } a < b$$

Both input images are unsigned. **Clip Input Overflow** is provided.
Arithmetic-operated image is as unsigned.

- **MIN** — Compares the pixels of the two input ROIs making the output value the smaller one of the two.

$$C = a \quad \text{if } a < b$$

$$C = b \quad \text{if } a > b$$

Both input images are unsigned. **Clip Input Overflow** is provided.
Arithmetic-operated image is as unsigned.

- **Output Gain and Output Offset** — Apply gain and offset to the arithmetically operated resultant image.

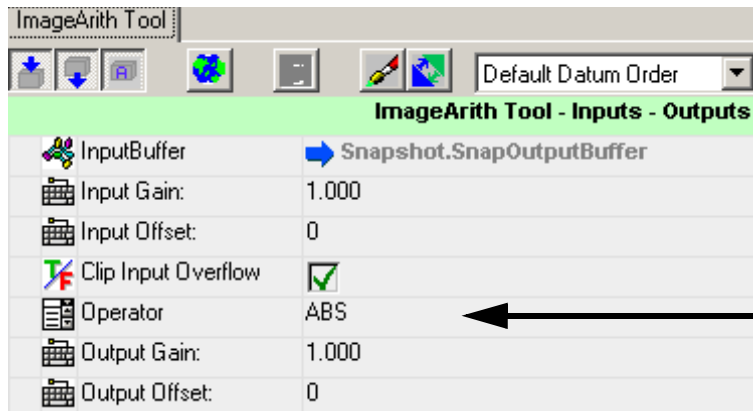
Output Gain	a floating point value from -255.0 to 255.0
Output Offset	a number from -255 to 255.

- **Signed Output Image and Clip Output Overflow** — Specify whether the arithmetic-operated image is signed or unsigned. When signed and the **Clip Output Negative** is checked, any negative value in the operated image will be clipped to 0. When unsigned and the **Clip Output Overflow** is checked, any value larger than 255 after the operation will be clipped to 255.

Operator Not Set to ADD, COPY, DIFF, or SUB

ImageArith Tool allows editing through the ImageArith Tool property page, as shown in Figure 13–7. For these operations, all input and output are treated as unsigned. The items in this page are a subset of those in the property page shown in Figure 13–6.

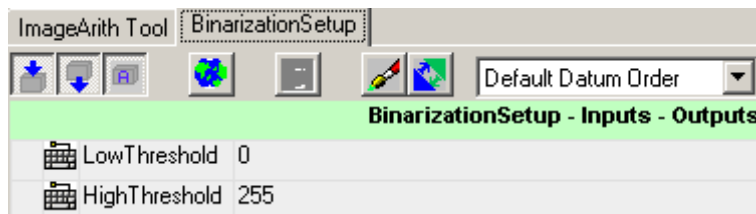
FIGURE 13-7. ImageArith Tool Properties Page



Operator Set to BINARIZE or PASSRANGE

When Operator is set to *BINARIZE* or *PASSRANGE*, a tab named *BinarizationSetup* is available for setting the thresholds, as shown in Figure 13-8.

FIGURE 13-8. BinarizationSetup Properties Page

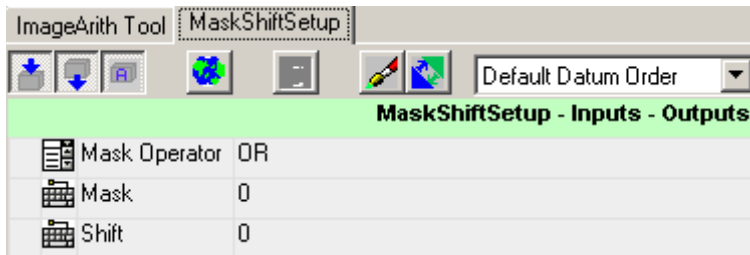


- **High Threshold / Low Threshold** — When Operator is set to *BINARIZE*, any value outside this range in the pre-processed input is converted to 0; any value inside this range is converted to 1. When Operator is set to *PASSRANGE*, any value below Low Threshold is converted to the value of the Low Threshold. Any value above High Threshold is converted to the High Threshold. Values in between the two thresholds are unchanged. Both thresholds should be an integer ranging from 0 to 255.

Operator Set to MASKSHIFT

When Operator is set to MASKSHIFT, MaskShiftSetup is inserted into the Job by the ImageArith Tool. MaskShiftSetup allows you to specify a mask value to bitwise-operate on each pixel. The MaskShiftSetup property page is shown in Figure 13–9.

FIGURE 13–9. MaskShiftSetup Properties Page



- **Mask Operator** — There are two options, OR and AND. Perform bitwise AND or OR operation on the pixels from the two input ROIs.
 - $C = a \& b$ for AND
 - $C = a | b$ for OR
 - $C = a \wedge b$ for XOR

Both input images are unsigned. Clip Input Overflow is provided. Arithmetic-operated image is as unsigned.

- **Mask** — Denotes the mask value.
- **Shift** — Denotes the number of shifts:

Left Shift — Negative entry in the Shift field.

Right shift — Positive entry in the Shift field.

Range: -7 to 7

The shift operation multiplies (left shift) or divides (right shift) the masked result by 2 the number of shift times.

For example:

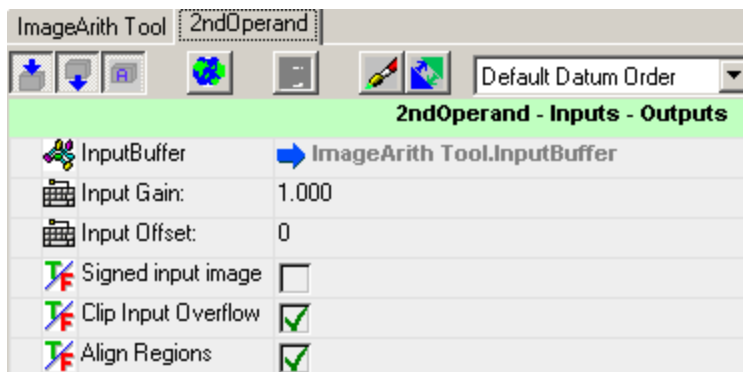
- $C = (a \& \text{mask}) \gg \text{shift}$ for mask AND and right shift

- $C = (a \mid \text{mask}) \gg \text{shift}$ for mask OR and right shift
- $C = (a \& \text{mask}) \ll \text{shift}$ for mask AND and left shift
- $C = (a \mid \text{mask}) \ll \text{shift}$ for mask OR and left shift

Operator Set to Any Two-Operand Operation

When Operator is set to be any of the two-operand operations, the 2ndOperand properties page is displayed to set up the 2ndOperand, as shown in Figure 13–10.

FIGURE 13–10. 2ndOperand Properties Page



- **Input Gain** — Apply gain to the second input image.

Range: -255.0 to 255.0

Signed Input Image and Clip Input Overflow — When the input image is unsigned, you can clip overflows when they occur. When the input image is signed, this parameter name changes to **Clip Input Overflow**.

Clip Input Overflow default: Enabled

- **Align Regions** — When enabled, the two operands' ROIs are aligned in their common part coordinates. When one of the two ROIs is relocated, the other ROI is aligned with it. Both ROIs are always maintained at the same size (the size of the ImageArith Tool) by the tool.

Training

None.

Results

- *Status* — Set to true after a successful execution of the step.
- *ImgArith Output Buffer* — Contains the resultant image. This buffer is accessible for display or as input to other steps. A transformation matrix is available through the input and output buffers' OwnerPart() to transform the coordinate in the output buffer to that in the input buffer. In case ImageArith is set for a 2-operand operation, the first-operand input buffer is assumed to be the input buffer for coordinate transformation.

I/O Summary

ImageArith Tool provides a I/O summary in the Status Bar located at the bottom of the FrontRunner™ window.

Inputs:

- Arith: xxx In(Gain, Offset)-(ggg, ooo) Out(Gain, Offset)-(ggg, ooo) or
- Arith: yyy Thresh-(tttl, ttth) In(Gain, Offset)-(ggg, ooo) Out(Gain, Offset)-(ggg, ooo) or
- Arith: MASKSHIFT: MaskShift-(mmm, sss) In(Gain, Offset)-(ggg, ooo) Out(Gain, Offset)-(ggg, ooo)

Where:

xxx =	is one of the following operations: NOT, NEGATE, COPY, ABS, ADD, SUB, DIFF, AND, OR, XOR, MIN, or MAX
ggg =	gain
ooo =	offset value
yyy =	BINARIZE or PASSRANGE
tttl =	low threshold value
ttth =	high threshold value
mmm =	Mask value
sss =	Shift value

Outputs:

- Operation succeeded, or
- ***Null input buffer, or
- ***Out of memory, or
- ***Fail to operate

MeanLP Filter

This step applies a weighted neighborhood average operation to the pixels of the input image. You can specify the number of operations to be applied to the image. MeanLP Filter softens the contrasts in the image and attenuates the sharpness at edges.

Other Steps Used

None.

Theory of Operation

The weighted neighborhood average operation for each pixel uses its nine pixel neighborhood labeled below:

```

a  b  c
d  e  f
g  h  i

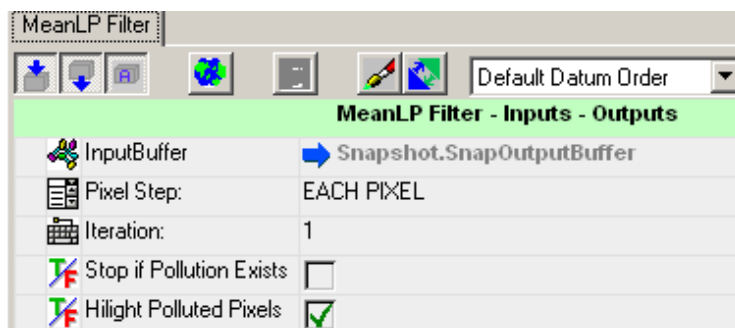
```

Then, the weighted average = $((a + c + g + i) + 2*(b + d + f + h) + 4*e)/16$

Description

MeanLP Filter allows editing through the MeanLP Filter properties page, as shown in Figure 13–11.

FIGURE 13–11. MeanLP Filter Properties Page



Settings

- **Pixel Step** — The averaging process can be applied to EACH PIXEL or EVERY OTHER PIXEL of the input image.
 - **EACH PIXEL** — The resultant image is of the same size as that of the input ROI. This is the default.
 - **EVERY OTHER PIXEL** — The width and height of the resultant image are half of those of the input ROI, respectively.
- **Iteration** — The number of averaging operations to be applied to the input image.

Default: 1

- **Stop if Pollution Exists** — When an ROI is placed close to the boundary of the image display, the number of iterations may cause some boundary pixels of the output image to become polluted. These polluted pixels are the result of the averages involving some padded (unknown) pixels. When this setting is true, MeanLP Filter will stop the execution and return an error. Otherwise, when the execution completes, the polluted pixels may be highlighted depending on the setting of **Highlight Polluted Pixels**.

Default: Disabled

- **Highlight Polluted Pixels** — When enabled, and **Stop if Pollution Exists** is disabled, the polluted pixels will be painted red in the output buffer.

Default: Enabled

Training

None.

Results

An output image whose pixel values are a weighted neighborhood average of the input image pixels. The size of the image depends on the setting of the **Pixel Step** and the number of Iterations:

- **EVERY PIXEL** — The width and height of the output image are the same as those of the ROI regardless of what number of Iterations has been applied to the filtering process.
- **EVERY OTHER PIXEL** — The width and height of the output image are those of the ROI divided by 2 the number-of-iterations times.

I/O Summary

None.

Sobel Filter

This step provides nine neighborhood-based image processing operations. The neighborhood size can be 3x3, 5x5, or 7x7.

Other Steps Used

None.

Theory of Operation

This filter tool processes an input region by replacing each pixel with a function of its neighbors. The typical neighborhood is a 9-pixel neighborhood labeled:

a b c

d e f

g h i

The gradients along the x- and y- coordinates:

- $\text{GradientX} = ((c + 2f + i) - (a + 2d + g)) / \text{variable}$

Where variable can be 1, 2, 4, or 8.

- $\text{GradientY} = ((g + 2h + i) - (a + 2b + c)) / \text{variable}$

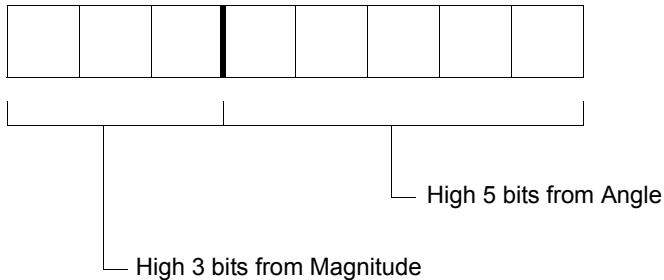
Where variable can be 1, 2, 4, or 8.

The various combinations of the magnitude and angle of the gradients:

- $\text{Magnitude} = \sqrt{\text{sqr}(\text{GradX}) + \text{sqr}(\text{GradY})}$
- $\text{Angle} = \arctan(\text{GradY}/\text{GradX})$ scaled from 0 to 360 into 0 to 255
- **Qualified Angle** — Is the Angle divided by 2 (right-shifted by 1). If, at a pixel point, the Magnitude is greater than a specified threshold, noted as Magnitude Qualification, then the highest bit of the Qualified Angle is set to 1. Therefore, a qualified angle at a qualified point is a value between 128 and 255. For non-qualified points, the qualified angle value is below 128.
- **Absolute Gradient-X** — Absolute value of the GradientX.
- **Absolute Gradient-Y** — Absolute value of the GradientY.

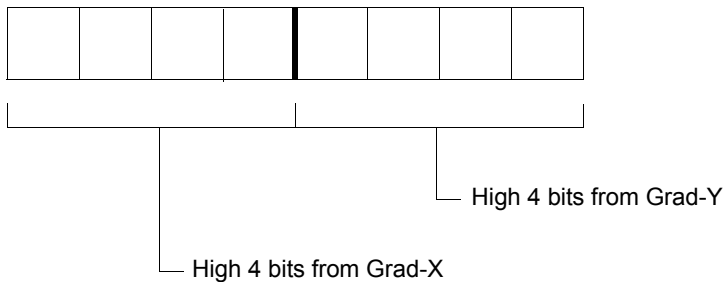
- **3-Bit Mag & 5-bit Ang** — Is a combination of 3-bit magnitude and 5-bit angle. The placement of these bits are shown in Figure 13–12.

FIGURE 13–12. Magnitude and Angle Bit Placement



- **4-Bit GradX & 4-Bit GradY** — Is a combination of 4-bit Grad-X and 4-bit Grad-Y. The placement of these bits are shown in Figure 13–13.

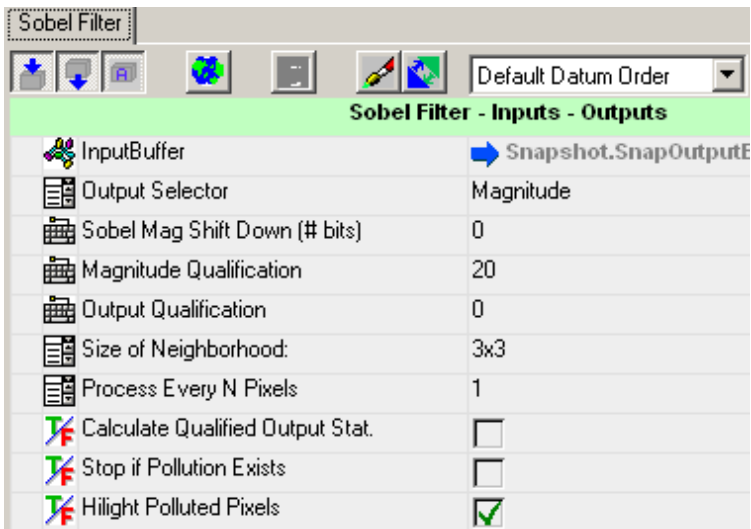
FIGURE 13–13. 4 Bit Grad-X and 4 Bit Grad-Y



Description

Sobel Filter allows editing through the Sobel Filter properties page, as shown in Figure 13–14.

FIGURE 13–14. Sobel Filter Properties Page



Settings

- **Output Selector** — Selects one from this list to be the pixel value of the output image:
 - Dx — Gradient along the x-axis.
 - Dy — Gradient along the y-axis.
 - Magnitude (Default) — Magnitude of the gradient.
 - Angle — Angle of the gradient.
 - Qualified Angle — Is the Angle divided by 2 (or right-shifted by 1). For pixel points whose Magnitude exceeds the **Magnitude Qualification** (shown in the datum view), the highest bit of the Qualified Angle is set to 1, resulting in a value between 128 and 255. In other words, for the qualified points, the Qualified Angle values are between 128 and 255; while for the non-qualified points, the Qualified Angle values are half of the original Angle value and, therefore, are between 0 and 127.
 - Absolute Gradient-x — Absolute value of the gradient along the x-axis.
 - Absolute Gradient-y — Absolute value of the gradient along the y-axis.

- 3-Bit Mag & 5-Bit Ang — Is a combination of 3-bit magnitude and 5-bit angle. The placement of these bits are shown in Figure 13–12 and Figure 13–13.
 - 4-Bit GradX & 4-Bit GradY — Is a combination of 4-bit Grad-X and 4-bit Grad-Y. The placement of these bits are shown in and Figure 13–13.
 - *Sum of Absolute Gradients* — Absolute value of the gradient along the x-axis plus the absolute value of the gradient along the y-axis.
- **Sobel Mag Shift Down (# bits)** — The amount of shift operation on the gradients Dx and Dy. Right-shift is a type of high-pass filter to the gradients. This is applied to the gradient along the x-axis and y-axis individually:
 - None
 - Shift 1 bit down
 - Shift 2 bits down
 - Shift 3 bits down
- **Magnitude Qualification** — Used as a threshold to qualify a point for the calculation of the Qualified Angle. It also calculates the following statistics:
 - **Sum of Qualified Gray Values** — Sum of the gray values of those pixels whose gradient magnitude exceeds the **Magnitude Qualification**.
 - **Cnt of Qualified Gray Values** — Count of those pixels whose gradient magnitude exceeds the **Magnitude Qualification**.
- **Output Qualification** — Threshold used for calculating the following output statistics:
 - **Sum of Qualified Output** — Sum of the sobel outputs values that exceed this **Output Qualification**.
 - **Cnt of Qualified Output** — Count of those pixel points where the sobel output values exceed this **Output Qualification**.

Note: These statistics are calculated only when **Calculate Qualified Output Stat** is enabled (checked).

- **Size of Neighborhood** — Provides the following selections:
 - 3x3 (default)
 - 5x5
 - 7x7
- **Calculate Qualified Output Stat** — When enabled (checked), **Output Qualification** is a threshold to calculate the statistics Sum of Gray Values and Cnt of Gray Values in the Output Image.
- **Stop if Pollution Exists** — If the neighborhood operation would involve padded (unknown) pixels, the sobel processing will stop if this setting is enabled.
- **Highlite Polluted Pixels** — If any pollution exists along the border of the ROI, the polluted pixels, where the sobel outputs are calculated with padded (unknown) pixels, will be highlighted red when enabled.

Training

None.

Results

An output image whose size is the same as the input ROI and whose pixel values (or sobel outputs) correspond to **Output Selector**.

- **Status** — Set to true after a successful execution of the step.
- **Sum of Qualified Gray Values** — Sum of the gray values of those pixels in the input image whose sobel gradient magnitude exceeds the **Magnitude Qualification**.
- **Cnt of Qualified Gray Values** — Count of those pixels in the input image whose gradient magnitude exceeds the **Magnitude Qualification**.
- **Sum of Gray Values in the Output Image** — Sum of all sobel output values.
- **Cnt of Gray Values in the Output Image** — Count of pixel points where the sobel output value is non-zero.
- **Sum of Qualified Output** — Sum of the sobel outputs that exceed this **Output Qualification**. This is calculated only when **Calculate Qualified Output Stat** is enabled.

- **Cnt of Qualified Output** — Count of those pixel points where the sobel output values exceed this Output Qualification. This is calculated only when **Calculate Qualified Output Stat** is enabled.

I/O Summary

None.

Binarized Color Thresholding Filter Tool

This tool allows the user to set thresholds for the RGB or HSI image channels. The user selects to which color representation the thresholding will be applied (RGB or HSI). The user then selects the lower and upper pixel range (Range 0-255) for each image channel. These ranges determine the binarized image, i.e. outside the range: 0 inside the range: 255 (for an 8-bit image). The three planes are then AND'd together to produce the output image from the tool. There is also an option to invert the output result for use in masking operations.

Note that the HSI plane selection available is further refined to allow the user to select a hue value of zero to represent either pure red, pure green or pure blue. This is necessary in order to avoid wrap around issues. For instance, when looking at reddish hues with a hue value of zero representing pure red, a color slightly towards green has a low hue value whereas a color slightly towards blue has a value close to 255. Choosing to use hue 0 = green or blue for this case would eliminate this problem for the two reds above.

FIGURE 13–15. Hue Wheel

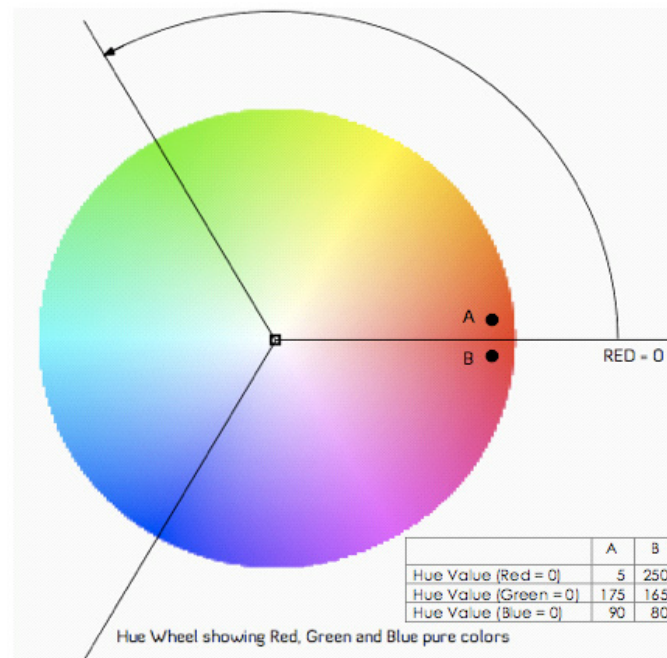


Figure 13-16 shows the Color Thresholding Filter Step being inserted into the inspection. Note that this step should only be available when the snapshot buffer is a color image.

FIGURE 13–16. Insertion of Color Thresholding Filter

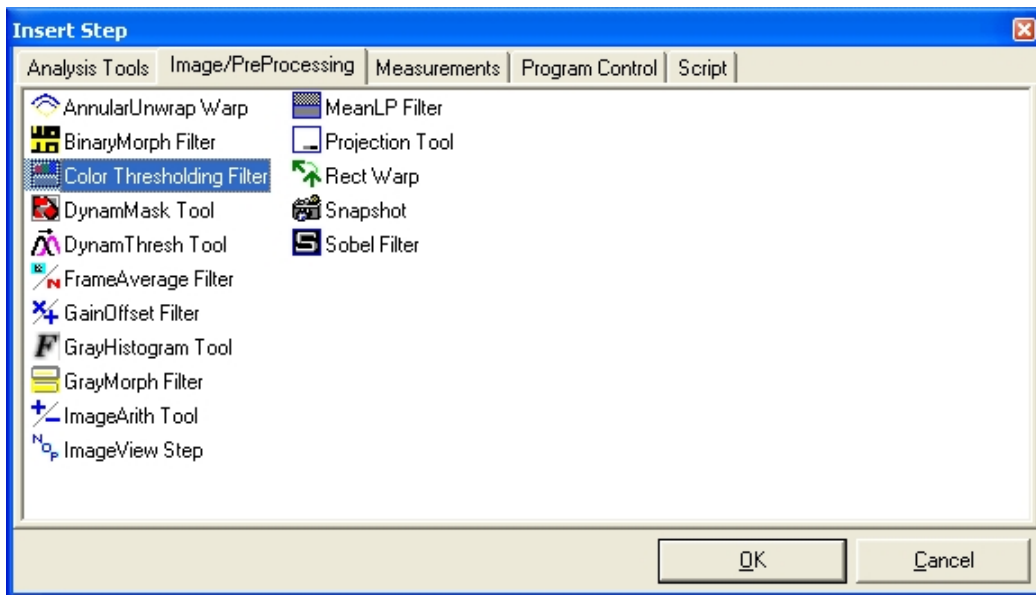


Figure 13-17 displays the options available for Color Thresholding in RGB space.

FIGURE 13–17. Options for Color Thresholding Filter

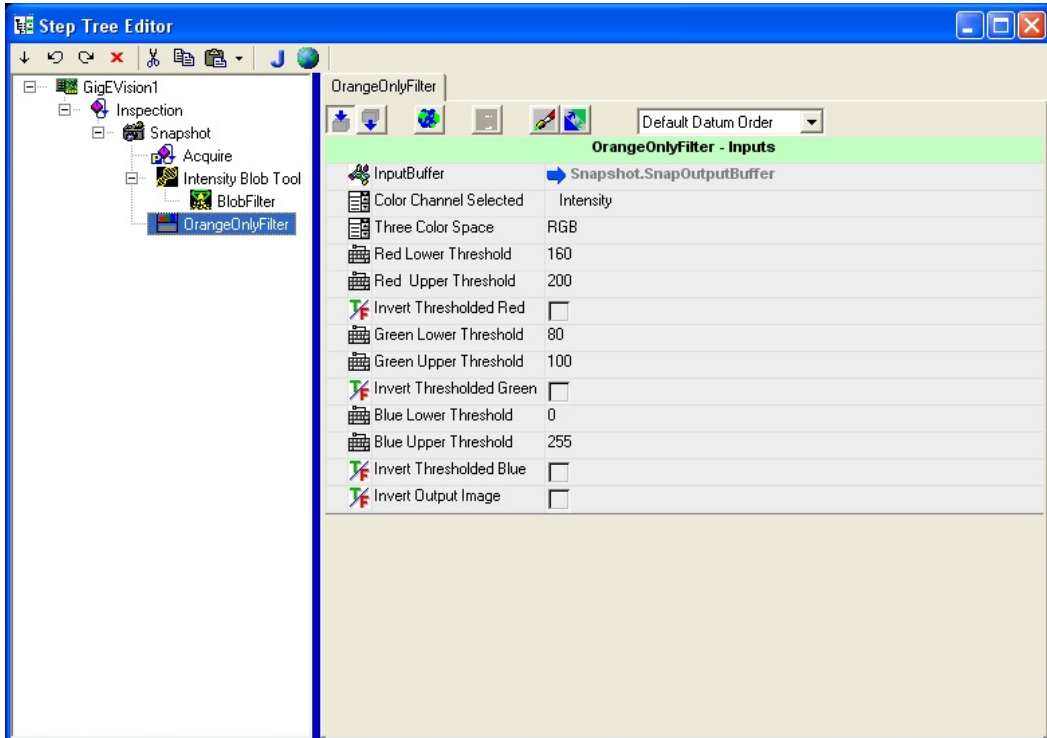


Figure 13-18 displays the selection of the different color space models available. They are RGB, HSI (Hue Red =0), HSI (Hue Green =0), HSI (Hue Blue =0).

FIGURE 13–18. Options for RGB Color Space Thresholding

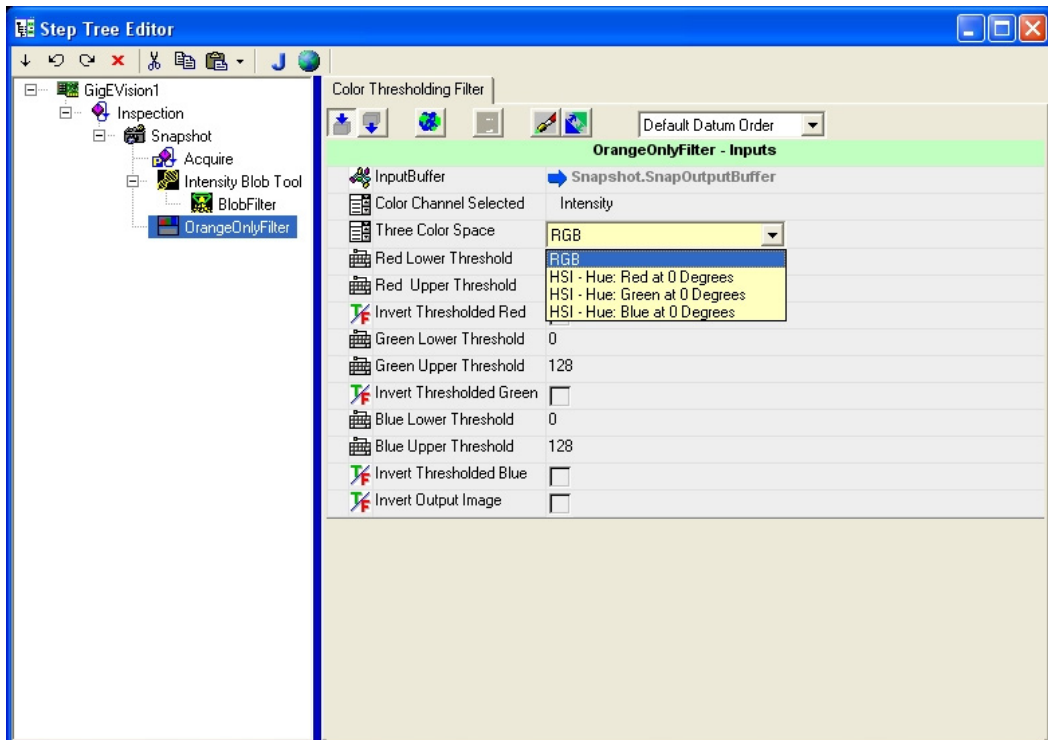


Figure 13-19 displays the parameters available when the color space HSI (Hue Green =0) is selected.

FIGURE 13–19. Options for HUE Color Space Thresholding

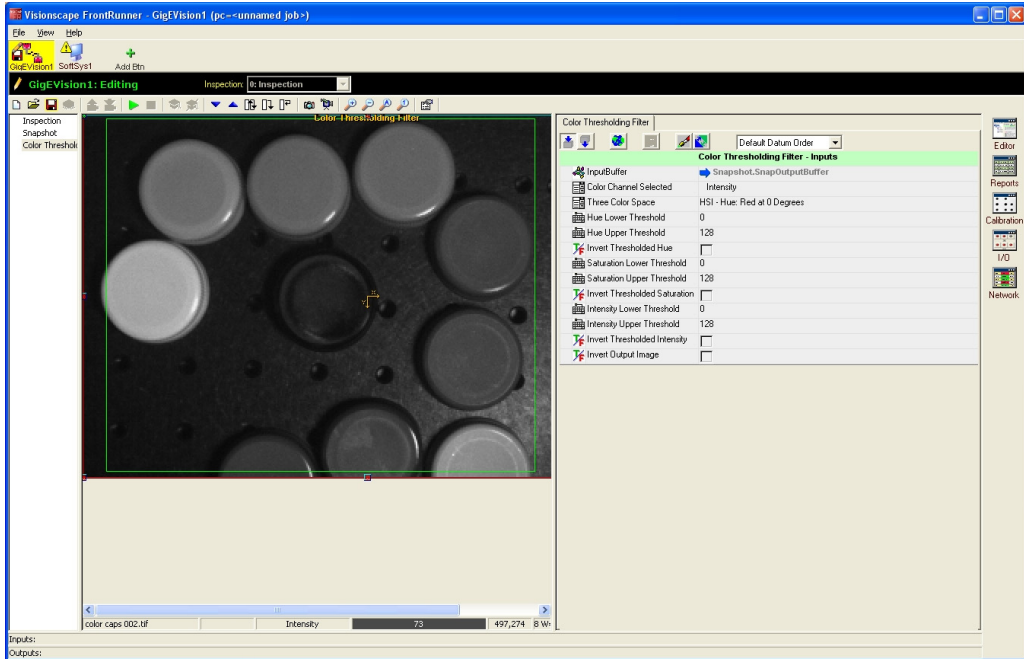
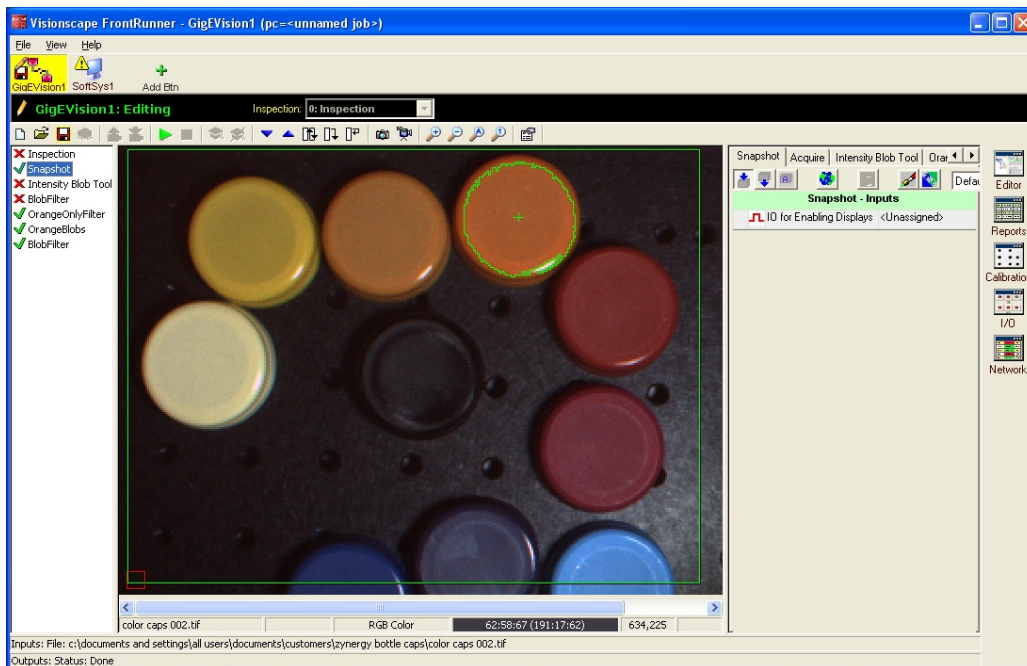


Figure 13-20 displays the output of the Color Thresholding Filter Color Space RGB threshold range 0-128 for all planes.

FIGURE 13–20. Results of Color Thresholding RGB Space 0-128



Zooming and Binning

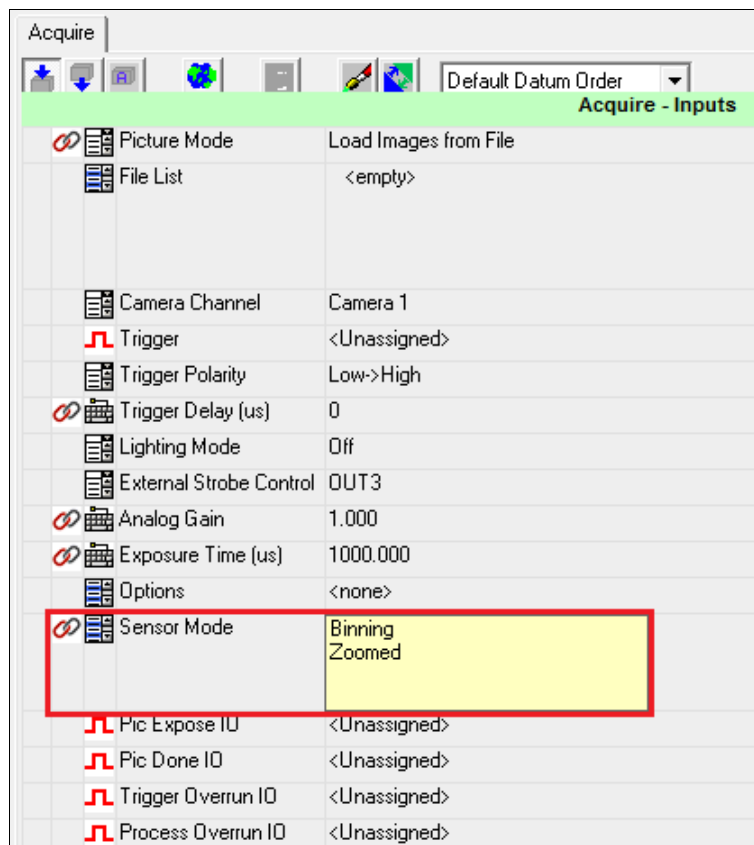
Important: Zooming and Binning functionality is available for SXGA cameras only.

The **Zoomed** and **Binning** options allow you to change how images are acquired in order to increase capture rate (Zoomed) or improve the signal-to-noise ratio (Binning).

Zoomed provides a quarter-resolution centered partial-scan image. This image has a smaller field of view than the default full-scan image.

Binning provides a quarter-resolution full-scan image with the same field of view and capture rate as a full-scan image. The reduction in total pixels reduces image resolution.

FIGURE 13–21. Sensor Mode



Error Codes

This appendix contains error codes for Visionscape tools.

Abstract Vision Step Error Codes

TABLE A-1. Abstract Vision Step Error Code

General Error	Code	Code
ABSVISION_ER_CONSTRUCT	139001	0x21ef9

Arith Step Error Codes

TABLE A-2. Arith Step Error Codes

General Error	Code	Code	Description
AR_ER_INPUT_BUFFER	125001	0x1e849	Invalid or no input buffer
AR_ER_OUTOFMEM	125002	0x1e84a	Error allocation internal resource
AR_ER_ROIOFF	125003	0x1e84b	Second operand ROI off buffer

Auto/Dynamic Mask Step Error Codes

TABLE A-3. Auto/Dynamic Mask Step Error Codes

General Error	Code	Code	Description
AUTOMASK_ER_THRESHOLDING	142001	0x22ab1	Private step that computes threshold failed when run.
AUTOMASK_ER_COMPPOLARITY	142002	0x22ab2	Private step that computes polarity failed when run.

Binary Morphology Agent Error Codes

TABLE A-4. Binary Morphology Agent Error Codes

General Error	Code	Code	Description
BADBMOPCODE	3001	0xbb9	Invalid morph operation specified.
BINMORPH_TIMEOUT	3002	0xbba	

Binary Morphology Step Error Codes

TABLE A-5. Binary Morphology Step Error Codes

General Error	Code	Code	Description
BINMORPH_ER_BINARIZE	116001	0x1c521	Binarization failed

Blob Agent Error Codes

TABLE A-6. Blob Agent Error Codes

General Error	Code	Code
BLOB_ER_HWOVERFLOW	1001	0x3e9
BLOB_ER_HWSEGOVERFLOW	1002	0x3ea
BLOB_ER_IMGTOOCOMPLEX	1003	0x3eb

Blob Datum Error Codes

TABLE A-7. Blob Datum Error Codes

General Error	Code	Code
IMGANALYSIS_E_BADBLOBVTARG	2049	0x801
IMGANALYSIS_E_BADBLOBNUM	2050	0x802
IMGANALYSIS_E_BADBLOBFEATURE	2051	0x803

Blob Step Error Codes

TABLE A-8. Blob Step Error Codes

General Error	Code	Code
BLOB_ER_NULLAGENT	134001	0x20671
BLOB_ER_AGENTFAILURE	134002	0x20672
BLOB_ER_TOTAREA_OUTRANGE	134003	0x20673
BLOB_ER_NUMBLOBS_OUTRANGE	134004	0x20674

Buffer Datum Error Codes

TABLE A-9. Buffer Datum Error Codes

General Error	Code	Code	Description
BUFFERDM_ER_REFRESHSCRN	140001	0x222e1	Error occurred mapping pixels for transfer to vga

Buffer Manager Error Codes

TABLE A-10. Buffer Manager Error Codes

General Error	Code	Code
BUFMGR_E_NOBUFDM	1281	0x501
BUFMGR_E_NOOFFSCREENDC	1282	0x502
BUFMGR_E_NOBUFVIEW	1283	0x503
BUFMGR_E_NOBUFFER	1284	0x504

C2D Error Codes (Math Library 2D Calibration)

TABLE A-11. C2D Error Codes (Math Library 2D Calibration)

General Error	Code	Code
C2D_ER_DIMS	201001	0x31129
C2D_ER_MEMORY	201002	0x3112a
C2D_ER_TRANSPOSE	201003	0x3112b
C2D_ER_CENTER	201005	0x3112d
C2D_ER_SOLVEA	201006	0x3112e
C2D_ER_SOLVEB	201007	0x3112f
C2D_ER_NULL	201008	0x31130
C2D_ER_CANTFIND	201009	0x31131

These error values may be combined with an agent code to indicate the agent that was executing when the general error occurred. For example, an error code of 16505 would indicate that CPU memory was exhausted (16505) when the OCV agent was executing (16505). Five hundred (16505) may be added to all such error codes to ensure that they are unique for each agent.

Table A–12 lists the current agent codes.

TABLE A–12. Current Agent Codes

0	WARP_AGENT
1	BLOB_AGENT
2	ARITH_AGENT
3	BINMORPH_AGENT
4	GRAYMORPH_AGENT
5	SOBEL_AGENT
6	GRADSCAN_AGENT
7	PROJECT_AGENT
8	THRESH_AGENT
9	HIST_AGENT
10	DECIMATE_AGENT
11	CONV_AGENT
12	CORRSEARCH_AGENT
13	CORRMATCH_AGENT
14	PROBE_AGENT
15	HOUGH_AGENT
16	OCV_AGENT
17	BALLMEASURE_AGENT
18	MASK_AGENT
19	TRANSPOSE_AGENT
20	CORRSRCH_AGENT
22	BREAK_AGENT
23	AUTOFOCUS_AGENT
25	SEGMENT_AGENT
26	FEATEXTRACT_AGENT
27	EDGEFAST_AGENT
28	EDGEPRIM_AGENT
29	VECTORFAST_AGENT

Correlation Search Agent Error Codes

TABLE A-13. Correlation Search Agent Error Code

General Error	Code	Code	Description
CORRSRCH_ER_PREGOUTILFAIL	20001	0x4e21	An allocation error occurred during setup of corr search agent

Custom Step Error Codes

TABLE A-14. Custom Step Error Codes

General Error	Code	Code	Description
CUSTOM_ER_SCRIPT_EMPTY	152001	0X251C1	"None" script is selected.
CUSTOM_ER_SCRIPT_EMPTY_OR_ERROR	152002	0X251C2	"None" script or script not found.

DataMatrix/BarCode Step Error Codes

TABLE A-15. Data Matrix/BarCode Error Messages

General Error	Code
IP_NO_EDGE_CANDIDATE_FOUND	127001
IP_FIRST_EDGE_NOT_FOUND_OR_TOO_SMALL	127004
IP_SECOND_EDGE_NOT_FOUND	127005
IP_THIRD_EDGE_NOT_FOUND	127011
IP_FOURTH_EDGE_NOT_FOUND	127012
IP_FOUR_CORNERS_NOT_FOUND	127020
IP_SIZE_TEST_FAILED	127021
IP_ROW_COL_TEST_FAILED	127022
IP_INSPECTION_TIMEOUT	127030
IP_BORDER_MATCH_TEST_FAILED	127033
IP_ECC_UNDECODABLE	127048
IP_CONTRAST_CALIBRATION_FAILURE	127050
IP_CELLUNIT_CALIBRATION_FAILURE	127051
IP_VERIFICATION_PROCESS_ERROR	127100
IP_VERIFICATION_UNSUPPORTED	127101
IP_VERIFICATION_TIMEOUT	127102
IP_ISO_V_ECC200_REQUIRED	127110
IP_ISO_V_APERTURE_TOO_SMALL	127111
IP_ISO_V_APERTURE_TOO_LARGE	127112
IP_ISO_V_INSUFFICIENT_SPACE	127113
IP_ISO_V_FAIL_RDA_STEP_F_1	127114
IP_ISO_V_FAIL_RDA_STEP_F_2	127115
IP_ISO_V_FAIL_RDA_STEP_F_3	127116
IP_ISO_V_FAIL_RDA_STEP_A_E	127117
QR_CODE_DESIGN_UNIMPLEMENTED	128400
QR_CODE_IP_GENERAL_ERROR	128401
QR_CODE_IP_RATIO_ERROR	128402
QR_CODE_IP_FINDER_ERROR	128403
QR_CODE_IP_LINE_FIT_ERROR	128404

TABLE A–15. Data Matrix/BarCode Error Messages (continued)

General Error	Code
QRCODE_IP_LINE_INTERSECT_ERROR	128405
QRCODE_IP_CORNER_ERROR	128406
QRCODE_DEC_UNKNOWN_ERROR	128420
QRCODE_RS_LEVEL_INVALID	128421
QRCODE_FORMAT_INFO_FAILED	128422
QRCODE_VERSION_INFO_FAILED	128423
QRCODE_ROWS_COLS_INVALID	128424
QRCODE_DATA_CODEWORD_INVALID	128425
QRCODE_TOTAL_CODEWORD_INVALID	128426
QRCODE_MODE_INDICATOR_INVALID	128427
QRCODE_MODE_UNIMPLEMENT	128428
QRCODE_RS_DECODE_FAILED	128429
QRCODE_BCH15_5_UNDECODABLE	128430
QRCODE_MODEL_INVALID	128431

Edge Step Error Codes

TABLE A-16. Edge Step Error Codes

General Error	Code	Code	Description
EDGE_ER_WARPRUN	101001	0x18a89	Error running embedded warp agent
EDGE_ER_PROJECTAGNT	101002	0x18a8a	Error running embedded projection agent
EDGE_ER_GRADSCANAGNT	101003	0x18a8b	Error running embedded gradscan agent

File I/O Error Codes

TABLE A-17. File I/O Error Codes

General Error	Code	Code
FIO_ER_MALLOC	3000	0xbb8
FIO_ER_ALRDYOPEN	3001	0xbb9
FIO_ER_INVDOFFSET	3002	0xbba
FIO_ER_WRITEHEADER	3100	0xc1c
FIO_ER_READTAGS	3101	0xc1d
FIO_ER_TAGNOTFOUND	3102	0xc1e
FIO_ER_BADTYPE	3103	0xc1f
FIO_ER_READFILE	3104	0xc20
FIO_ER_READHEADER	3105	0xc21
FIO_ER_TIFF	3106	0xc22
FIO_ER_SAMPLE	3107	0xc23
FIO_ER_COMPRESSION	3108	0xc24
FIO_ER_GRAYRESPCURVE	3109	0xc25
FIO_ER_PHOTOINTERP	3110	0xc26
FIO_ER_PUTTAGS	3111	0xc27
FIO_ER_PREDICTOR	3112	0xc28
FIO_ER_OUTBOUNDS	3113	0xc29
FIO_ER_PIXELSIZE	3114	0xc2a

TABLE A-17. File I/O Error Codes (continued)

General Error	Code	Code
FIO_ER_LZW	3115	0xc2b
FIO_ER_STRIPOFFSET	3116	0xc2c
FIO_ER_STRIPCOUNT	3117	0xc2d
FIO_ER_BADFILE	3118	0xc2e
FIO_ER_TFLUSHVOL	3200	0xc80
FIO_ER_TOPEN	3201	0xc81
FIO_ER_TREAD	3202	0xc82
FIO_ER_TWRITE	3203	0xc83
FIO_ER_TSETEOF	3204	0xc84
FIO_ER_TCLOSE	3205	0xc85
FIO_ER_TCREATE	3206	0xc86
FIO_ER_TDELETE	3207	0xc87
FIO_ER_TMEMORY	3208	0xc88
FIO_ER_TCLIP	3209	0xc89
FIO_ER_TFILEOPEN	3210	0xc8a
FIO_ER_TPUTTAGS	3211	0xc8b
FIO_ER_TWRITEIMAGE	3212	0xc8c
FIO_ER_TREADTAGS	3213	0xc8d
FIO_ER_TWRITETAGS	3214	0xc8e
FIO_ER_TREADHEADER	3215	0xc8f
FIO_ER_TWRITEHEADER	3216	0xc90
FIO_ER_TPHOTOINTERP	3217	0xc91
FIO_ER_TTIFF	3218	0xc92
FIO_ER_TIMAGECROPWARN	3219	0xc93
FIO_ER_TSAMPLE	3220	0xc94
FIO_ER_TCOMPRESSION	3221	0xc95
FIO_ER_TMINSAMPLE	3222	0xc96
FIO_ER_TMAXSAMPLE	3223	0xc97
FIO_ER_TGRAYRESPCURVE	3224	0xc98
FIO_ER_TREADIMAGE	3225	0xc99
FIO_ER_TIFFINFO	3226	0xc9a

TABLE A-17. File I/O Error Codes (continued)

General Error	Code	Code
FIO_ER_TBITSERPPIXEL	3227	0xc9b
FIO_ER_TIFFOUTSIDE	3229	0xc9d
FIO_ER_TIFFCLIPPED	3230	0xc9e
FIO_ER_TSKIPTOEND	3231	0xc9f
FIO_ER_TNULLPTR	3232	0xca0
FIO_ER_TBUFINFO	3233	0xca1
FIO_ER_TBUFOUTSIDE	3234	0xca2
FIO_ER_TIFFID	3235	0xca3
FIO_ER_TOPENINFO	3236	0xca4
FIO_ER_TSAVEINFO	3237	0xca5
FIO_ER_TCOLORMAP	3238	0xca6
FIO_ER_TWRONGRECT	3239	0xca7
FIO_ER_VOLUME	3240	0xca8
FIO_ER_INVPLANE	3241	0xca9

GainOffset Step Error Codes

TABLE A-18. GainOffset Step Error Codes

General Error	Code	Code	Description
GAINOFFSET_ER_ARITHAGENT	128001	0x1f401	Error occurred running internal arith agent

General Error Codes

TABLE A-19. General Error Codes

General Error	Code	Code	Description
SYS_ER_NOMEM_SWBUF	1	0x1	Failure allocating buffer in non-asic memory
SYS_ER_NOMEM_ASIC_BANKA	2	0x2	Failure allocating buffer in asic memory bank A
SYS_ER_NOMEM_ASIC_BANKB	3	0x3	Failure allocating buffer in asic memory bank B
SYS_ER_NOMEM_BUFFER_POOL	4	0x4	Failure allocating buffer from the memory pool
SYS_ER_NOMEM_CPU	5	0x5	insufficient memory for general allocation
SYS_ER_MONST_PROC	6	0x6	Error running last procedure on ASIC

Gradient Scan Agent Error Codes

TABLE A-20. Gradient Scan Agent Error Codes

General Error	Code	Code	Description
GRADSCAN_ER_BADROW	6001	0x1771	Row number out of range in gradient data
GRADSCAN_ER_NOPOINTS	6002	0x1772	No gradient points in gradscan output data
GRADSCAN_ER_NOMEMORY	6003	0x1773	Unable to allocate internal resources (NOT USED)

Gray Morphology Agent Error Codes

TABLE A-21. Gray Morphology Agent Error Codes

General Error	Code	Code	Description
BADGMOPCODE	4001	0xfa1	Invalid morphology opcode specified
GRAYMORPH_TIMEOUT	4002	0xfa2	Timeout occurred when running on asic
NOSCRATCHBUFFER	4003	0xfa3	A scratch buffer could not be allocated for internal use
MORPH_ER_BADKERN	4004	0xfa4	Invalid kernel specified
MORPH_ER_MALLOC	4005	0xfa5	Unable to allocate internal resources

Hough Agent Error Codes

TABLE A-22. Hough Agent Error Codes

General Error	Code	Code
HOUGH_ER_INTERSECT	15001	0x3a99
HOUGH_ER_WRONGTWO	15002	0x3a9a
HOUGH_ER_TOOMUCH	15003	0x3a9b
HOUGH_ER_NULL	15004	0x3a9c
HOUGH_ER_SCANSLOT	15005	0x3a9d
HOUGH_ER_DIVZERO	15006	0x3a9e
HOUGH_ER_NOTINT	15007	0x3a9f
HOUGH_ER_MEMORY	15008	0x3aa0
HOUGH_ER_RESULT	15009	0x3aa1
HOUGH_ER_DUNNO	15010	0x3aa2
HOUGH_ER_TEMPL	15011	0x3aa3
HOUGH_ER_SCHEDNO	15012	0x3aa4
HOUGH_ER_TOOMANY	15013	0x3aa5
HOUGH_ER_NOTEMPS	15014	0x3aa6
HOUGH_ER_NORES	15015	0x3aa7
HOUGH_ER_RESCORRUPT	15016	0x3aa8

TABLE A-22. Hough Agent Error Codes (continued)

General Error	Code	Code
HOUGH_ER_TEMPHANDLE	15017	0x3aa9
HOUGH_ER_ROIHANDLE	15018	0x3aaa
HOUGH_ER_ACCUOVERFLOW	15019	0x3aab
HOUGH_ER_NOTENOUGHSLLOT	15020	0x3aac

Hough Agent Error Codes (Data Table)

TABLE A-23. Hough Agent Error Codes (Data Table)

General Error	Code	Code
TBL_ER_INITED	15800	0x3db8
TBL_ER_MALLOC	15801	0x3db9
TBL_ER_BADTBLNO	15802	0x3dba
TBL_ER_NOTINUSE	15803	0x3dbb
TBL_ER_BADENTRY	15804	0x3dbc
TBL_ER_ALRDYFREED	15805	0x3dbd
TBL_ER_ALRDYLOCKED	15806	0x3dbe
TBL_ER_READLOCK	15807	0x3dbf
TBL_ER_ALRDYUNLOCKED	15808	0x3dc0
TBL_ER_RANGE	15809	0x3dc1
TBL_ER_ALLUSED	15810	0x3dc2

Hough Agent Error Codes (Line Fit)

TABLE A-24. Hough Agent Error Codes (Line Fit)

General Error	Code	Code
L2D_ER_MALLOC	15601	0x3cf1
L2D_ER_LINE	15602	0x3cf2
L2D_ER_SAMEPTS	15603	0x3cf3
L2D_ER_NORMALIZE	15604	0x3cf4
L2D_ER_DTYPE	15605	0x3cf5
L2D_ER_POINTS	15606	0x3cf6
L2D_ER_PTDIMS	15607	0x3cf7
L2D_ER_PARALLEL	15608	0x3cf8
L2D_ER_RECT	15609	0x3cf9
L2D_ER_RECTDIMS	15610	0x3cfa
L2D_ER_LINEDIMS	15611	0x3cfb
L2D_ER_NOTONLINE	15612	0x3cfc
L2D_ER_STRAIGHTLINE	15613	0x3cfd
L2D_ER_UNKNOWNALG	15614	0x3cfe
L2D_ER_NOTENOUGHPTS	15615	0x3cff

Hough Agent Error Codes (Matrix)

TABLE A-25. Hough Agent Error Codes (Matrix)

General Error	Code	Code
MAT_ER_NULL	15700	0x3d54
MAT_ER_DIMS	15701	0x3d55
MAT_ER_TYPE	15702	0x3d56

Hough Agent Error Codes (Scan)

TABLE A-26. Hough Agent Error Codes (Scan)

General Error	Code	Code
SCAN_ER_TEMPL	15501	0x3c8d
SCAN_ER_BUFHANDLE	15502	0x3c8e
SCAN_ER_SLOTNO	15503	0x3c8f
SCAN_ER_NULL	15504	0x3c90
SCAN_ER_DYNAM	15505	0x3c91
SCAN_ER_BOUNDS	15506	0x3c92
SCAN_ER_MEMORY	15507	0x3c93
SCAN_ER_NOTINT	15508	0x3c94
SCAN_ER_CANTSORT	15509	0x3c95
SCAN_ER_NOPOINTS	15510	0x3c96
SCAN_ER_MALLOC	15511	0x3c97
SCAN_ER_TOOMANYPTS	15512	0x3c98
SCAN_ER_SCHEDNO	15513	0x3c99
SCAN_ER_NOTENOUGHSLLOT	15514	0x3c9a

Line Functions Error Codes

TABLE A-27. Line Functions Error Codes

General Error	Code	Code	Description
LINE_ER_PARALLEL	200001	0x30d41	Input lines are parallel when looking for intersection point, bisect point or included angle
LINE_ER_INTERSECT	200002	0x30d42	Less than 2 points found when computing the intersection of a line and a rectangle
LINE_ER_NORMALIZE	200003	0x30d43	Cannot normalize the input line when computing point-to-line distance, or normal to a line through a point
LINE_ER_POINT_COUNT	200004	0x30d44	Not enough points to fit a line
LINE_ER_BAD_FIT	200005	0x30d45	Not enough points used for a line fit (<30%)

Mask Agent Error Codes

TABLE A–28. Mask Agent Error Codes

General Error	Code	Code	Description
MASK_ER_NOMASK	18001	0x4651	(not used)

Mask Datum Error Codes

TABLE A–29. Mask Datum Error Codes

General Error	Code	Code
MASKDM_ER_COUNTONPIX	141001	0x226c9
MASKDM_ER_MERGEMASK	141002	0x226ca
MASKDM_ER_MERGE_DIFFPART	141003	0x226cb
MASKDM_ER_MERGE_SAMEPART	141004	0x226cc
MASKDM_ER_DILATEMASK	141005	0x226cd
MASKDM_ER_THRESHMASK	141006	0x226ce
MASKDM_ER_UPDT1BITMASK	141007	0x226cf
MASKDM_ER_UPDT8BITMASK	141008	0x226d0
MASKDM_ER_CONSTRUCTOR	141009	0x226d1
MASKDM_ER_ALLOC8BIT	141010	0x226d2
MASKDM_ER_ALLOC1BIT	141011	0x226d3
MASKDM_ER_RESETVMASK	141012	0x226d4
MASKDM_ER_MOVEMASK	141013	0x226d5
MASKDM_ER_CREATEMBUF	141014	0x226d6

Measurement Step Error Codes

TABLE A–30. Measurement Step Error Codes

General Error	Code	Code	Description
MEAS_ER_INVALID_INPUT	131001	0x1ffb9	An input datum to a measurement step is invalid

Mean Filter Step Error Code

TABLE A-31. Mean Filter Step Error Codes

General Error	Code	Code
MF_ER_ROI_BOUNDARY	5401	0x1519

Npin Find Step Error Codes

TABLE A-32. Npin Find Step Error Codes

General Error	Code	Code	Description
NPINFIND_ER_NOTRIGID	122001	0x1dc91	Pins geometry not rigid enough
NPINFIND_ER_PINOUTOFPOS	122002	0x1dc92	One or more pin is out of position

Point-Point Distance Step Error Codes

TABLE A-33. Point-Point Distance Step Error Codes

General Error	Code	Code	Description
PT2PT_ER_COMPART_NOTFOUND	132001	0x203a1	The two input points do not share a common coordinate system

Project Step Error Codes

TABLE A-34. Project Step Error Codes

General Error	Code	Code
PROJECT_ER_WARPSTEP	133001	0x20789
PROJECT_ER_FILLOUTBUF	133002	0x2078a
PROGMGR_E_ILLEGALSTEPLOAD	517	0x205
PROGMGR_E_NOTINLOAD	518	0x206
PROGMGR_E_REENTRANCY	519	0x207
PROGMGR_E_ONEJOBONLY	520	0x208

Shape Datum Error Codes

TABLE A-35. Shape Datum Error Codes

General Error	Code	Code	Description
SHAPEDM_ER_BUFBOUNDS	143001	0x22e99	Shape is not on the buffer

Snapshot Error Codes

TABLE A-36. Snapshot Error Codes

General Error	Code	Code	Description
SNAPSHOT_ER_NO_CAM	113001	0x1b969	The camera definition does not match the digitizer type or connected camera.
SNAPSHOT_ER_FIFO_OVERRUN	113002	0x1b96a	The digitizer cannot DMA the image across the bus because the bus bandwidth is being exhausted.
SNAPSHOT_ER_PROC_OVERRUN	113003	0x1b96b	The processing is taking too long for the trigger rate. The driver has run out of buffers for image storage.
SNAPSHOT_ER_TRIG_OVERRUN	113004	0x1b96c	The triggers are occurring faster than the camera can take pictures.
SNAPSHOT_ER_TIME_OUT	113005	0x1b96d	A faulty camera or cable has caused an image capture to not complete in a timely fashion.
SNAPSHOT_ER_NULL_FRAME	113006	0x1b96e	The driver passed an invalid image buffer to Snapshot.
SNAPSHOT_ER_ABORTED	113007	0x1b96f	The last image buffer indicated the inspection is being stopped.
SNAPSHOT_ER_CAM_DISCONNECT	113008	0x1b970	The camera is no longer connected to the digitizer.
SNAPSHOT_ER_CAM_OVERRUN	113009	0x1b971	Two or more snapshots have attempted to capture an image from the same camera at the same time.
SNAPSHOT_ER_UNKNOWN	113010	0x1b972	Some other internal error has occurred.

Sobel Step Error Codes

TABLE A-37. Sobel Step Error Codes

General Error	Code	Code
SOBEL_ER_ROI_BOUNDARY	130001	0x1fbd1
SOBEL_ER_3BY3	130002	0x1fbd2
SOBEL_ER_5BY5	130003	0x1fbd3
SOBEL_ER_7BY7	130004	0x1fbd4

Vector Error Code

TABLE A-38. Vector Error Code

General Error	Code	Code	Description
VECTORTOOL_ER_NOTRANS	153001	0x255a9	No points found along vector

FTP Output Step Error Codes

TABLE A-39. FTP Output Step Error Code

General Error	Code	Code	Description
FTPOUTPUT_ERR_INVALID_INPUT	163001	0x27cb9	Input datums not set
FTPOUTPUT_ERR FTPCONNECT	163002	0x27cba	Failed to connect to server
FTPOUTPUT_ERR FTPPUTFILE	163003	0x27cbb	Failed to upload file

Sample Calibration Targets

The dimensions of the calibration target used in the examples in this manual are shown in the following pages. You may use a calibration target of any dimensions you choose, as long as you enter the target centers of that calibration target; for more information, see “Using Robust Calibration” on page 2-7. The size of the calibration target depends on the camera’s FOV. In general, the calibration target should fill the FOV. Back lighting the calibration target ensures a maximum contrast image and optimum calibration accuracy. The dimensions are shown in inches.

Note: The following targets are examples; do not use them as actual calibration targets. Refer to “Calibration Target Design Criteria” on page 2-19 for a detailed description of Calibration Targets.

FIGURE B-1. Calibration Targets (4.0x4.0 inches)

Calibration Target 4.0x4.0 in.

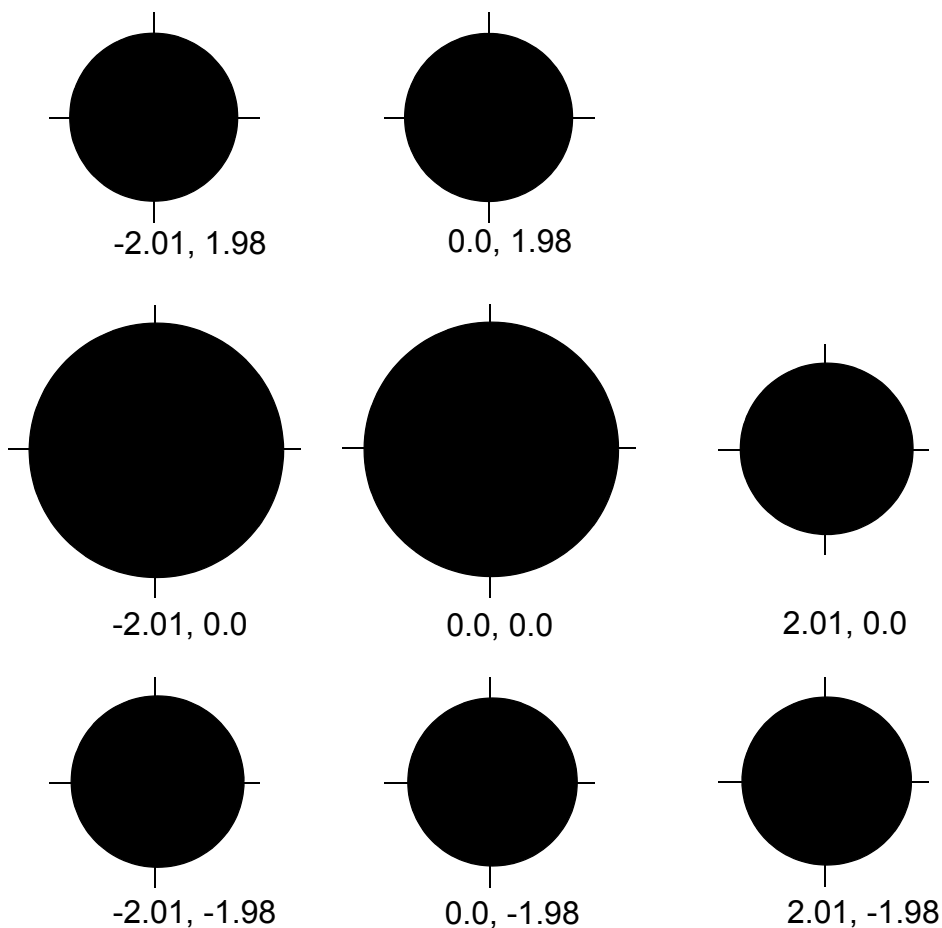
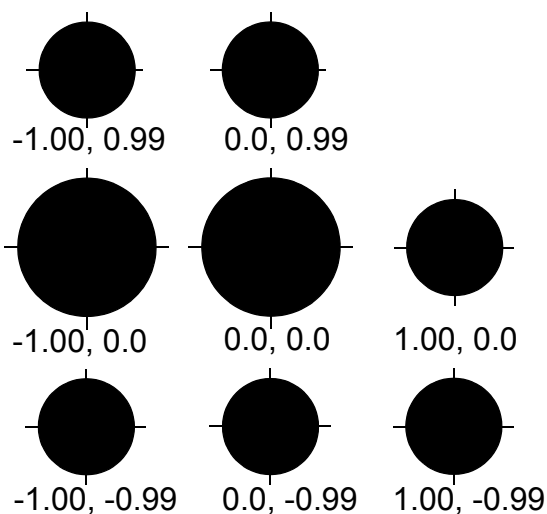


FIGURE B-2. Calibration Targets (2.0x2.0 inches)

Calibration Target 2.0x2.0 in.

**FIGURE B-3. Calibration Targets (1.0x1.0 inches)**

Calibration Target 1.0x1.0 in.

