CVL 9.0
Release Notes

2019 October 02
Revision: 9.0.0.1

Legal Notices

CVL 9.0 Release Notes

CVL Version 9.0 contains new or changed features since the previous release of CVL 8.0.

System Requiments

Your PC must meet minimum requirements to install and use CVL.

PC Requirements

Your PC should meet the following minimum hardware requirements:

- A 1 GHz or faster Intel Pentium (or compatible) CPU with MMX/SSE2 instruction set support.
- One open PCI slot if you are using a PCI-bus Cognex frame grabber or a GigE camera adapter
- An appropriate port for a direct-connect camera (Gigabit-Ethernet for GigE Vision)
- PCI, PCI Express (recommended), or AGP video adapter
  - CVL requires a minimum of 8 MB of video card memory, which provides limited functionality.
  - Cognex recommends a standalone video card that plugs into a motherboard slot over a chip-based video system integrated with the motherboard.
  - Cognex recommends an AGP or PCI Express x16 standalone video card with at least 16 MB of memory.
  - The video card must support DirectX 8.1 or later.

Cognex also makes the following recommendations:

- The PC should have enough RAM that your CVL application does not need to page to disk.
- Set the PC's BIOS to support a non plug-and-play operating system.
- Set the Windows desktop to 8-bit, 16-bit, or 32-bit color mode. The desktop must be in 16-bit or 32-bit mode to display color images.
- Cognex recommends a PC with bus-mastered DMA disk drivers, as opposed to PIO drivers for IDE (ATA) hard disks. Check with your motherboard vendor to determine if it provides bus-mastered IDE drivers.
- If your PC has 64 MB of RAM or more, perform the following steps to disable the paging of the kernel and drivers to keep them in memory for faster access:
  - Make a backup of your current registry settings.
  - Open the registry editor and navigate to:
    HKLM/SYSTEM/CurrentControlSet/Control/Session Manager/Memory Management
  - Set the key *DisablePagingExecutive* to 1 to disallow kernel and driver paging.
  - Reboot your PC.

Operating Systems

CVL 9.0 supports development and deployment on single or multiprocessor machines using native languages (English, Japanese, German, Korean, and Simplified Chinese) with the following Windows 64-bit operating systems:

- Windows 7 Premium
- Windows 7 Professional

- Windows 7 Ultimate (64-bit)
- Windows 10
- Windows 10 IoT Enterprise

Be aware this may be the last CVL release to support Windows 7.

Supported Microsoft Visual Studio Development Environments

CVL 9.0 supports developing and deploying vision applications using the following development environments:

Development Environment	Compiler Version	Service Pack
Microsoft Visual Studio 2015	Visual C++ v14.0	Update 3
Microsoft Visual Studio 2017	Visual C++ v14.16	

CVL ships with both Microsoft Visual Studio 2015 and Microsoft Visual Studio 2017 CVL C++ libraries.

CVL 9.0 is an unmanaged C++ SDK , and only supports applications built with Microsoft Visual Studio in unmanaged applications. CVL 9.0 does not support applications built with managed .NET (using the */clr* compiler switch); also known as *mixed mode*.

Supported Acquisition Platforms

CVL 9.0 supports the following acquisition platforms:

- Cognex MVS-8511
- Cognex MVS-8511e
- Cognex MVS-8514
- Cognex MVS-8514e
- GigE Vision Cameras
- CFG-8724 series frame grabber
- Third-party Imaging Devices that conform to the Imaging Device Interface, described in the *CVL User’s Guide*.

New Features in This Release

CVL 9.0 is a general availability release supporting new features.

New USB Security Keys

This release requires a new Cognex USB security key attached to your hardware platform to ensure the software is properly licensed. None of the security options supported in previous releases will work with this CVL release. Your existing Cognex frame grabbers will still perform their respective image-acquisition functions, as in previous releases.

Cognex supports security keys on a time-limited basis for application development as well as deployment keys for use with applications in a production environment.

Contact your Cognex sales representative for more information.

Visual Studio 2017

This release adds support for developing and compiling applications using Microsoft Visual Studio 2017 (v. 15.10).

LineMax Tool

This release adds support for the LineMax tool, which locates all candidate edge points in an image and then fits the best possible line segment based on criteria that you specify. The LineMax tool supports 8-bit and 16-bit images.

The LineMax tool is more robust than the Fitting tool, which requires precisely placed calipers in order to find individual edge points. Cognex recommends the LineMax tool for new applications, and continues to support the Fitting tool for existing applications.

The CVL Class Reference included in this release does not include a description of the API for the LineMax tool. Locate the header file and its source code comments at *%VISION_ROOT%*\defs\ch_cvl.

See the CVL Vision Tools Guide for details on the LineMax tool.

CFG-8724 Frame Grabber

This release supports the CFG-8724 4-port GigE Vision frame grabber with integrated security licensing. The CFG-8724 allows you to connect up to four GigE Vision cameras to a single PC. In addition, multiple CFG-8724 frame grabbers can be installed in the same PC. The CFG-8724 can also store your Cognex software licenses to make sure your Cognex software is properly authorized for use. Be aware that CVL will only read the software licenses from a single CFG-8724. See the CVL User's Guide for more information on Security and Version Information.

Cognex offers the CFG-8724 as a direct replacement for the CFG-8174e, which will not be available in the future.

If you install your Cognex vision software after installing the CFG-8724, be aware you must turn off the power to your computer after the software has been installed. Performing only a warm reboot can prevent the CFG-8724 from functioning correctly.

Support for 16-bit Image Display

This release adds support for displaying 16-bit images of type **ccPelBuffer<c_UInt16>** using the same API you use for 8-bit images of type **ccPelBuffer<c_UInt8>**. CVL automatically maps the 16-bit range of grey pixel values to an 8-bit range for display.

Be aware that 16-bit images with a wide range of grey values will maintain that range in the display, while 16-bit images with a narrow range will see that range expanded in the display.

For example, if your 16-bit images contain pixels with dark and light grey values, CVL will display that image in an 8-bit range of dark and light values. If your 16-bit images, however, contain exclusively dark grey values, the pixel values at the higher end of the range will appear as light grey values in the 8-bit display.

Refer to the header file `%VISION_ROOT%\defs\ch_cvl\display.h` for the CVL image display API.

CNLSearch Model Size

This release supports a new 4x4 model size for a CNLSearch tool. Previous CVL releases supported a minimum CNLSearch model size of 8x8.

16-Bit Image Support for More Vision Tools

This release adds 16-bit image support for the following vision tools:

- Caliper
- Image Warper
- Filter masks for Gaussian sampling
- Masks for a PelSet tool

Compatability Notes

Refer to the following sections for issues related to using Cognex hardware and CVL 9.0 software.

Large Image Compatibility Issues

This section describes compatibility issues related to support for large images.

Affine Sampling Tool Interface Changed

In CVL 6.7 CR1 and earlier releases, the members used to specify the number of samples in the X- and Y-direction for the affine sampling tool (**ccAffineSamplingParams::xNumSamples** and **ccAffineSamplingParams::yNumSamples**) were of type **c_Int16**.

In order to support sampling large images, in releases starting with CVL 6.7 CR3, the number of samples is stored using a **c_Int32** value.

The setter functions in **ccAffineSamplingParams** are simply changed to take a **c_Int32** value; this does not cause incompatibility with existing code because the C++ compiler promotes the 16-bit value to a 32-bit value. The getter functions, however, cannot be changed without becoming incompatible with existing code. CVL 6.7 CR3 added two new members to **ccAffineSamplingParams**:

- **ccAffineSamplingParams::xNumSamples32()** returns the number of samples in the X-direction.
- **ccAffineSamplingParams::yNumSamples32()** returns the number of samples in the Y-direction.

The existing getter functions that return 16-bit values are deprecated, and they will now throw an error if the number of samples is greater than or equal to 32768.

Not all Affine Sampling Modes Supported for Large Images

Only the following sampling modes are supported by the Affine Sampling Tool for input or output images with more than 32768 pixels in either the X- or Y-dimension:

- *ccAffineSamplingParams::eNone*
- *ccAffineSamplingParams::eBilinearAccurate*

The remaining modes (*eBilinearApprox*, *eHighPrecision*, and *eBilinear*) are not supported; if you specify one of the unsupported modes and either the input or output image has an X- or Y-dimension greater than 32768, the tool will throw an error.

Edge Tool Interface Changed to Support Large Images

In CVL 6.7 CR1 and earlier releases, the Edge tool stored the whole-pixel locations of sub-pixel edges (edgelets) using 16-bit values. This meant that any edgelet that had whole pixel positions outside the range -32768 to +32767 could not be represented.

Starting with CVL 6.7 CR3, the Edge tool allows you to specify the storage method for edgelets.

- If you request *ccEdgeletDefs::eRequestContiguous*, the edgelets will be stored in a single contiguous block. If no edgelet has a whole-pixel location with an image X- or Y-coordinate outside the range -32768 through +32767, the edgelets are stored in a vector of **ccEdgelet** and accessed through **ccEdgeletSet::edges()**. If any edgelet has a whole pixel location with an image X- or Y-coordinate that is outside the range -32768 through +32767, the edgelets are stored in a vector of **ccEdgelet2** and accessed through **ccEdgeletSet::edges2()** or through the **ccEdgeletIterator** or **ccEdgeletIterator_const** classes obtained by calling **ccEdgeletSet::begin()**.

- If you request *ccEdgeletDefs::eRequestEdges2*, the edgelets are stored in a vector of **ccEdgelet2** and accessed through **ccEdgeletSet::edges2()** or through the **ccEdgeletIterator** or **ccEdgeletIterator_const** classes obtained by calling **ccEdgeletSet::begin()**.
- If you request *ccEdgeletDefs::eRequestEdgesAndOffsets*, the edgelets are stored in one or more separate blocks of memory, where each block contains an array of **ccEdgelet** objects and a pair of 32-bit offset specifiers. Edgelets are accessed using the **ccEdgeletIterator** or **ccEdgeletIterator_const** classes. You obtain one of these objects by calling **ccEdgeletSet::begin()**.

Multiple blocks are only used when at least one edgelet has a whole pixel location with an image X- or Y-coordinate that is outside the range -32768 through +32767. If only a single block is created, you can use both **ccEdgeletSet::edges()** and the **ccEdgeletIterator()** methods for accessing the set's edgelets.

Because the **ccEdgeletIterator** or **ccEdgeletIterator_const** classes obtained by calling **ccEdgeletSet::begin()** can be used to access edgelets regardless of the storage method, Cognex recommends that you use this interface.

CDB Files Created by Certain CVL and VisionPro Versions are Invalid

Any CDB or IDB files created using the IDBEdit program supplied with CVL 6.8 CR1, CVL 6.8 CR2, or VisionPro 6.0 are unreadable by the IDBEdit program supplied with earlier versions of CVL and VisionPro.

The IDBEdit program supplied with CVL 9.0 (and with VisionPro 6.1 or 6.2) can be used to repair invalid CDB or IDB files. Simply open the affected file using the newer IDBEdit program, then select **File->Save As...** and save it under a different name. The newly saved file is readable by *all* versions of IDBEdit.

CDB Files with Large Images are Not Compatible with Earlier CVL Releases

If you use a CVL application built with CVL 6.8 CR3 to save an image that is larger than 32768 pixels in either the X- or Y-dimension, attempts to load that image into a CVL application built using an CVL releases 6.7 CR1 and earlier will produce a 1x1 pixel image.

Device Driver Backward Compatibility

Device drivers supplied with CVL 9.0 are backward compatible with the versions listed in the table below. Backward compatibility in this sense implies that the specified older CVL version will continue to operate when using the newer driver from this release.

Frame Grabber or Camera	Earliest CVL Version Compatible with 8.0 Driver (Windows 7)	Earliest CVL Version Compatible with 8.0 Driver (Windows 8.1 and 10)
MVS-8511 MVS-8514 MVS-8511e MVS-8514e	CVL 7.0 (64-bit)	CVL 8.0
GigE Vision Performance Drivers	CVL 7.1 CR2 (64-bit)	CVL 8.0

MVS-8510 Series Frame Grabber Compatibility Issues

This section describes compatibility issues that affect the use of the MVS-8510 and MVS-8510e frame grabbers.

Contrast and Brightness on MVS-8510 Series Frame Grabbers

Contrast and brightness settings are made in CVL with functions of the **ccContrastBrightnessProp** class. Within the commonly used range of 0.05 to 0.95 for contrast and 0.20 to 0.80 for brightness settings, the grey level values in acquired images will vary slightly depending on which board you are using.

- At the same contrast and brightness settings, the MVS-8511, MVS-8511e, MVS-8514, and MVS-8514e frame grabbers produce grey level values that may be up to 10 grey levels higher or 6 grey levels lower than the pixel values produced by the (unsupported) MVS-8501 and MVS-8504 frame grabbers with the same input signal. The median difference in grey values across a typical range of inputs is approximately 0.
- At the same contrast and brightness settings, the MVS-8511, MVS-8511e, MVS-8514, and MVS-8514e frame grabbers produce grey level values that may be up to 4 grey levels higher or 12 grey levels lower than the pixel values produced by the (unsupported) MVS-8500Le and MVS-8504e frame grabbers with the same input signal. The median difference in grey values across a typical range of inputs is approximately -2 (the values produced by the newer frame grabbers are lower).

You may be able to minimize the grey value differences for a particular range of input values by adjusting the brightness and contrast settings.

Software Addresses with Cable 300-0406

Cognex cable 300-0406 is intended for connecting a Sony DXC-390 color camera plus one monochrome analog camera to the MVS-8514 and MVS-8514e.

When using this cable, the R, G, and B cable branches connecting to the color camera consume camera positions 0, 1, and 2. This means that the monochrome camera connected to the fourth cable branch has a software address of 3. Use this address in your CVL code to control the monochrome camera when using cable 300-0406.

Strobe Polarity Reversed When Using TTL-Only Cable Option

For two of the three parallel I/O cable options for all MVS-8510 series frame grabbers, the strobe lines are opto-isolated. However, when you use the TTL pass-through cable option (cable 300-0390 and the TTL connection module, 800-5818-1), the four strobe lines have reversed polarity, compared to the same lines when opto-isolated.

Thus, when using strobes on the all-opto (300-0389) or half and half (300-0399) cable options, the information about **ccStrobeProp::strobeHigh()** in the *CVL Class Reference* is accurate as written:

```
void strobeHigh(bool polarity);
```

When *polarity* is true, the active pulse is set to high. If false, sets the active pulse to low.

However, when using strobes on the TTL pass-through cable option, 300-0390, the polarity information is reversed:

```
void strobeHigh(bool polarity);
```

When *polarity* is true, the active pulse is set to low. If false, sets the active pulse to high.

Open Bugs

This section lists open bugs in this release.

Acquisition

This section describes open issues concerning image acquisition in CVL 9.0.

RGB Pack Function Copies Transform Incorrectly

The **cfRGBPack()** function does not copy the transform correctly if the destination pel buffer is preallocated and not the same size as the source pel buffer.

Workaround

Do either of the following:

- Copy the transform manually after calling **cfRGBPack()**
- Ensure the destination pel buffer you pass to **cfRGBPack()** is unbound.

ccLightProp::LightsOff() Does not work.

Calling the **ccLightProp::lightsOff()** function has no effect. You can work around this bug using the following code:

```
fifo->properties().lightPower(0.0);

fifo->properties().auxLightPower(0.0);

fifo->prepare(0.0);
```

MVS-8500 Acquisition

This section describes open issues concerning MVS-8500 acquisition in CVL 9.0.

Using 8500 Frame Grabbers, ExtSync Acquisition Can Lose Lock when Acquiring from Video Signals Containing Color Information

If a video signal containing color information (like NTSC or PAL) is supplied to the 8500 frame grabber, ExtSync acquisition can fail depending upon the type of scene the camera is looking at. The problem is due to the color burst and the color information that is encoded as a carrier with a frequency of ~3.58 MHz(NTSC) and 4.43MHz(PAL) as part of the RS170 or CCIR signal. This color information can interfere with the 8500 detecting HSync pulses correctly and causes incorrect lock.

Workaround

The workaround is to remove the color information from the video signal before feeding it into the 8500. One easy way to do this is to get a converter from composite video to S-Video (Y/C). Once converted, the Y component does not have any color information.

Assertions Can Occur with Sony DXC-390 Camera

Assertions can occur when using an external sync source, if the camera cable is disconnected while acquisitions are underway.

MVS-8500e Acquisition Error on Quad-Core CPU with Queued Software Triggers

Under certain specific system configurations, the MVS-8500e frame grabber may experience occasional isAbormal() acquisition failures. These failures, which occur in approximately 1 out of 250,000 to 1,000,000 acquisitions, occur when all of the following conditions are met:

- The application is running on a quad-core CPU with all cores enabled.
- The application is using manual (software) triggering.
- The application queues multiple acquisition starts (calls **ccAcqFifo:start()** multiple times before calling **ccAcqFifo::completeAcq()**).
- The application asynchronously acquires images from 3 or 4 cameras simultaneously (not using master-slave acquisition).
- All of the cameras are high-speed cameras such as the Sony XC-HR70 or XC-HR90.

The acquisition error is transient, it affects only a single acquisition, and it is self-recovering. This error does not affect any acquisition that uses an external trigger; it only affects software-initiated acquisition that uses multiple queued starts.

Workaround

You can prevent this error from happening by using any of the following workarounds:

- Run your application on a single- or dual-core CPU.
- If you are using a quad-core CPU, disable one core.
- Do not call **ccAcqFifo:start()** more than once before calling **ccAcqFifo::completeAcq()**.
- Use an external trigger to acquire images.

Display

This section describes issues concerning image display.

Ellipse Arcs Drawn Incorrectly

Ellipse arcs are drawn incorrectly when the ellipse centers are far outside the display region. Using a radius value that is larger than **sizeof(short)** would result in this effect. This problem is caused by limitations in the Microsoft GDI graphics drawing code, and exists on all supported Windows platforms (2000, and XP).

Display Zooming Inconsistent When Using ccDisplayConsole

The maximum zoom that you can use with **ccDisplay::mag()** is 65,536. The minimum zoom depends on the size of the image; for example, for a 640x480 image the minimum is -60. This can cause unwanted behavior when zooming using **ccDisplayConsole**. As displayed in the status bar at the bottom of the display console window, zooming out (using the toolbar buttons for zooming) produces magnifications of 1/2, 1/4, 1/8, 1/16, 1/32, and then 1/60, but zooming in from here produces magnifications of 1/30, 1/15, and so on.

Parenting Does Not Work with ccUIManShape-Derived Children

Parenting does not work as expected when the child shape in a parent-child relationship is derived from **ccUIManShape**. If you drag the parent shape, only the handle of a **ccUIManShape**-derived child will go with its parent; the child shape itself will not move.

Parenting Does Not Work with ccUIGenAnnulus as Parent

In general, parenting works as expected when the parent shape in a parent-child relationship is derived from **ccUIManShape** and the child shape is derived from **ccUIPointShapeBase**. However, **ccUIGenAnnulus** (a **ccUIManShape**-derived shape) is an exception and does not work appropriately as a parent.

stopLiveDisplay() Hangs if FIFO Passed to startLiveDisplay() Goes Out of Scope

You must ensure that the FIFO passed to **ccDisplay::startLiveDisplay()** is not destroyed before calling **ccDisplay::stopLiveDisplay()**. For example, if you create a FIFO in a function, call **startLiveDisplay(fifo)** passing in this FIFO, and then call **stopLiveDisplay()** after the function in which the FIFO was created returns, the **stopLiveDisplay()** method will hang and not return.

No Support for Multiple Monitors

CVL does not provide an API to allow you to target a display console window to a specific monitor if the system has more than one monitor.

Cannot Fill Ellipse Annulus Sections

The **ccGraphicProps::penfill()** function does not work with **ccEllipseAnnulusSection** objects.

Setting Fill for ccUIGenAnnulus Fills Entire Graphic

Setting the fill property to true for a **ccUIGenAnnulus** fills the entire shape, not just the annulus.

Blank Display, Sony DXC-390 and 8-Bit Desktop

A test case was set up to run live display using the MVS-8504, a Sony DXC-390 camera, the "Sony DXC-390 640x480 IntDrv CCF" video format, and **ccAcqFifoPtrh**. When displayed onto an 8-bit Windows desktop, displayed images were observed to be blank.

Workaround

When using this configuration, view displayed images using a 16-bit or 32-bit desktop color depth.

Drivers

This section describes open issues concerning drivers in CVL 9.0.

Inherent PC Interrupt Latencies Can Cause Acquisitions to Be Missed at High Speeds

During testing, Cognex has observed some PCs to take a relatively long time (5-6 ms) to respond to interrupts. This long latency does not cause problems with most CVL applications. However, when the acquisition rate is higher than one acquisition in every 6 ms, this prolonged delay could cause interrupts not to be serviced on time, resulting in missed acquisitions. The long latency of some PCs can also cause problems with other devices that require timely interrupt response.

The followings PCs are known to exhibit this problem:

IBM Intellistation Z Pro: model # MT-M 6221-22U (2.66 GHz Xeon single CPU)

IBM Intellistation M Pro: model # MT-M 6219-42U (3.06 GHz P4 single CPU)

This problem has implications for acquisition with a region of interest (ROI). When an image is small, the frame rate can be higher than usual, potentially causing interrupt overrun. The noted problem can therefore occur when using cameras that support ROI control and decimation with the MVS-8500, which supports partial scan (with the same end result as ROI).

Development Framework

This section describes open issues concerning the CVL development environment in CVL 9.0.

Cannot Use Both Old and New iostream Libraries

CVL uses the newer iostream library from the std namespace; therefore, including any CVL headers also includes the new iostream headers. Using the old iostreams library, or another product (such as MatLab) that uses the old iostreams library, may result in compilation errors when a single source file attempts to include the headers for both libraries. The problem appears to stem from the Windows ios class.

This is both a compile-time problem for certain classes, and a potential run-time problem for others, because enums available in both classes have different values. If your software currently uses the old iostream library, Cognex recommends updating to the new iostream library. If your software requires the use of both the old and the new iostream libraries, Cognex recommends always including the new iostream library first.

When using stdafx.h (so every source file includes it first), you can just add lines such as the following to the top of stdafx.h:

```
#include <ch_cvl/defs.h>

#include <fstream>

#include <iostream>
```

The first include of defs.h is to take advantage of the pragma in defs.h, which disables the "debug symbol name too long" compiler warning (C4786). This pragma must occur before the file that generates the warning is included, or the warnings will occur. You could also use this pragma directly instead of including defs.h.

Sample Code

This section describes open issues concerning sample code in CVL 9.0.

Pmalign4 Sample Crashes During Runtime

The pmalign4 sample application shipped with CVL (under $VISION_ROOT\sample\cvl) crashes with a “plate.idb can’t be opened” error when run from either Windows Explorer or a Command Prompt command line with the current directory set to the location of the executable. The problem is that the relative path to the plate.idb file is not set correctly during installation.

Workaround

Either build and run the sample inside Visual Studio, or move the plate.idb and plate.dxf files to the \cvlproj\ directory.

Serial I/O

This section describes open issues concerning serial I/O in CVL 9.0.

Serial I/O: Read Operation Ignores Timeout Until Receipt of First Byte

ccSerialIO::read() blocks indefinitely until receiving at least one byte across the serial port, even if a non-infinite timeout is specified. After the serial port receives the first byte, **read()** works correctly from that point forward.

Shape Models

This section describes open issues concerning shape models in CVL 9.0.

Attempting to Record Unbound Pel Buffer Can Cause Crash

The **ccShapeModel::record()** method requires the supplied pel buffer to be bound. However, it does not throw and will eventually cause the application to crash because of attempted division by 0 if an unbound pel buffer is supplied. For example, the following call will end in a crash. This is because the default-constructed 8-bit **ccPelBuffer** supplied as the second argument is unbound by default.

```
ccShapeModel::record(diagObj, ccPelBuffer<c_UInt8>(), shape,

gprops, gprops, gprops, cmT("Synthetic model"));
```

Workaround

Always supply a bound pel buffer with non-zero width and height as the second argument to the **ccShapeModel::record ()** method.

Caliper Tool

This section describes open issues concerning the Caliper tool in CVL 9.0.

Caliper: High Precision Mode Identifies Extra Edges when Clipping Enabled

When Caliper runs in eHighPrecision mode, it finds extra edges when the affine rectangle is set to clip the image. For example, when a projected image (corresponding to a reverse read of the original image) in the result reads 200 214 198 197 194 192 190 185 183 111 ..., the pixel value 214 causes extra edges to appear on the right side of the image when the contrast threshold is set to 5. If you change the mode to eBilinear, the projected image reads 198 198 197 196 193 191 189 184 176 107 ..., which represents the original image well.

Workaround

The nature of high precision mode is to ring on edges. The workaround is to use the bilinear mode.

Caliper: Result Graphics not Drawn if Clipped

If you specify ccCaliperDefs::eDrawProjFilter in the drawMode argument to **ccCaliperOneResult::draw()**, the graphics (which show a graphical representation of the projection image and the filtered projection image) are not displayed if the projection region is clipped by the input image. To prevent this from happening, supply an input image that fully encloses the projection region.

Caliper: Projected and Filtered Images Not Drawn when Clipping Enabled

Regardless of the projection mode, the ccCaliperResultSet::draw() method does not draw the projected image and the filtered image as specified when clipping is enabled. For example, the following code will draw only the label and the region.

```
clpResultSet.draw(graphList, ccCaliperDefs::eDrawLabel

      | ccCaliperDefs::eDrawProjFilter

      | ccCaliperDefs::eDrawProjectionRegion

      | ccCaliperDefs::eDrawStandard);
```

It is then not convenient to keep track of the project image.

Edge Detector Tool

This section describes open issues concerning the Edge tool in CVL 9.0.

Edge: ccDiagObject Recording Does Not Perform Deep Copy by Default

Whether defaulting to deep copy or forcing shallow copy with diagFlags, it is possible to draw into the source image, magnitude image, and angle image after **cfEdgeDetect()** has recorded a diagnostic object, and for those modifications to show in the diagnostic viewer.

Histogram Tool

This section describes open issues concerning the Histogram tool in CVL 9.0.

cfPelHistogram() Behavior Not Consistent with Header File Comment

The comment for the **cfPelHistogram()** function in the <ch_cvl/histo.h> header file inaccurately states that the function resizes and zeroes out the passed-in vector if that vector is of size 0. This function does not resize or zero a vector that has no elements.

ID Tool

This section describes open issues concerning the ID tool in CVL 9.0.

For Composite Codes, the Subtype of the 2D Component Cannot be Determined when Combining Results

If you specify true for **ccSymbologyParamsComposite::combineResults()**, you cannot determine the subtype of the 2D component (CC-A or CC-B). Only the subtype of the linear part is returned.

OCV Tool

This section describes open issues concerning the OCV tool in CVL 9.0.

Enabling Rotational Uncertainty Causes Position Verification to Fail Following False Verification of Confusing Character

The OCV Tool can exhibit anomalous behavior when trying to verify a char 'A' in the image with the current key set to '4' (or vice versa) and character rotational uncertainty set to greater than zero. These characters can be easily confused when rotation is allowed.

When rotational uncertainty is set, the tool usually verifies an A-4 mismatch, though with a low score, then all positions following the mismatch fail to verify. This behavior has been observed with three different fonts (Courier, Lucida Sans, Terminal) and with a range of uncertainties (5 to 45 degrees). For some uncertainties, the Courier font may verify one or two positions beyond the mismatch and then start failing. This behavior is consistent regardless of both where the mismatch occurs (whether near the beginning or end of the string) and the number of mismatches.

Space (Key 32) Not Usable as Confusion Override

The OCV tool assumes that a blank character is never confusable with an actual character. So, for example, the tool never checks to see if a character that is expected to be an ‘a’ looks blank. The opposite is not true, however. The tool does check to see whether a character that you expect to be blank looks like an ‘a’ (if ‘a’ is considered confusable).

Line Arrangement Bounding Box Excludes Wildcards at End of Line

The OCV line arrangement bounding box does not include a wild card that is at the end of the line.

Diagnostic Shows Red Cross at Final Pose When Line Not Found

When a line is completely not found (that is, all characters are not found and the score is 0.0), the “Line Arrangement at Final Pose” diagnostic draws a red cross of the line arrangement pose. The cross is drawn in client coordinates.

PatInspect

This section describes open issues concerning PatInspect in CVL 9.0.

PatInspect: Incorrectly Reported Extra or Missing Edges

PatInspect boundary difference mode may incorrectly report features that lie close to the edge of the inspection region as missing or extra. This happens, for example, when the inspection region coincides with the training region, in which case all the boundary features should be of the matching type. This behavior has been observed in both CVL 5.4 backward and non-backward compatibility mode.

Workaround

Make sure the rectangular inspection region provides at least five pixels of padding around the pattern you want to train.

PatMax

This section describes open issues concerning PatMax in CVL 9.0.

PatFlex Ignores Nominal DOF Values in Runtime Parameters

The PatFlex algorithm ignores any nominal DOF values set in cc_PMRunParams using the nominal() setter.

Workaround

To use nominal DOF values with PatFlex, enable the given DOF and set both its high and low zone values to the desired nominal value. For example, instead of:

```
runParams.zoneEnable(0);
```

```
runParams.nominal(ccPMAlignDefs::kUniformScale, 2.0);
```

use

```
runParams.zoneEnable(ccPMAlignDefs::kUniformScale);
```

```
runParams.zone(ccPMAlignDefs::kUniformScale, 2.0, 2.0);
```

Wafer Pre-Align Tool

This section describes open issues concerning the Wafer Pre-Align tool in CVL 9.0.

Flat Length Computation Inaccurate in NotchMax Mode

In NotchMax mode, the flat length computation uses wafer area, inner diameter, and outer diameter in computing the flat length, which you can query with **ccWaferPreAlignResult::flatLength()**. However, there may be 1, 2, or even 3 flats per wafer. This can affect the area computation, which in turn affects the inner and outer diameter computations. This can make the returned flat length inaccurate. For this reason, NotchMax mode is not supported on flat wafers.

Limitations

A limitation is behavior that is imposed on CVL by hardware, operating system software, product architecture, or other external constraints. Workarounds may exist for some limitations.

Security

This section describes limitations concerning software security in this release.

ccSecurityInfo Not Supported In This Release

This release does not support the **ccSecurityInfo** API described in the CVL Class Reference.

Acquisition

This section describes general limitations concerning image acquisition.

Default Exposure is for Strobed Acquisition

The default exposure of 20 µs for all cameras is designed for strobed acquisition. If you use ambient lighting with a rapid reset camera such as the Sony XC-55, you may need to increase the exposure time to more than 1 ms.

Pulnix TM-6CN Static Video Format Created with Wrong Sync Model

The static video format for the Pulnix TM-6CN is instantiated with internal sync enabled, which is not correct for this camera and can produce odd images. The CCF version of the TM-6CN's video format correctly sets the sync model.

Workaround

When using the static video format, set the sync property manually in your code to cfSyncExternalIPLLComposite().

MVS-8500 Acquisition

This section describes limitations of MVS-8500 acquisition in CVL 9.0.

Acquisition Brightness Property Default Value

For 8504 and 8501 frame grabbers, the acquisition brightness property default value is changed to 0.5. Because of this, images acquired with this release may look different than acquired with a previous release using a different default value right after creating the acquisition fifo.

CFG-8704e Acquisition

This section describes limitations of the CFG-8704e frame grabber in CVL 9.0.

No Support For More Than Two CFG-8704e Frame Grabbers

Be aware you cannot use more than two CFG-8704e frame grabbers in the same computer.

Display

Display

This section describes limitations in displaying images.

Adding and Removing Many UI Shapes Takes a Long Time

Adding a large number of manipulable shapes to the display, or removing them from the display, one at a time in a loop can take a long time.

Workaround

Call **ccDisplay::disableDrawing()** before adding or removing manipulable shapes, and then to call **ccDisplay::enableDrawing(true)** after all shapes have been added or removed.

ccUIGenAnnulus::move() Implemented Incorrectly

The **ccUIGenAnnulus::move()** method moves a **ccUIGenAnnulus** to the location given by the supplied **cc2Vect**. This is incorrect behavior. It should adjust the position of the shape by the x- and y-values of the supplied **cc2Vect**.

Workaround

Retrieve the position of the front child object of the UI generalized annulus with **ccUIGenAnnulus::frontKid()**, add the offset to the result, and then call **ccUIGenAnnulus::pos(p)** passing the adjusted position as an argument.

Interpolated Zoom in High Precision Mode on 8-Bit Desktops

On 8-bit desktops, setting interpolated zoom to high precision mode by calling ccUITablet::interpolation(eHighPrecision) may cause colors outside the grey-scale range to be displayed.

Workaround

Use a non-8-bit display and/or bilinear rather than high-precision interpolation.

W2K Power Management Interferes with Live Video

If you are using Windows 2000, disable the power management feature, as described in the section Tuning System Configuration for Best Performance in the CVL User's Guide. In particular, make sure that your system is configured to never shut off the video monitor. If the power management feature is configured to shut off the video monitor, your CVL application may have problems attempting to display graphics with live video.

Ellipses Archived in CVL 5.3 Cannot be Read in Newer CVL Versions

The internal structure of the ellipse geometry class (**ccEllipseGeom**) was changed in CVL 5.4. Applications built against CVL 5.4 or later that attempt to display ccUISketch lists containing **ccEllipse**, **ccEllipseArc**, **ccEllipseAnnulus**, or **ccEllipseAnnulusSection** objects that were archived in CVL 5.3 or earlier may hang.

Static Shapes Render Differently in Release and Debug Modes

Static shapes are rendered differently in release and debug modes. Depending on the geometry of the shape, slight differences may occur between which pixels are rendered in the two modes. Do not rely on the exact same pixels being rendered between debug and release builds.

Development Framework

Development Framework

This section describes limitations of the CVL development environment.

Project settings for Visual Studio for "treat wchar_t as builtin type"

Visual Studio uses a default of **true** for **treat wchar_t as builtin type**. Cognex uses Visual Studio defaults so your projects must also use **treat wchar_t as builtin type == true**.

Installation

This describes limitations you should be aware of when installing CVL.

Clicking in Command Prompt Window Pauses Uninstallation

When uninstalling CVL, if QuickEdit mode is enabled for command prompt windows and you click in the command prompt window that opens during uninstallation, the cursor goes into select mode and the uninstallation appears to have halted. Right click in the command prompt window to exit select mode and resume the uninstallation.

Discrete I/O

This section describes limitations of discrete I/O in CVL 9.0.

Parallel I/O: Output Polarity on External I/O Module Inconsistent

The external configuration of the universal parallel I/O card (P/N 800-5726-3) is used with the external I/O module (P/N 800-5712-2) to provide opto-isolated connection points for parallel I/O lines. When used with the MVS-8120, the polarities of output lines 4 through 7 on the external I/O module are inverted, compared to output lines 0 through 3, and compared to opto-isolated output lines on other Cognex hardware.

The CVL Class Reference correctly states that the **ccOutputLine::set(true)** member function sets the no-current-flowing state for opto-isolated output lines. This is true for output lines 0 through 3 on the external I/O module, and for other Cognex boards. However, for output lines 4 through 7 of the external I/O module when used with the MVS-8120, the meanings of **set(true)** and **set(false)** are reversed.

Actual Pulse Width Can Be Shorter than Specified

In some cases, the actual pulse signal can be shorter than the width parameter supplied to **ccOutputLine::pulse()**. For example, if you create a 500 ms pulse as follows:

```
oline.pulse(true, 0.5, false); // low-to-high transition,

                              // 0.5 second width, non-blocking
```

the resulting pulse may actually be as short as 490 ms in duration.

This behavior is a known limitation of operating system timers, and affects all Cognex hardware platforms that support the **pulse()** feature. This problem is also known to occur more frequently when making simultaneous **pulse()** calls on multiple parallel output lines.

Workaround

If your application requires an absolute minimum expected pulse duration, supply a width value to the **ccOutputLine::pulse()** function that is 1 ms longer than the actual desired pulse width. This will bump actual the pulse width to the next interval that matches the granularity of the multimedia timer.

In addition, reducing the multimedia timer period to 1 ms using the Microsoft Win32 API call **timeBeginPeriod(1)** can improve pulse accuracy. However, if you use this method, be aware that modifying the timer will affect overall system performance because this will result in timer interrupts needing to be serviced more frequently.

To use **timeBeginPeriod()**, you must include the Win32 mmsystem.h header file and the winmm.lib library in your code. In Visual C++ 6.0, the header file is located in the VC98\include directory; in Visual C++ .NET (7.0), it is in the VC7\PlatformSDK\include directory. It can also be found among the Platform SDK and DDK include files.

For example, you can add the following code to your application to adjust the system timer accuracy:

```
#include <windows.h> // Add this to avoid compiler errors
```

```
#include <mmsystem.h>
#pragma comment(lib, “winmm.lib”)
timeBeginPeriod(1);
// Set pulse width and perform I/O operations in CVL
timeEndPeriod(1); // Call before application ends to return
                    // the multimedia timer to original state
```

See also related notes on modifying the system timer in the Displaying Live Images section of the Displaying Images chapter of the CVL User’s Guide.

Utilities

This section describes limitations and restrictions that apply to utility programs shipped with CVL 9.0.

CVL command-line utilities must remain in the CVL bin directory

CVL command line utilities such as Cogtool are normally run only under the direction of Cognex Technical Support. As of this release, the executable for these utilities must reside in the directory that contains the CVL-provided DLLs. This means that a utility's executable file (such as cogtool.exe) must remain in %VISION_ROOT%\bin\win32\cvl to produce correct results. If you copy a utility's executable file to another directory and run it from there, the tool will run, but will incorrectly report that there are no Cognex boards in the system, even if the Cognex bin directory is in the PATH.

This is a change from the CVL 6.2 behavior, in which you could copy a utility's executable file to another directory and run it from there.

Barcode

This section describes limitations of the Barcode tool in CVL 9.0.

Parameters Added to decode() Functions Break Backward Compatibility

Between CVL 5.5.0 and 5.5.1, the boolean parameter autoRetry, with a default value of true, was added to the argument list to several virtual functions of the **ccAcuBarCodeTool** class, specifically the **decode()** functions. As this alters the function signature, if you were overriding these functions using the old signatures in CVL 5.5.0, not only do you no longer override them in CVL 5.5.1 and later, your overrides will no longer be called. No compiler errors will alert you to the fact that the function signature has changed. When upgrading CVL applications from CVL 5.5.0 to 5.5.1 or later, make sure that your overrides of the **ccAcuBarCodeTool::decode()** functions match the latest function signatures.

Blob

This section describes limitations of the Blob tool in CVL 9.0.

Blob: Persisting Scene Description by Reference Leaks Memory

If you persist a **ccBlobSceneDescription** object by reference and then reload it from the archive, your application will leak memory. You can prevent this from happening by persisting **ccBlobSceneDescription** objects by pointer rather than by reference.

Calibration

This section describes limitations of the Calibration tool in CVL 9.0.

Checkerboard Minimum Tile Size

When performing feature correspondence calibration, the feature detector does not work reliably with tile sizes smaller than 15x15 pixels. For most applications, the checker size should be larger than 15x15 to avoid operating at the range limit.

Caliper

This section describes limitations of the Caliper tool in CVL 9.0.

Pixel Range of Filtered Clipped Projection is Incorrect

The pixel value range in the filtered clipped projection is not correct. The range in the filtered image is from -255 through +255 while it should be from -255*n through +255*n, where n is the number of samples in the projection. To get the correct range, rescale (multiply by n) the filtered image.

Gaussian Sampling

This section describes limitations of the Gaussian Sampling tool in CVL 9.0.

cfGaussSample() Produces Incorrect Result

The **cfGaussSample()** overload that uses a 16-bit destination pel buffer clamps result values to 32,767. Therefore, any values that would normally produce a result that is greater than 32,767 (for example, a scale setting of 255 with an input pel value greater than 128) produce incorrect results.

Workaround

Set the scale low enough that all results are less than 32,767.

Grey-Scale Morphology

This section describes limitations of the Grey-Scale Morphology tool in CVL 9.0.

NxM Morphology with Configurations other than 3x3 Kernel Is Not Fully Multi-core Optimized

The NxM Morphology variable-size kernel image tool with configurations other than 3x3 kernel is not fully multi-core optimized.

ID

This section describes limitations of the ID tool in CVL 9.0.

Only RSS Composite Symbols Supported for EAN.UCC Composite Symbology

The only EAN.UCC Composite symbols supported in this release are those using RSS Composite symbology.

acuRead

This section describes limitations of the acuRead tool in CVL 9.0.

End Characters Missing in User Defined Fonts

In user-defined fonts, end characters can be missing. This is true of both fixed and variable length modes. In fixed length mode, the last character becomes a space.

Workaround

Widen the ROI to allow more blank pixels at the beginning and end.

OCV

This section describes limitations of the OCV tool in CVL 9.0.

Characters Falsely Verified Despite Blank Space

If a position in a line arrangement is defined with a blank character among its current keys, and if that position is not blank in the runtime image and does not match any of the other current keys, then the OCV tool may incorrectly verify or confuse that position. A blank current key always returns a non-zero score regardless of what is actually in the search position at runtime. Scores returned for a blank key are generally low, but for some narrow characters such as '-' or ',' the score may exceed the accept or confusion thresholds and thus not fail verification.

Some examples of blank character scores are listed below:

Character	Score
-	0.423
.	0.422
‘	0.382
:	0.312
,	0.276

This occurs whether multiple or single current keys are defined for the positions, though an instance has been observed in which the score was lower with a blank plus another current key than with just the blank alone. This problem has to do with the design of the OCV tool, where blank and "normal" characters are treated differently. OCV does not check for the confusion of blank characters with anything, it simply looks at the blankness score to establish the state of the blank character. Hence small characters can be easily overlooked.

PatMax

This section describes limitations of the PatMax tool in CVL 9.0.

Synthetic Align Model Training Issues

Training PatMax using a shape with a bounding box that is significantly larger than 2Kx2K when projected to image coordinates using the training time transform will result in either an access violation or an application hang from within train().

Closed Bugs

This release contains fixes for customer reported issues in previous releases.

Brightness and Contrast Values Producing Errors

When acquiring images, values for brightness and contrast no longer produce errors when mapped to invalid equivalent settings for cameras that have a non-1 interval

Documentation Supplied with This Release

Your CVL installation includes the following titles available from the **Start->Cognex->CVL->Documentation**:

- CVL User's Guide

  An overview of CVL programming, including working with the sample code, acquiring and displaying images and graphics, and an introduction to vision tools and concepts.

- CVL Vision Tools Guide

  Descriptions of the functionality of each CVL vision tool, the parameters each accepts, and the results they generate.

- CVL Class Reference

  A class-by-class reference for the CVL C++ class library

- CFG-8724 Hardware Reference

  Details the CFG-8724 frame grabber, a PCI Express x4 compliant device for GigE Vision image acquisition.

Notes About Future Releases

Be aware a future CVL release will remove support for the following API:

- ccAcuSymbol* classes associated with specific 2D symbologies:
  - **ccAcuSymbolDataMatrixDefs**
  - **ccAcuSymbolDataMatrixLearnParams**
  - **ccAcuSymbolDataMatrixTool**
  - **ccAcuSymbolDefs**
  - **ccAcuSymbolFinderParams**
  - **ccAcuSymbolLearnParams**
  - **ccAcuSymbolQRCodeDefs**
  - **ccAcuSymbolQRCodeLearnParams**
  - **ccAcuSymbolQRCodeTool**
- ccAcuRead* classes associated with basic OCR reading and tuning functions:
  - **ccAcuRead**
  - **ccAcuReadDefs**
  - **ccAcuReadFont**
  - **ccAcuReadResult**
  - **ccAcuReadResultSet**
  - **ccAcuReadRunParams**
  - **ccAcuReadTuneParams**
- ccAcuBarCode* classes associated with the 1D symbology tools:
  - **ccAcuBarCodeCalibrationResult**
  - **ccAcuBarCodeDefs**
  - **ccAcuBarCodeResult**
  - **ccAcuBarCodeRunParams**
  - **ccAcuBarCodeTool**
  - **ccAcuBarCodeTuneParams**